Workgroup: Deep32 Technical Report
Published: 20 November 2026
Author: J. A. Cummings
Deep32

Deep32: Design of a Local Retrieval System for Agentic Network Research

Abstract

Deep32 lets an agentic language model working in a terminal, such as Claude Code or Codex, answer networking questions from a local corpus of RFCs and vendor documentation, and check every quotation against the stored text. Retrieval, verification and the index run on the user's own machines; the language model is the only hosted part.

This report describes the design of release 0.1.0-alpha.2: the corpus and index format, the hybrid retrieval pipeline, the eleven tools the agent calls, the three machines a deployment consists of, the protocols between them, the signed release format, and the evaluation behind the numbers it quotes.

Status of This Document

This is Deep32 Technical Report 1 (D32-TR-1), the design reference for Deep32 release 0.1.0-alpha.2, published by its author. It is not a product of the IETF or of any other standards body and has no standing there. It uses the RFC format because the system it describes is built around RFCs, and because the format is good at stating behavior precisely.

Each Deep32 release that changes the design updates this report under a new version of its DOI. Errors can be reported on the project's issue tracker.

This version: DOI 10.5281/zenodo.24326998.

Copyright Notice

Copyright (C) 2026 Jeffrey A. Cummings. This document is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Text quoted from RFCs remains subject to the IETF Trust's Legal Provisions Relating to IETF Documents.

Table of Contents

1. Introduction
   1.1. Problem Statement
   1.2. Scope
   1.3. Requirements Language
2. Terminology

3. System Overview
   3.1. Parts
   3.2. Release Images
   3.3. Two Ways In
   3.4. Trust Boundaries
4. Corpus
   4.1. Collections
   4.2. Document Manifest
   4.3. Acquisition and Conversion
   4.4. Lookup Tables
5. Index Generations
   5.1. Identifiers
   5.2. Bundle Layout
   5.3. Building
   5.4. Validation Gate
   5.5. Switching and Rollback
6. Retrieval Pipeline
   6.1. Query Planning
   6.2. Keyword Leg
   6.3. Dense Leg
   6.4. Fusion
   6.5. Reranking
   6.6. Folding
   6.7. Release Selection
   6.8. Delivery
   6.9. Degraded Operation and Admission
7. Agent Interface
   7.1. Transport
   7.2. Tools
   7.3. Quotation Check
   7.4. Answer Protocol
8. Workstation

8.1. Platform

8.2. Hardening

8.3. Exact Vector Search

8.4. Ask Window and Library

8.5. Install Status

9. Forge

9.1. Services

9.2. GPU Pack

9.3. PDF Conversion

9.4. Agreement Across Cards

10. Network and Pairing

10.1. Tailnet and Publication

10.2. Tokens and Keys

10.3. Pairing by Short Code

10.4. Transport Switch

11. The Ask Link (D32U/1)

11.1. Transport and Identity

11.2. Framing

11.3. Handshake

11.4. Channels and Flow Control

11.5. Pairing by Short Authentication String

11.6. Reverse MCP

11.7. The Agent Box

12. Distribution and Verification

12.1. Hosting

12.2. Signature Check

12.3. Download

12.4. Bills of Materials and Reproducibility

12.5. Doctor and Self-Test

13. Evaluation

13.1. Test Set

13.2. Metrics

13.3. Results

13.4. Limitations

14. Security Considerations

14.1. Hosted Model

14.2. Credentials

14.3. Pairing

14.4. Network Exposure

14.5. Prompt Injection

14.6. Supply Chain

14.7. Host Considerations

15. IANA Considerations

16. References

16.1. Normative References

16.2. Informative References

Appendix A. Sections and Decisions

Appendix B. Changes from 0.1.0-alpha.1

Appendix C. Implementation Status in 0.1.0-alpha.2

Acknowledgments

Author's Address

1. Introduction

Language models get networking specifics wrong in ways that are hard to spot without opening the source document. Deep32 gives an agent a local corpus, a retrieval pipeline tuned on that corpus, and tools that check what the agent quotes. The agent still writes the answer; Deep32 puts the evidence where the agent can find it and read it in full, and checks quotations mechanically. The arrangement is the usual retrieval-augmented one [RAG], with two retrievers instead of one and the retrieval exposed to the agent as tools.

This report covers how the system is built. The reasons for each choice, with the alternatives I ruled out and the measurements behind them, are in the Deep32 Decision Record [D32-DR]; this report cites its entries as DR-nn. The software, images and this report are archived with the release [DEEP32].

Deep32's code is licensed under the Apache License, Version 2.0 [APACHE-2.0], with a NOTICE file that names its author and goes with every copy. The two parts that use MuPDF, the Forge's PDF converter and the Workstation Library's renderer `e26-render`, are licensed under AGPL-3.0-or-later [AGPL-3.0] (DR-110). This report and the other documentation are under CC BY 4.0.

1.1. Problem Statement

Most errors I see in model answers about networking fall into four groups:

Vendor default given as the standard: "The BGP hold time is 180 seconds." That is the Cisco default; [RFC4271] suggests 90 seconds.

Obsolete RFC cited as current: [RFC1771] quoted for BGP-4 after [RFC4271] replaced it.

Command from the wrong release: Syntax from one software train given for another.

Invented quotation: "The RFC states ..." followed by text that is not in the RFC.

Two of these can be checked mechanically: an RFC's status is a lookup, and a quotation either occurs in the stored text or it doesn't. The other two depend on the agent noticing a difference that the corpus makes visible. Deep32 returns RFC text and vendor text as separate sources, tags every passage with its platform and release, and has the agent follow a fixed protocol (Section 7.4) that compares them.

1.2. Scope

In scope: the corpus, the index and its build, retrieval, the verification tools, the deployment of the three parts, the network protocols between them, the release format, and the retrieval evaluation.

Out of scope: the language model, which its provider hosts, and the quality of the answers it writes. Retrieval quality is measured (Section 13); answer accuracy has not been measured yet, and this report makes no claim about it.

1.3. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In this report they state the behavior of the released implementation. A reimplementation of one of the protocols in Sections 10.3, 11 and 12 interoperates only if it follows the same rules.

2. Terminology

Workstation: The machine or virtual machine that holds the index and runs the search engine, the MCP gateway, the Ask window and the Deep32 Library. It runs OpenBSD 8.0.

Forge: The machine with the NVIDIA graphics card. It runs the embedding and reranking models in Docker, and optionally the PDF converter and the index builder.

Headend: The PC where the agent (Claude Code or Codex) runs, with the Deep32 research skill and, optionally, the Ask relay.

Tailnet: The user's Tailscale network. All three parts sign in to it with the user's own account.

Engine: The search service: hybrid search, section reader, identifier lookup and quotation check.

Gateway: The stateless MCP server in front of the engine.

Collection: A group of documents of one kind, named by its `doc_type`: `rfc`, `cisco`, `powershell` or `system`.

Opt-in collection: A collection that searches, greps and listings skip unless the request names it.

Passage: The unit of retrieval, also called a chunk: a span of one document within one section.

Hit: One passage in a search result, with its citation and a text window.

Generation: One complete, immutable build of the index. The engine serves exactly one at a time.

Pairing: The one-time step that introduces two parts to each other and leaves each holding the other's certificate pin.

Pin: The SHA-256 digest [FIPS180-4] of a peer's DER-encoded certificate, stored at pairing and used as the peer's only trust anchor.

Image: One of the three release files (disc 1, 2 and 3).

3. System Overview

3.1. Parts

A deployment has three parts on one tailnet (Figure 1). They can share a machine or sit on three; the protocols and the configuration are the same either way.

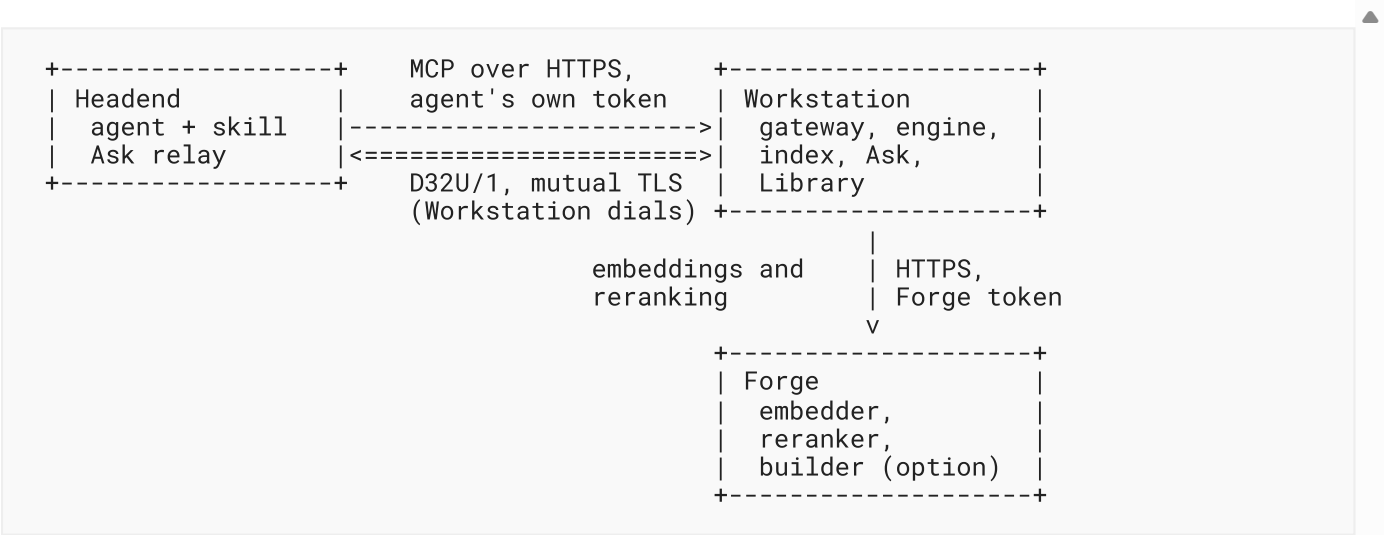


Figure 1: Deployment Topology

The Workstation holds a whole generation and runs the engine locally. It calls the Forge only to embed a query and to rerank candidates, so it still answers keyword searches when the Forge is off (DR-50, DR-58).

3.2. Release Images

Image	Contents	Installs
Disc 1	Bootable OpenBSD installer with the Deep32 system	Workstation
Disc 2	Corpus, a prebuilt generation, and the Deep32 Library	Content, imported by the Workstation
Disc 3	GPU pack: a shared core, one embedder add-on per card class, and an ingestion add-on (all three installed on a Forge in 0.1.0-alpha.2, DR-106)	Forge

Table 1

The release itself is a small set of signed metadata on GitHub that names the images and where to get them (Section 12).

3.3. Two Ways In

An agent reaches Deep32 in one of two ways.

Direct MCP:
: The agent calls the gateway's tools itself while it works, as it would any MCP server [MCP]. The headend holds a token that the Workstation issued to it at pairing.

The Ask window:
: A question typed on the Workstation travels over the Ask link to the relay on the headend:

1. The Workstation sends the question on a research session of the D32U/1 link (Section 11).
2. The relay starts the agent in a box (Section 11.7) with the research skill.
3. Each tool call the agent makes goes to a per-session proxy on the relay, which sends it back over the same link to the Workstation's gateway.
4. The answer streams back to the window.

An Ask session needs no MCP entry on the headend and never reaches the gateway over the tailnet.

3.4. Trust Boundaries

- The model provider receives the question, the passages the agent reads, and the agent's reasoning about them. Once installed, a deployment talks to no other third party except Tailscale's coordination service, time servers and Let's Encrypt, which Tailscale uses to issue the HTTPS certificate for a published service.
- Being on the tailnet grants nothing. Every service checks a credential that was created on its own machine and handed over only inside an authenticated pairing (Section 10.3).
- Documents are untrusted input. Tool output is document text, never instructions (Section 14.5).
- Mirrors are untrusted transport. An image is trusted only when it matches the signed release metadata (Section 12).

4. Corpus

4.1. Collections

doc_type	Documents	Passages	Contents	Default
rfc	365	12,041	IETF RFCs	Yes
cisco	489	49,923	Configuration guides, command references, TechNotes, validated designs and other Cisco documentation	Yes
powershell	241	5,149	Windows PowerShell 5.1 help, 7.x differences, networking modules	No
system	85	695	Deep32's own code, configuration and design notes	No
Total	1,180	67,808		

Table 2: Collections in the Public Release

Restricted and non-redistributable collections from my own deployment are not in the public release. Every derived table (lookups, outlines, catalog) in the release was rebuilt from the kept collections, and every row in it cites a kept document (DR-17). The Cisco documentation remains Cisco's copyright; RFCs fall under the IETF Trust's provisions.

The opt-in collections have their own FTS5 table, so a default search ranks exactly as it would if they didn't exist (Section 6.2, DR-14). A request reaches them only by naming them with `doc_type` or `docs`.

4.2. Document Manifest

Each document has a manifest entry with its key, type, title, path, source and page URLs, platform, release, SHA-256, size, license, terms URL, a redistribution flag and its near-duplicates (Figure 2). The platform and release drive release-aware search (Section 6.7); the license and redistribution flag decide what may ship.

```
{
  "doc_key": "rfc4271",
  "title": "A Border Gateway Protocol 4 (BGP-4)",
  "doc_type": "rfc",
  "path": "rfc/rfc4271.txt",
  "source_url": "https://www.rfc-editor.org/rfc/rfc4271.txt",
  "sha256": "98b851146abcd56ecbb398dd1a818bcd
             e2dcac6fde3d901c423d988f86eaab76",
  "bytes": 222702,
  "redistribute": true
}
```

Figure 2: A Manifest Entry (Abridged)

The digest is shown wrapped to fit; it is one string.

4.3. Acquisition and Conversion

Documents come from their publishers' own sites. The fetcher sends an honest User-Agent and keeps a daily byte budget with a 24-hour cool-off after a refusal (DR-09). HTML and PDF are converted to plain text in a sandbox with no network access. Every new document then passes quality checks and a scan for lines that read as instructions, invisible characters and control bytes; a person reviews each flagged line before the document is admitted (DR-11).

4.4. Lookup Tables

Some questions are lookups: whether an RFC is current, what an IANA code point means, or the exact syntax of a command. A ranked search can return a close but wrong passage for those, so each build also produces tables that the identifier tools read directly (DR-16):

- 9,842 RFCs with status and relations (obsoletes, obsoleted-by, updates), and 1,217 errata records;
- 2,366 IANA registry entries;
- about 150,000 CLI command entries with mode, platform and release, with IOS abbreviation expansion;
- 2,578 feature-history entries, and a glossary.

The RFC index covers the whole RFC series, not only the 365 RFCs whose text is in the corpus, so `lookup_rfc` can report the status of any RFC an answer cites.

5. Index Generations

5.1. Identifiers

Identifier	Form	Derivation
Document	`doc_` + 64 hex digits	SHA-256 of the source file
Passage	`chk_` + 64 hex digits	SHA-256 over the document hash, the passage's position and its text
Vector point	UUID	UUID version 5 [RFC9562] in the URL namespace, of a fixed prefix and the passage ID

Table 3

Because the identifiers derive from content, a passage whose text and embedding input did not change keeps its ID and reuses its vector in the next generation (DR-42). One update embedded 904 new passages and reused 62,617. The content image reuses all 67,808 vectors of the generation it was cut from.

5.2. Bundle Layout

A generation ships as a knowledge-base bundle (contract C3 in the repository), shown in Figure 3.

```
<generation>/
  BUNDLE.json          compatibility record (Table 3)
  MANIFEST.json        every other file with size and SHA-256
  root/
    sqlite/rag.db      documents, sections, passages; FTS5 tables
    kb/kb.sqlite       RFC index, errata, IANA, CLI, glossary,
                       feature history
    sources/           the corpus the generation was built from
    config/            serving settings for this generation
    BUILD_REPORT.json
  dense/
    vectors.f32        N x D float32, little-endian, point order
    points.bin         N records of 96 bytes: point, passage and
                       document IDs, flags
    EXPORT.json        collection settings, counts, digests
```

Figure 3: Knowledge-Base Bundle

Field	Value
format, format_version	"deep32-kb-bundle", 1; readers refuse any other
generation, fingerprint	The generation name; the corpus fingerprint (16 hex digits)
created	A UTC time given to the builder, never read from the clock, so the same inputs give the same bytes
engine	The range of engine versions that may serve the bundle
embedding	Model ID, revision, dimension (1024), number format (float32-le), distance (cosine), pooling (last-token), and the query and document prompts
reranker	Model ID and revision used by the build's checks
blocks, counts, opt_in	The corpus blocks, per-collection counts, and the opt-in collections
manifest_sha256, signer	Digest of MANIFEST.json, and the key that signed the bundle

Table 4: BUNDLE.json Fields

An importer accepts a bundle only when its signature holds, every file matches MANIFEST.json, both formats are known, the installed engine lies in the `engine` range, and the embedding model ID, revision and dimension equal what the paired Forge serves. On that last mismatch, meaning search stays off and the importer says why. Disc 2's bundle is trusted through the release signature; a bundle a user's own Forge builds is signed with that Forge's key, which the Workstation enrolled at pairing.

5.3. Building

Passages are cut by heading scope: text merges within one top-level section and never across it, and a table that has to be split repeats its caption and header row in every piece (DR-36). Each passage is embedded with its document title and heading path (Section 6.3). The build writes the FTS5 tables, the section index, the vectors and the lookup tables, then a build report.

On a Forge, a new generation starts from a clone of the previous generation's vector snapshot and adds to it. Two approximate-index builds from identical vectors differ by a question or two at rank 1, which would make a zero-slack gate a coin flip; a clone changed no top-50 list in 500 test queries (DR-45).

5.4. Validation Gate

A generation is served only after the labeled question set (Section 13.1) runs against it and against the generation in service, question by question. It passes when all of the following hold (DR-44):

- no net loss at hit@1 or at hit@8 overall;
- no net loss at hit@8 within any topic;
- delivery@8 (Section 13.2) no lower;
- no opt-in document in a default result, no hit on an out-of-scope question, no degraded response, and no query that only succeeded on retry;
- the SQLite FTS5 integrity check passes (DR-47).

Wins and losses are counted per question, with a sign test and a paired bootstrap interval. A change made only for speed must leave every ranked list identical instead (DR-06).

5.5. Switching and Rollback

The engine never edits the index it serves. Switching to a new generation is one setting and a service restart, with a smoke test before and after; if the test after the switch fails, the engine returns to the previous generation by itself (DR-41).

6. Retrieval Pipeline

Every search runs a keyword leg and a dense leg in parallel, fuses their rankings, reranks the best candidates with a cross-encoder, folds duplicates, and delivers the result in three tiers (Figure 4).

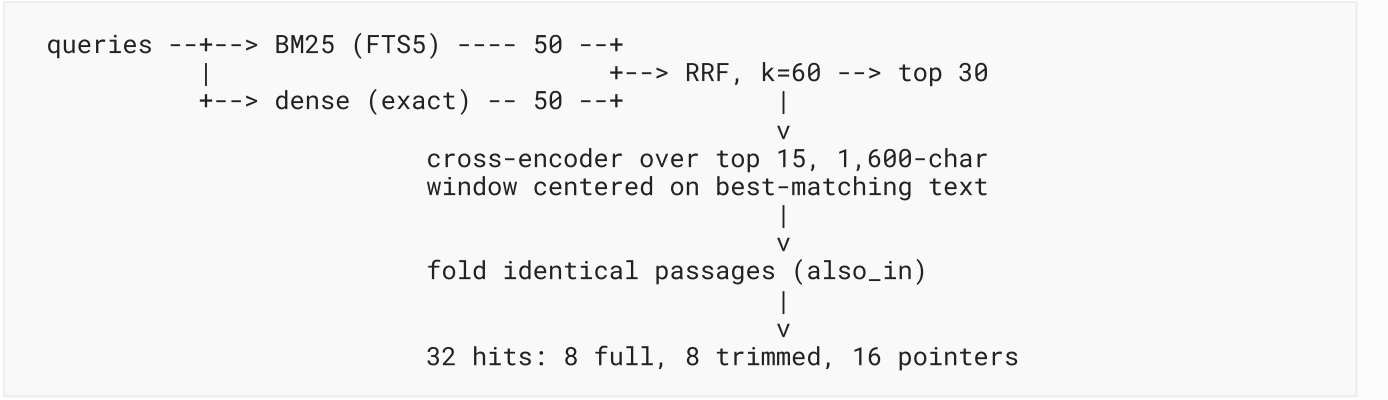


Figure 4: One Search

6.1. Query Planning

A `search_many` call carries one to eight queries; the research skill sends identifiers as keyword-only queries and paraphrases as dense queries. When a query contains identifiers, the engine also forms a keyword variant from them and sends it to the BM25 leg only, where it improves ranking and costs no embedding (DR-23).

6.2. Keyword Leg

The keyword leg uses SQLite FTS5 [FTS5] with the `unicode61` tokenizer and diacritics removed. BM25 [BM25] scores two columns, the passage text at weight 1.0 and the heading path at weight 2.0. Ranking happens inside FTS5 and only the top 50 rows are read. Query terms that occur in at least half of all passages are dropped before the search; a request can turn this off (DR-22).

Opt-in collections live in a second FTS5 table with the same definition. BM25 statistics count every row of the table a query runs on, including rows a filter later excludes, so separate tables are what keep default rankings unaffected by opt-in documents (DR-14).

6.3. Dense Leg

Embeddings come from `jina-embeddings-v5-text-small` [JINA-EMB], served by Text Embeddings Inference 1.9 [TEI] on the Forge: 1024 dimensions, float32, last-token pooling, cosine distance, with the model revision pinned. The input forms are fixed:

```
query:     "Query: {text}"
passage:   "Document: {title}\nSection: {section path}\n\n{text}"
```

The Workstation searches the vectors with an exact scan (Section 8.3). A Linux engine loads the same vectors into Qdrant [QDRANT] with an HNSW index [HNSW]. My production deployment keeps HNSW, where exact search changed 17 of 726 final answers with one loss and no wins (DR-46).

6.4. Fusion

BM25 scores and cosine similarities are on different scales, so the legs are combined by reciprocal rank fusion [RRF], which uses ranks only. With r_i(p) the 1-based rank of passage p in leg i, and w_i the leg's weight:

```
score(p) = sum over legs i that returned p of  w_i / (60 + r_i(p))
```

Both legs have weight 1.0. Each leg contributes 50 candidates and the top 30 fused passages are kept (DR-21). On the test set, hit@1 is 0.710 for BM25 alone, 0.783 for dense alone and 0.822 fused.

6.5. Reranking

A cross-encoder, `jina-reranker-v3.5` [JINA-RR], reads the question together with each of the top 15 fused candidates (10 per query inside `search_many`) and scores how well the passage answers it. It reads a window of 1,600 characters centered on the candidate's best-matching lines, not the passage head (DR-28). The reranked order replaces the fused order for those candidates.

Reranking is the largest single gain (hit@1 from 0.822 to 0.955) and most of a search's time: about 400 ms of a 540 ms median (DR-26).

6.6. Folding

Vendor documentation repeats itself across releases. Passages with identical normalized text and an identical delivered window fold into one hit, which lists the other locations in `also_in`. A passage half-covered by a better-ranked hit from the same file is dropped (DR-29). Folding happens after reranking.

6.7. Release Selection

`release_max` skips documents from a newer release on the same major line. `prefer_release` chooses which copy of a folded passage is shown, without changing rank or text. A hit from a release other than the preferred one is marked (DR-30).

6.8. Delivery

By default a search returns 32 hits in three tiers: ranks 1 to 8 with up to 2,400 characters each, ranks 9 to 16 with 800, and ranks 17 to 32 as 250-character pointers (DR-31). Each window is centered on the hit's best-matching line. A request with `limit` gets exactly that many hits instead. The agent reads whole sections through `get_outline` and `get_section`; the windows exist to locate text, not to be quoted from blindly (DR-35).

6.9. Degraded Operation and Admission

The Forge is often a desktop PC, so the engine treats it as unreliable (DR-32). After three consecutive failed calls to a model service, a circuit breaker skips that service for 30 seconds and then lets one trial call through. BM25 always answers. Every response carries a `degraded` list:

Failure	Search behavior	degraded
Embedding service or vectors	BM25 alone	dense
Reranker	Fused order, not reranked	reranker
Lookup tables	Hits without RFC status	kb
Section index	Hits without section references	structure

Table 5

Admission is bounded (DR-33). The Workstation runs two searches at once and queues sixteen; any further request receives HTTP 503 with Retry-After. Connections to the model services stay open for 300 seconds, because a new TLS handshake across the tailnet cost 61 to 73 ms against 4 to 5 ms on a reused connection (DR-34).

7. Agent Interface

7.1. Transport

The gateway is an MCP server using the Streamable HTTP transport with JSON responses [MCP]. It is stateless and holds no documents: it checks the bearer token in constant time and passes the call to the engine. On the Workstation it listens on loopback and is published to the tailnet by Tailscale Serve at `https://<node>.<tailnet>.ts.net/mcp` (Section 10.1).

7.2. Tools

All eleven tools only read (DR-73).

Tool	Returns
search_rag	Ranked hits for one query, each with citation, section path, RFC status and a text window
search_many	One to eight queries fused into one ranked list
get_outline	A document's section tree, to a depth or under one chapter
get_section	A section's full text with its neighbors, paged up to 100,000 characters per call
grep_corpus	Matching lines with document, line and section; literal by default, bounded regular expressions optional
find_command	CLI index rows: command, mode, platform, release
lookup_id	What an identifier is and where it is defined
lookup_rfc	Status, obsoletes, obsoleted-by, updates and errata of an RFC
verify_quote	exact, normalized or not_found, with the location
list_documents	The catalog: key, type, release, RFC status, size, title
rag_status	Generation, fingerprint, counts, backend and breaker state

Table 6: Gateway Tools

7.3. Quotation Check

`verify_quote` takes a quotation and a document, optionally a section, and returns one of three results (DR-37):

exact: The quotation occurs verbatim.

normalized: It occurs after whitespace runs are collapsed, RFC page headers and footers are removed, and words hyphenated across a line break are joined.

not_found: Neither.

The check is a string match with no model involved, so it gives the same result every time. It shows that the words are in the source; it says nothing about whether the reasoning built on them is right.

7.4. Answer Protocol

Given tools, a model will often still answer from memory. The research skill that ships for Claude Code and Codex therefore has the agent follow a fixed procedure (DR-40):

1. Look up every identifier (RFC number, LSA type, path attribute, CLI command) before using it.
2. Run one `search_many` call with three to six queries, restricted to one collection where the question allows.
3. Read the whole section, and any Restrictions or Defaults section next to it.
4. Run `lookup_rfc` on every RFC cited.
5. Run `verify_quote` on every quotation; drop any that returns `not_found` and reread the section.
6. Compare the RFC's text with the vendor's documented behavior and defaults, and report any difference.

7. If the corpus does not cover the question, say so. Before saying so, open the outline of the likeliest document and read the matching section (DR-80).

The default depth runs at most three bounded search rounds and one batch of quotation checks. Deeper modes are available on request (DR-79).

8. Workstation

8.1. Platform

The Workstation is OpenBSD 8.0 on amd64 with a CDE desktop in the HP-UX style (DR-51). It installs on a physical machine or as a virtual machine: Hyper-V Generation 1, or KVM with BIOS firmware, IDE disks, a VGA display and an e1000 network adapter. Native code and Python packages are built for the plain amd64 baseline, with no processor-specific flags (DR-53).

The engine and gateway are the same Python programs as my production deployment, so their results are identical by construction (DR-52). They run as separate users under pledge(2) [PLEDGE] and unveil(2) [UNVEIL]; the engine listens on a UNIX socket and the gateway on loopback.

8.2. Hardening

- pf denies by default. Outbound traffic is allowed only to Tailscale's infrastructure, to time servers and to Let's Encrypt, where the Workstation's Tailscale client gets the HTTPS certificate for the published gateway; there is no route to package mirrors (DR-56).
- The system runs at securelevel 1 with system files immutable. Updates install only from signed update sets, in the boot window or in single-user mode, with hash checks and rollback.
- Nothing administrative can be changed remotely (DR-54). Only pairing and publication open listeners.
- Features that would contact anything else ship off (DR-96).

Securelevel 2 was tested and rejected: it broke encrypted USB media, disk formatting and connection handling.

8.3. Exact Vector Search

Qdrant does not run on OpenBSD without Docker, so the Workstation scans `vectors.f32` exactly with a small C kernel (DR-55). The kernel uses SSE with four 128-bit accumulators and one fixed order of operations on any amd64 processor, and refuses to be built with FMA or AVX code, so a given query produces the same scores on every machine.

On a 78,275-vector index the exact scan took 12.7 ms against Qdrant's 19.8 ms. Its scores matched production's exact mode with no mismatch over 154,392 compared pairs. Qdrant's own kernel takes an AVX path when the processor has both AVX and FMA, so on such hosts the two can differ in the last bits; release acceptance therefore compares ranked hits and allows score differences below a stated tolerance. The cost of an exact scan grows linearly with the corpus.

8.4. Ask Window and Library

The Ask window is the Workstation's client for the Ask link (Section 11), and `d32-ask` sends the same requests from a terminal. The Deep32 Library is a separate reference collection (an encyclopedia subset with media, papers and an RFC shelf) with its own storage and indexes. The gateway's tools never search it; a document

enters the corpus only when the user sends it there and a new generation is built (DR-19). The encyclopedia's article text is from Wikipedia under CC BY-SA 4.0, and each picture keeps its author's license. It ships as a Kiwix archive whose SHA-256 matches the one Kiwix publishes for it.

8.5. Install Status

The installer and first boot report progress as single-line records on the serial console (contract C6), so a host can follow an unattended install without screen scraping.

At first start the Workstation waits for its Tailscale sign-in. It shows the sign-in address on the text console, in a C6 record on the serial line, and in the First Start window on the desktop. Once it is signed in and has its tailnet address, it looks for disc 2 in its CD drives and imports it without further input; First Start and the console follow the import. The importer checks the signature first, then reads the disc once, in two parts, checking each file against the signed manifest as it reads it (Section 5.2). The first part is the knowledge base: it is copied, checked, installed and switched on by itself, so search, Ask, publication and pairing work from then on. The Library follows in the same run, at a lower priority, and switches on when all of it has checked out.

On 20 November 2026, on a KVM virtual machine reading disc 2 from a virtual CD drive, the knowledge base was in use about 8 minutes after the import started, and a complete import took about 2 hours.

9. Forge

9.1. Services

The Forge runs the embedding and reranking models in Docker. Both listen on loopback behind one door that checks the Forge token, and the door is published to the tailnet with Tailscale Serve (DR-63). The token is made on the Forge and handed to the Workstation at pairing. One Linux Docker form with Compose and restart policies runs on every host (DR-60). On Windows it runs in Docker Desktop or in an Ubuntu 24.04 WSL 2 distro with Docker Engine. Release 0.1.0-alpha.2 needs Docker's classic image store (`overlay2`): its images do not load into the containerd store, which Docker Engine 29 and Docker Desktop use by default.

9.2. GPU Pack

Text Embeddings Inference builds are specific to a GPU generation, and a pack carrying every build was mostly bytes a given user would never run. The GPU pack is therefore split (DR-61):

Shared core: The door and the reranker as container images, the files of both models (embedding and reranking), and the runtime: the loader, the Compose files and the pairing program. Every Forge needs it.

Embedder add-on: One small add-on per card class, holding only that class's Text Embeddings Inference build as a container image. The embedding model itself is in the shared core. A Forge downloads the one for its card.

Ingestion add-on: The PDF converter, the index builder and Qdrant as container images, with the converter's models and source. In 0.1.0-alpha.2 a Forge installs it too, because the Forge loader always loads the converter image (DR-106).

Supported cards are NVIDIA compute capability 7.5 and newer, which is the RTX 20 series onward, in seven classes: 7.5, 8.0, 8.6, 8.9, 9.0, 10.0 and 12.0 (DR-62). `deep32-inventory` names the class before anything is downloaded. Older cards and other vendors' graphics get a keyword-only Workstation.

In 0.1.0-alpha.2 the shared core is 18.6 GB, each embedder add-on 3.1 to 3.3 GB and the ingestion add-on 15.3 GB, so a Forge downloads about 37.1 GB. The images are stored uncompressed in this release.

9.3. PDF Conversion

A PDF conversion job takes the whole card: it pauses the embedding and reranking services and runs one job at a time. The MinerU [MINERU] stage needs AVX2; on older processors the converter uses PyMuPDF only (DR-64).

9.4. Agreement Across Cards

Users run different GPU generations and vectors differ in the last bits. The two measured cards agree within 1e-4, and a release ships vectors computed on one card (DR-65). The Workstation's check of a paired Forge allows a cosine of at least 0.999 for an embedding and 0.005 for each rerank score, so any supported card generation passes while a wrong or damaged model fails (DR-107). Measured on an RTX 3080 and an RTX 2080 Ti over 19 known-answer queries: minimum cosine similarity between the two cards' vectors 0.999999854, 14 of 19 vectors bit-identical, largest component difference 5.9e-5.

10. Network and Pairing

10.1. Tailnet and Publication

Every part signs in to the user's tailnet with the user's own account, during install or, on the Workstation, at first start (Section 8.5); no image carries an auth key (DR-66). Services listen on loopback and are published with Tailscale Serve [TAILSCALE] over HTTPS, on the tailnet only. The installer adds the routes it owns, refuses to touch routes it did not add, refuses Funnel, and never changes tailnet settings itself (DR-67).

Serve's HTTPS certificates put machine names in the public Certificate Transparency logs [RFC9162]. Default machine names carry nothing personal, and publication asks for explicit agreement once.

Parts should reach each other directly. A relayed path measured 70 to 157 ms against 1 to 2 ms direct, and `deep32 doctor` reports a relayed path as a warning (DR-70).

10.2. Tokens and Keys

Every token is 32 random bytes written as 64 hexadecimal digits. Each is made on the machine whose service checks it, at first boot or at pairing, and kept where only that service can read it. No token appears on an image, in a log or in a status line (DR-68).

Credential	Made on	Used for
Engine token	Workstation	Gateway to engine
MCP token, one per paired agent	Workstation	Agent to gateway
Forge token	Forge	Engine to the Forge's door
Forge bundle key	Forge	Signs bundles the Forge builds
Link certificates and pins	Each end of a link	Mutual TLS

Table 7

10.3. Pairing by Short Code

The Workstation pairs with a Forge, and an agent pairs with the Workstation's gateway, by the same procedure (DR-69). The side being joined is the responder; the other side is the joiner.

1. The responder opens a pairing window and shows a one-time code of ten symbols from the RFC 4648 base32 alphabet, `[A-Z2-7]{10}`, normally in two groups of five. The window lasts five minutes and allows one attempt.
2. The person types the code on the joiner.
3. Both sides run SPAKE2 [RFC9382] over the P-256 group [SP800-186] with the code as the password and each side's TLS certificate pin as its identity, inside a TLS 1.3 connection [RFC8446].
4. On success each side stores the other's pin. Inside the authenticated channel the responder hands over what the joiner needs: a Forge gives its token and its bundle public key; the Workstation gives the agent a new MCP token of its own.

Three failed windows within 30 minutes lock pairing for 15 minutes. The window deadline uses a monotonic clock; the lockout uses wall-clock timestamps that survive a restart. Removing a pairing on either side revokes what that side gave.

10.4. Transport Switch

With `transport=loopback`, every service is reached on 127.0.0.1 and sign-in, Serve and discovery are skipped. Tokens and pairing still apply, so every test runs on one machine with no tailnet (DR-71).

11. The Ask Link (D32U/1)

The Ask link carries research sessions between the Workstation and the relay. The Workstation's side of the protocol is the C link client that has shipped on the Workstation since before this release; the portable relay implements the other side. This section states the protocol as that client implements it.

11.1. Transport and Identity

The Workstation (role `node`) dials a TCP connection to the relay (role `relay`). The relay binds loopback and is reached across the tailnet through a TCP forwarder that its installer creates.

Both ends use mutual TLS 1.3 [RFC8446]; TLS 1.2 is allowed only by explicit local configuration, and there is no plaintext form. Each end has a self-signed X.509 v3 certificate [RFC5280] with an ECDSA P-256 key, valid for ten years. The trust anchor is the pin: the SHA-256 of the peer's whole DER certificate. Chain, host name and validity-period checks are not used. A peer whose certificate does not match the stored pin **MUST** be refused, except during an armed pairing window (Section 11.5).

11.2. Framing

Every frame has a 12-byte header (Figure 5). All integers are unsigned and big-endian.

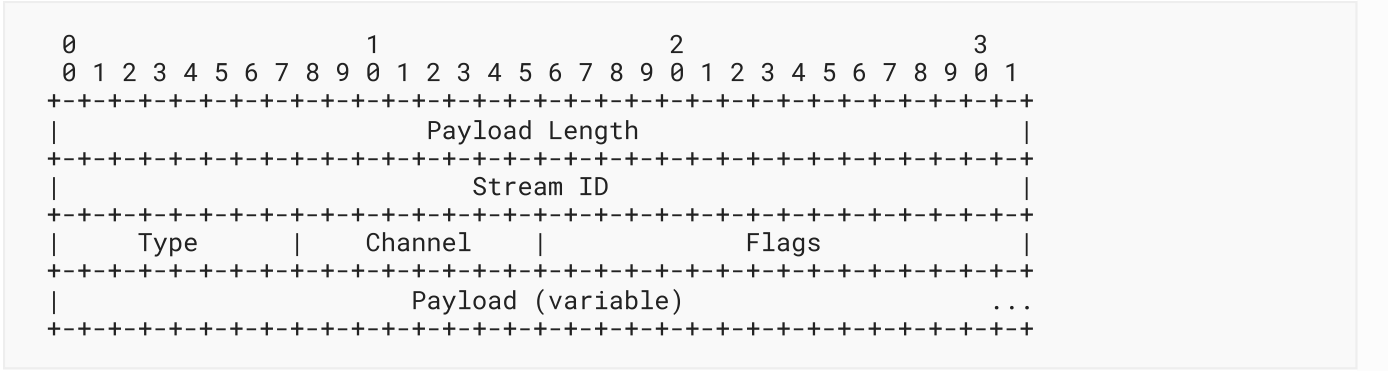


Figure 5: D32U/1 Frame

- Payload Length excludes the header and **MUST NOT** exceed 1,048,576.
- Control frames (HELLO, HELLO_ACK, PING, PONG, ERROR, PAIR_CONFIRM, PAIR_COMMIT) carry Stream ID 0 and Channel 0; every other frame carries non-zero values in both.
- Only DATA frames may set flags, and the only flag is 0x0001, END_MESSAGE. Any other flag is a protocol error.

Type	Name	Payload
1	HELLO	JSON, at most 16,384 bytes
2	HELLO_ACK	JSON, at most 16,384 bytes
3	OPEN	JSON, at most 16,384 bytes
4	OPEN_ACK	Empty
5	DATA	Non-empty
6	WINDOW_UPDATE	4-byte credit
7	CLOSE	Empty
8	RESET	4-byte error code
9	PING	8 opaque bytes
10	PONG	The 8 bytes of the PING it answers
11	ERROR	4-byte error code
12	PAIR_CONFIRM	Empty
13	PAIR_COMMIT	Empty

Table 8: Frame Types

Type 0 and types above 13 are protocol errors.

11.3. Handshake

After TLS completes, the node sends HELLO with a fresh 32-byte nonce. The relay answers with its own HELLO and then, after the node's HELLO_ACK, its HELLO_ACK. The node's HELLO is:

```
{"proto":"D32U/1","id":"workstation","version":1,"versions":[1],
 "role":"node","features":["research-v1"],
 "nonce":"<64 lower-case hex digits>","sessions":[]}
```

The relay's HELLO has the same members with role `relay`, and may add `profiles`: a list of at most one entry, named `research`, that lists the models and the strengths (`efforts`) the relay offers. Each entry in those lists is a string of at most 64 bytes, and the node keeps at most 16 of each. HELLO_ACK is `{"version":1,"features":["research-v1"],"paired":true}`, or `false` for `paired`. When both acknowledgements say paired, the link is active. Messages are strict UTF-8 JSON [RFC8259]: no byte order mark, no duplicate keys, no unknown keys, nesting at most 32 deep. Reconnecting does not resume sessions; the `sessions` array is always empty.

11.4. Channels and Flow Control

Channel	Opened by	Carries
1	Node	Research sessions: questions, streamed answers
2	Relay	Reverse MCP: the agent's tool calls to the Workstation's gateway
4	Relay	Relay status

Table 9

An OPEN on any other channel receives RESET. The node allocates odd stream IDs from 1 and the relay even ones, each increasing and never reused on a connection.

A research session starts with a `session.open` message, the first DATA on a new channel 1 stream. It names the model and the strength chosen for the conversation, from the lists in the relay's HELLO; the relay turns away a model or a strength it did not offer.

Each stream starts with 256 KiB of credit, replenished by WINDOW_UPDATE. Each side sends PING every 15 seconds while active and drops the link after 45 seconds without a valid complete frame; a sender waits at most 180 seconds for credit. A side sends at most 256 frames per second, counting control frames. The node reconnects with backoff of 1, 2, 4, 8, 16 and 30 seconds, each multiplied by a random factor between 0.8 and 1.2, and only to a pinned peer.

11.5. Pairing by Short Authentication String

The Ask link does not use the short code of Section 10.3. It keeps the comparison that its Workstation client already implements; replacing it would have meant a Workstation protocol change for no security gain (DR-69).

1. A person arms a pairing window on each side's console. A window lasts 300 seconds and allows one connection from an unpinned peer. Only while it is armed **MAY** a side complete TLS with an unknown

certificate.

2. Each side computes the string from the two pins and the two HELLO nonces. Let P_lo and P_hi be the two 32-byte pins in unsigned lexicographic order (equal pins fail), N_node and N_relay the nonces, and

```
D = SHA-256(P_lo || P_hi || N_node || N_relay)
```

3. The first 110 bits of D, most significant bit first, are split into ten 11-bit indices into the 2,048-word English list of [BIP39]. The ten words are shown on both consoles. They are a display only, not a seed, and carry no checksum.
4. The person types, on each side, the words the other side shows. A side whose typed words match its own string sends an empty PAIR_CONFIRM; a mismatch ends the attempt.
5. After both confirmations, each side writes the peer's pin and ID to protected storage and only then sends PAIR_COMMIT. The link becomes active when both sides have committed.

A stored pin is replaced only by removing the pairing first. Removing it deletes the pin, ends the link and ends every session on it.

11.6. Reverse MCP

The agent in an Ask session reaches the gateway through the link. The relay runs a loopback proxy for each session with its own random bearer, and forwards a call only while a verified turn is running. Each call travels on channel 2 to the Workstation, which passes it to its gateway on loopback. The headend never dials the Workstation's published gateway for an Ask session.

11.7. The Agent Box

An agent started by the relay can call the eleven tools of Table 6 and nothing else (DR-76): no shell, no file reads or writes outside an empty per-session folder, no web access, no other MCP server, and none of the user's own configuration, skills or MCP entries.

- Claude Code runs with its built-in tools disabled, only the eleven tools allowed, one MCP server under strict MCP configuration, no settings layers, its own configuration folder, a cleared environment and no session persistence. The relay offers the models and strengths that Claude Code lists, and each conversation can choose among them. The relay checks the session's initialization event for exactly the chosen model, the folder, the tool list and the server, and ends the session otherwise.
- Codex runs with a read-only sandbox, approvals set to never, web search, history and other features (shell, patching, plugins, hooks, apps, multi-agent, memories, image viewing) off, and strict configuration. Before the first turn the relay checks the effective configuration and that exactly one MCP server is enabled.

During a turn the relay reads the agent's output as a stream of typed events and keeps only the answer text, the final result and the turn's token counts. A call to any tool outside the eleven, or an event of a kind the relay does not know, ends the turn with an error. Claude Code's report of its usage window between messages (`rate_limit_event`) is a known kind: it carries no answer text and does not end the turn. A turn whose final result is an error still fails. When Claude Code's profile has no usable sign-in at the start of a session, or the model provider refuses the sign-in during a turn, the relay stops with the reason `agent-sign-in-required` and does not pass the agent's error line on as an answer.

At install and after each sign-in, the relay runs an install check that uses no model. It starts the agent as far as its initialization report, records the agent's version, the model and strength it starts with and the models and strengths it lists, and stops it before any question. The record is bound to the SHA-256 of the agent program, so after an agent upgrade the relay refuses Ask sessions until the check has run again.

A live self-test goes further: scripted attempts to escape the box (run a command, write outside the session folder, fetch a web page, reach another MCP server, read the user's agent configuration) must all fail, and one allowed tool call must succeed. Each attempt uses the user's model allowance, so it runs only when the user asks for it.

12. Distribution and Verification

12.1. Hosting

The images are whole files hosted on the Internet Archive [IA], which also serves an official torrent of each. The GitHub release holds the signed metadata: a root checksum list and its detached signature, a mirror list, an asset map, one child checksum list per image, each image's README and its software bill of materials (DR-90). A mirror is untrusted transport; trust comes only from the root list's signature.

12.2. Signature Check

Release metadata is signed with one long-lived OpenPGP key [RFC9580] (Ed25519) as a detached, armored, binary-mode signature (DR-86). The key's fingerprint is pinned in the repository and printed in the README. Each verifier runs GnuPG with a new, empty home directory and only the release key shipped with it, so the user's own keyring is never used. There are two verifiers:

On the host that downloads the images:
: `deep32-get` and `verify-release` in `src/release-tools`, in their shell and PowerShell forms, share one Python helper that runs `gpg`. The helper refuses a key file that holds a secret key or does not hold exactly one primary key with the pinned fingerprint, imports that key alone, and runs `gpg --verify` on the root checksum list; `verify-release` also verifies each image's child list the same way. A signature is good only when all of the following hold:

  - `gpg` exits 0;
  - its status output has exactly one VALIDSIG;
  - the signature class in VALIDSIG is 00 (a binary signature over the exact bytes; a text-mode signature would also match other line endings);
  - the primary-key fingerprint in VALIDSIG equals the pinned fingerprint.

On the Workstation:
: The disc 2 importer runs `gpgv` with a keyring holding only the release key installed with the system. A signature is good only when `gpgv` exits 0, its status output has exactly one GOODSIG and one VALIDSIG and none of BADSIG, ERRSIG, EXPSIG, EXPKEYSIG, REVKEYSIG or NO_PUBKEY, the signature class in VALIDSIG is 00, and its primary-key fingerprint equals the pinned fingerprint.

Nothing in the metadata is parsed before the signature is good. The verifier then parses the same bytes it verified, never a re-serialized copy.

12.3. Download

`deep32-get` downloads an image as follows:

1. Authenticate the root list (Section 12.2), then the mirror list and the image's child list against it.
2. Check that the destination file system has room for the remaining signed bytes plus a 16 MiB reserve.
3. Try the mirrors in their signed order, at most sixteen. Each mirror gets one request and at most five redirects, a 30-second connect timeout and a 60-second stalled-read timeout. No proxy, cookie or credential is sent.
4. Check each 64 MiB chunk against its signed digest as it arrives. After an interruption, keep only the complete chunks that verified and resume from there. A corrupt mirror cannot poison a later one.
5. Check the SHA-256 of the whole image before publishing the file under its final name.

If every mirror fails, the getter prints the torrent URL from the signed metadata as a manual alternative; it contains no torrent client.

12.4. Bills of Materials and Reproducibility

Each image has an SPDX 2.3 [SPDX] document beside it, covered by the root list. Keeping the bills of materials outside the images left disc 2 byte for byte the image that had already passed the release gates (DR-95).

The release notes state only the reproducibility that was demonstrated: which artifacts were built twice and compared byte for byte. Appendix C gives the state for this release.

12.5. Doctor and Self-Test

`deep32 doctor` checks every link of an installation without changing anything, prints one row per check with OK, WARN or FAIL, and prints a suggested fix next to each failure (DR-94).

`deep32 selftest` runs a public question set through the installation and compares retrieval with the release's reference values. The set has 63 RFC-only questions, nine in each of seven topics (BGP, BGP extensions, OSPF, IP, MPLS, segment routing, and other protocols), balanced across natural-language, keyword and identifier styles. Its 107 labels each name a document and an exact span of that document's shipped text. A question counts as found at the rank of the first of its first ten hits whose document matches a label and whose text contains that label's span. The tool reports hit@1, hit@3, hit@8 and MRR@10. The reference values come from three identical runs on the release host.

13. Evaluation

13.1. Test Set

The test set has 726 questions: 696 whose answer is in the corpus and 30 whose answer is not (DR-98). They cover enterprise routing, switching, VPN, SD-WAN, network services and access control, with at least three answerable questions per topic.

Questions and first labels were written per topic by an LLM pipeline with an author and an adversarial verifier. After each corpus change, newly retrieved passages are judged in pools and audited blind; the last two audits agreed on 22 of 27 and 15 of 16 items. Labels are copied to identical passages in sibling documents (DR-99).

There are 9,209 labeled spans across 463 documents. Evaluation data is never ingested into the corpus (DR-103).

13.2. Metrics

hit@k: The share of answerable questions with a relevant passage in the top k. A passage is relevant when it comes from a labeled document and contains a labeled span.

MRR@10: The mean over answerable questions of 1/r, where r is the rank of the first relevant passage within the top 10, or 0 if there is none.

delivery@8: The share of labeled spans of the top 8 hits that arrive intact in the delivered text windows. It measures whether the agent can see the evidence, not only whether it was ranked (DR-102).

Every comparison is paired by question: wins, ties and losses, a sign test, a paired bootstrap interval and a per-topic breakdown (DR-101). Three runs over one week returned identical metrics.

13.3. Results

All 696 answerable questions, through the gateway, on 2 November 2026:

Stage	hit@1	hit@8	Median ms
BM25 only	0.710	0.928	125
Dense only	0.783	0.970	122
Fused	0.822	0.983	140
Fused and reranked	0.955	0.991	542

Table 10: Retrieval by Pipeline Stage

Metric	Fused and reranked	Fused only
hit@1	0.955 (665)	0.822 (572)
hit@3	0.983	0.934
hit@8	0.991 (690)	0.983 (684)
MRR@10	0.970	0.882
delivery@8	0.909	0.904

Table 11: Full Metrics

Paired over the 696 questions, reranking wins 102 and loses 9 at hit@1, and wins 6 and loses none at hit@8.

Each of the 31 questions missed at rank 1 was judged afterwards. In 7, the rank-1 passage did answer the question but had no label. In 21, the reranker placed another passage above a relevant one. In 3, fusion left the relevant passage outside the reranker's top 15. With the 7 label gaps corrected, hit@1 would be 0.966 (672 of 696). I report 0.955 until the label set is frozen again (DR-100).

13.4. Limitations

- The questions were written by a language model and checked by another pass of one. They may be easier or more uniform than questions people ask.
- Labels come from pooled judgment, so a relevant passage that no system retrieved stays unlabeled.
- The figures measure retrieval only. A claim-level evaluation of answers is designed, with held-out questions, negative probes and blind grading against the saved evidence, and pass marks fixed in advance: at most 2% fabricated claims, every negative probe passed, every citation exact. It has not been run, and the beta release waits for it (DR-104).

14. Security Considerations

14.1. Hosted Model

The model provider sees each question, the passages the agent reads and the agent's reasoning. A user who cannot send those to a provider cannot use Deep32 as released. A slot for a locally served, OpenAI-compatible model exists in the adapter interface with no adapter behind it yet (DR-77).

14.2. Credentials

Tokens are generated on the machine that checks them and never leave it except inside an authenticated pairing. Agent credentials are held by the operating system's protected store (DPAPI for the current user on Windows; a mode 0600 file in a mode 0700 directory on Linux) and loaded into the agent's process only. They never appear in an MCP configuration file, an argument list or output (DR-78). The gateway compares bearer tokens in constant time.

14.3. Pairing

A pairing code has 50 bits of entropy and is never sent. SPAKE2 gives an attacker who observes or intercepts an exchange no way to test guesses offline, so each guess has to be made online and costs a whole pairing window. Only a person at the responder's console can open a window, each allows one attempt, and three failed windows in 30 minutes lock pairing for 15 minutes. The person sees every failed window.

The Ask link's string has 110 bits and depends on both certificates and both nonces. An attacker in the middle would have to make both sides display the same ten words for two different certificate pairs. Typing the other side's words, rather than glancing at both, keeps a person from approving a mismatch by habit. Unknown certificates are accepted only while a window armed on that side's own console is open.

14.4. Network Exposure

Services listen on loopback. The only routes into a part are the Serve routes its installer owns and, on the headend, the relay's forwarder. Funnel is refused, so nothing is published to the public Internet. The Workstation's pf permits outbound traffic only to Tailscale's infrastructure, to time servers and to Let's Encrypt, which Tailscale uses to issue the HTTPS certificate for a published service. Machine names appear in Certificate Transparency logs once HTTPS certificates are issued.

14.5. Prompt Injection

Documents can carry text written to steer a model. Ingest flags instruction-like lines for review, and the gateway's server instructions tell the agent that tool output is document text, never instructions (DR-11, DR-74). Both are mitigations; a model can still follow injected text. The agent box limits what an Ask session can do if it does: it can only search and read the corpus.

14.6. Supply Chain

Every image is checked against signed metadata before use, and a mirror cannot substitute content (Section 12). Model files and container images in the GPU pack are pinned by digest, and the SPDX documents list what each image contains. The release is built from pinned sources through an allowlist, then rebuilt and scanned as its own product, because built files can carry strings from material that was cut (DR-82, DR-83). Whoever holds the release signing key can sign a malicious release; the key is held by the author alone.

14.7. Host Considerations

On the Forge, membership in the `docker` group is equivalent to root, as on any Docker host. On the Workstation, system files are immutable at securelevel 1, so a compromised service cannot replace them without a reboot into the boot window. Admission control (Section 6.9) keeps a burst of searches from exhausting the engine's worker threads.

In this release Ask has no password of its own (DR-111). Root and every account in the Workstation's `_d32users` group can open the Ask broker's socket, from the Ask window or a terminal, and ask questions on the plan of the agent signed in on the relay. The pairing (Section 11.5) and the tailnet protect the link to the relay; who can ask on the Workstation itself is decided by its accounts, its login and its screen lock.

15. IANA Considerations

This document has no IANA actions.

16. References

16.1. Normative References

[FIPS180-4] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9382] Ladd, W., "SPAKE2, a Password-Authenticated Key Exchange", RFC 9382, DOI 10.17487/RFC9382, September 2023, <https://www.rfc-editor.org/info/rfc9382>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/info/rfc9562>.

[RFC9580] Wouters, P., Ed., Huigens, D., Winter, J., and Y. Niibe, "OpenPGP", RFC 9580, DOI 10.17487/RFC9580, July 2024, <https://www.rfc-editor.org/info/rfc9580>.

[SP800-186] National Institute of Standards and Technology, "Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters", NIST SP 800-186, DOI 10.6028/NIST.SP.800-186, February 2023, <https://doi.org/10.6028/NIST.SP.800-186>.

16.2. Informative References

[AGPL-3.0] Free Software Foundation, "GNU Affero General Public License, Version 3", November 2007, <https://www.gnu.org/licenses/agpl-3.0.html>.

[APACHE-2.0] The Apache Software Foundation, "Apache License, Version 2.0", January 2004, <https://www.apache.org/licenses/LICENSE-2.0>.

[BIP39] Palatinus, M., Rusnak, P., Voisine, A., and S. Bowe, "Mnemonic code for generating deterministic keys (BIP 39)", 2013, <https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki>.

[BM25] Robertson, S. and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond", Foundations and Trends in Information Retrieval, Vol. 3, No. 4, pp. 333-389, DOI 10.1561/1500000019, 2009, <https://doi.org/10.1561/1500000019>.

[D32-DR] Cummings, J. A., "Deep32 Decision Record (D32-DR)", November 2026, <https://github.com/hellobpdu/deep32/blob/main/docs/DECISIONS.md>.

[DEEP32] Cummings, J. A., "Deep32: Retrieval-augmented question answering for network and security engineering", Version 0.1.0-alpha.2, DOI 10.5281/zenodo.24371692, 2026, <https://doi.org/10.5281/zenodo.24371692>.

[FTS5] SQLite, "SQLite FTS5 Extension", <https://www.sqlite.org/fts5.html>.

[HNSW] Malkov, Y. A. and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 42, No. 4, pp. 824-836, DOI 10.1109/TPAMI.2018.2889473, 2020, <https://doi.org/10.1109/TPAMI.2018.2889473>.

[IA] Internet Archive, "Internet Archive", <https://archive.org>.

[JINA-EMB] Jina AI, "jina-embeddings-v5-text-small", Model card and weights; licensed CC BY-NC 4.0.

[JINA-RR] Jina AI, "jina-reranker-v3.5", Model card and weights.

[MCP] Model Context Protocol, "Model Context Protocol Specification", <https://modelcontextprotocol.io/specification>.

[MINERU] OpenDataLab, "MinerU", Version 4.0.7 in the PDF converter; MinerU Open Source License, the Apache License 2.0 with additional terms, <https://github.com/opendatalab/MinerU>.

[PLEDGE] The OpenBSD Project, "pledge(2): restrict system operations", <https://man.openbsd.org/pledge.2>.

[QDRANT] Qdrant, "Qdrant Documentation", <https://qdrant.tech/documentation/>.

[RAG] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W., Rocktaschel, T., Riedel, S., and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020, <https://arxiv.org/abs/2005.11401>.

[RFC1771] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, DOI 10.17487/RFC1771, March 1995, <https://www.rfc-editor.org/info/rfc1771>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/info/rfc9162>.

[RRF] Cormack, G. V., Clarke, C. L. A., and S. Buettcher, "Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods", Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 758-759, DOI 10.1145/1571941.1572114, 2009, <https://doi.org/10.1145/1571941.1572114>.

[SPDX] The Linux Foundation, "The Software Package Data Exchange (SPDX) Specification, Version 2.3", 2022, <https://spdx.github.io/spdx-spec/v2.3/>.

[TAILSCALE] Tailscale Inc., "Tailscale Documentation", <https://tailscale.com/kb>.

[TEI] Hugging Face, "Text Embeddings Inference", <https://github.com/huggingface/text-embeddings-inference>.

[UNVEIL] The OpenBSD Project, "unveil(2): unveil parts of a restricted filesystem view", <https://man.openbsd.org/unveil.2>.

Appendix A. Sections and Decisions

Where to find the reasons behind each part of this report in the Decision Record [D32-DR].

Section	Decisions
1, Introduction	DR-01 to DR-08, DR-110
4, Corpus	DR-09 to DR-19
5, Index Generations	DR-36, DR-41 to DR-49
6, Retrieval Pipeline	DR-20 to DR-35
7, Agent Interface	DR-37 to DR-40, DR-73, DR-74, DR-79, DR-80
8, Workstation	DR-50 to DR-57
9, Forge	DR-58 to DR-65, DR-106 to DR-108
10, Network and Pairing	DR-66 to DR-72
11, The Ask Link	DR-69, DR-75, DR-76, DR-111
12, Distribution	DR-81 to DR-97
13, Evaluation	DR-98 to DR-105

Table 12

Appendix B. Changes from 0.1.0-alpha.1

- Whole images on the Internet Archive with signed metadata and mirror failover replace images split into parts on GitHub (DR-90).
- The GPU pack is a shared core with one embedder add-on per card class and an ingestion add-on (DR-61); in 0.1.0-alpha.2 a Forge installs all three (DR-106).
- New: `deep32 doctor`, `deep32 selftest`, SPDX documents beside each image (DR-94, DR-95).
- New: a portable Ask relay for Windows, Linux and macOS (DR-75). With Claude Code it offers the models and strengths Claude Code lists, and each conversation can choose among them.
- The Workstation image includes the Tailscale package, which alpha.1 lacked; the alpha.1 images were withdrawn for that and other install defects (DR-89).
- Deep32's code moved from AGPL-3.0-or-later to the Apache License 2.0 with a NOTICE file. The PDF converter and `e26-render`, which use MuPDF, stay under AGPL-3.0-or-later (DR-110).
- New: the Forge installs from the signed disc 3 images. Before it writes anything, the installer checks the host against a signed host recipe, the driver and graphics memory floors and a signed storage budget. The recipes for Ubuntu 24.04 with Docker Engine and for Windows 11 build 26100 or later with Docker Desktop 4.99.0 or later on WSL 2 are accepted by my waiver as release owner, ahead of their full end-to-end runs (Appendix C).

Appendix C. Implementation Status in 0.1.0-alpha.2

The table compares what this report specifies with what release 0.1.0-alpha.2 does, in the places where the two differ or where the release has not been tested in full.

Section	Specified	In this release
10.1, 12.5	`deep32 doctor` checks every link of an installation and reports a relayed path as a warning	On the images, but not working yet: it does not read an installation's record, so it always reports the installation check as failed
12.5	`deep32 selftest` compares retrieval with the release's reference values	The reference values are not shipped
10.3	Removing a pairing on either side revokes what that side gave	Revoking a pairing on the Forge leaves the Forge unable to restart; revoke by reinstalling the Forge into a new data folder
11	The Ask link carries research sessions between the Workstation and the relay	Works end to end with Claude Code on the relay: a question asked on a Workstation, in the Ask window or with `d32-ask` in a terminal, is answered through the relay. The Ask window has no password of its own in this release, so every account in the Workstation's `_d32users` group can ask (DR-111, Section 14.7)
3.1, 8.1, 9.1	The three parts share a machine or sit on three; the Workstation runs on a physical machine, in Hyper-V or in KVM; the Forge runs on Linux or on Windows	A Workstation in a KVM virtual machine with a Forge on Linux is the tested layout; the all-in-Docker layout, which runs the engine and gateway in containers on the GPU host with no Workstation, is not in this release's images; a Hyper-V Workstation and a Docker Desktop Forge have not been run end to end
12.4	The release notes state which artifacts were built twice and compared byte for byte	The disc 2 build writes its image twice and stops unless the two copies are identical. The GPU pack's producer stages and writes the shared core image twice from the same inputs and stops unless the two images are byte-identical. Disc 1, the embedder add-ons and the ingestion add-on were each built once, and no byte comparison of them is published
9.1	One Docker form runs on every Forge host: Linux, an Ubuntu 24.04 WSL 2 distro and Docker Desktop	Forge host acceptance has not been run in full; the install is opened by my waiver as release owner, and full runs on Linux, WSL 2 and Docker Desktop follow in a signed revision
9.2	The capability manifest in the signed metadata gives a storage budget for each GPU pack component	The budgets are declared upper bounds until they are measured

Section	Specified	In this release
9.3	The Forge converts a PDF with MinerU, or with PyMuPDF alone on a processor without AVX2	On 19 November 2026 this release's ingestion image converted a one-page test PDF with `--tier standard` on an RTX 2080 Ti in about 2 minutes 22 seconds, with Docker Engine 29.1.3 in an Ubuntu 24.04 WSL 2 distro. A conversion with the default `--tier advanced`, or in Docker Desktop, has not been run

Table 13

Acknowledgments

Deep32 is built on the work of others: OpenBSD, SQLite, Qdrant, Text Embeddings Inference, the Jina AI embedding and reranking models, MinerU, MuPDF, PyMuPDF, Tailscale, and the RFC Editor's archive of the RFC series. Their licenses are listed in the repository's notices.

Author's Address

Jeffrey A. Cummings
Deep32
Email: jacummings@acm.org
URI: https://github.com/hellobpdu/deep32