# Darash — The Scripture Research Engine

PUBLIFYE AS

Updated 2026-07-25

# Contents

# Chapter 1

# What Darash Is

This is a getting-started guide: the one place that takes you — or your AI — from nothing to using Darash well. It is a starting point, not an exhaustive reference, and it grows as Darash does — new features are added here as they ship.

Most Bible software hands you what *other people* have said about the text — study notes, a commentary, a scholar's summary. Darash hands you the text itself, and the data beneath it, so you can see for yourself.

The name is a Hebrew word, dāraš דָּרַשׁ[H1875] — to *seek*, to *inquire*, to *study*, to *search out*. It is what the Bereans did, and why Luke calls them noble:

> **Acts 17:11, KJV** · These were more noble than those in Thessalonica, in that they received the word with all readiness of mind, and searched the scriptures daily, whether those things were so.

They did not simply accept what they were told — they *searched* to see whether it was so. Darash is built for that posture: not "take my word for it," but "see whether these things are so."

## 1.1 A research engine, not a reading app

Darash is an engine. Under one roof it holds:

- the Bible in **59 translations**, across more than thirty languages, side by side
- **Strong's** numbers for every Hebrew and Greek word, with more than **14,000 lexicon entries** standing behind them
- **446,544 cross-references** linking verse to verse
- **13 dictionaries and lexicons** — some 217,000 entries in all
- word-by-word **morphology** across **31,167 verses** — the grammar the original languages carry but English smooths away
- **gematria**, word frequency, hapax legomena, etymology trees, and the ancient **pictographs** behind the Hebrew letters

- a **keyless ELS** discovery engine over the 304,805 letters of the Koren Torah (in the heavyweight Engine edition — see chapter 3)

## 1.2 Everything is verifiable

The point of Darash is not to impress you with answers — it is to let you *check*. Every verse it quotes, every Strong's number, every cross-reference traces back to a source you can inspect. When Darash tells you what a Hebrew word means, it shows you the word, its Strong's index, and where else it stands in the text. Nothing rests on its own say-so.

That is the whole idea: a tool for people who would rather **search the Scriptures** than be handed a conclusion.

# Chapter 2

# What You Can Do With It

Each part of Darash answers a different question you might bring to the text.

## 2.1 Read it in many translations at once

No single translation is "the" Bible — each one is a set of choices. Darash carries **59 translations**, so you can lay them side by side and see where they agree and where a translator had to decide. When two versions differ, that difference is usually the interesting part: it marks a word the original leaves open. King James from 1611 is only one of those. As well as the Strongs index.

They span more than thirty languages — English, Norwegian and Nynorsk, German, French, Spanish, Italian, Dutch, Danish, Swedish, Finnish, Icelandic, Russian, Polish, Czech, Hungarian, Greek, Latin, Hebrew, Persian, Arabic-script and Asian editions among them — and range from the Vulgate of 405 and the Leningrad Codex to a modern Norwegian Bokmål edition. References parse in the book's own language too, so *1 Mos 1:1* and *Sal 23* find their way home as readily as *Gen 1:1*.

## 2.2 Reach the original word

Every Hebrew and Greek word in the text carries a **Strong's** number, and Darash gives you the full lexical entry behind it — the lemma, how it is pronounced, its range of meaning, and the grammar it is carrying. You do not need to read Hebrew or Greek to use it: Darash transliterates the word, glosses it, and links it to every other place it appears.

## 2.3 Follow a thread through the whole Bible

Darash holds **446,544 cross-references**. Pick a verse and it will show you the others Scripture itself ties to it — the quotations, the echoes, the fulfilments. A theme you notice in Genesis can be followed, link by link, to its answer in the Gospels. Is it perfect, no. It will work the best when you use an AI which has a solid context window and is able to think and reason-alike through the red threads. Such as the SOTA LLM models of today.

## 2.4 Go deeper with the lexicons

Thirteen **dictionaries and lexicons** stand behind the text — for a word, a name, a place, a topic. Where a single gloss is not enough, these give you the scholarship of the old.

## 2.5 See the grammar English hides

English flattens distinctions the original languages mark plainly — tense, voice, mood, who is acting on whom. Darash's **morphology** restores them word by word, so you can see, for instance, that a verb is passive where a translation made it active, or that a "you" is plural where English cannot say so.

## 2.6 Weigh and count

Darash also measures the text: **word-frequency** (how often, and where, a word is used), related words, and **gematria** (the numeric value Hebrew and Greek letters carry). These are tools for noticing emphasis and pattern.

## 2.7 Trace a word back to its root

A word rarely stands alone. Darash walks an **etymology tree** — following the derivation and "see also" links several levels deep — so one call shows you the family a word belongs to rather than the single entry you asked for. Alongside it sit synonyms, related Strong's numbers, and the **pictographs**: each of the 22 Hebrew consonants began as a picture, and Darash will spell a word out in those older images.

## 2.8 Find the rare and the paired

Some things only show up when you count. **Hapax legomena** — the words that occur exactly once in the whole Bible — are listed for Hebrew, Greek or both, and a word used once is nearly always used deliberately. And where you want two ideas held together rather than one, Darash will find the verses where two Strong's numbers **occur together**, which is how a phrase like *the sting of death* is studied properly rather than guessed at.

## 2.9 Search by meaning, not only wording

Plain keyword search finds the words you typed. **Semantic search** goes further in three layers: it finds the verses containing your words, then builds a meaning cluster by mapping them back to Strong's numbers and expanding one hop through synonyms and roots, and then takes the Hebrew words from that cluster and looks for them as ELS codes

in the Torah letters. You get the verses, the expanded word web with its sources traced, and the encoded hits together.

## 2.10 Take the text with you

Darash is one self-contained binary with the data embedded, so it runs offline on macOS, Linux or Windows — as an MCP server, an HTTP API, or a plain command line. A whole translation can also be **exported** as structured JSON through a single-use download link, book by book or entire.

## 2.11 Let the text speak first: keyless ELS

Finally, Darash carries a **keyless ELS** (equidistant letter sequence) discovery engine for the Hebrew Torah, yet this is usually a feature only if you can run the engine itself. Most ”Bible code” tools make you supply a word and then go looking for it — which all but guarantees a find. Darash works the other way round: it scans a passage and reports the sequences that are *actually there*, before you have named anything. It is a research instrument, not a parlour trick — and Darash is careful to tell you what a result does and does not mean. Although this feature can be challenging to even the most solid LLMs so the person using this needs to push it in order to have it pull out the answers so to speak, unless it also will glee over the gold which is there hidden behind the text without it understanding it. Often placement of Bible Codes (ELS) is just as important if not more and not just the statistics behind it - which often can be of no use. Even so, this is built into Darash and isnt a small feature as such but uses quite a lot of processing resources and memory.

Four later chapters open that engine up: what the ELS suite actually contains, how to *see* a finding rather than only score it, the unattended blind probe that hunts while you sleep, and the WRR tools that test the method itself.

This last one is reserved for the Darash Engine edition. It is both heavyweight — a thorough scan loads gigabytes of letter-index data and wants real CPU — and separately licensed, so the everyday hosted key does not include it. The next chapter explains the editions.

# Chapter 3

# The Three Editions

Darash comes in three editions. They share the same engine; they differ in how you run it, what it costs, and whether the heavy ELS discovery is included. (For current prices, see the store at darash.publifye.pro — they are set there, not in this book.)

## 3.1 Darash Access — the hosted key

**Darash Access** is the everyday edition: a personal API key to the Darash engine, already hosted and running — nothing to install. You point your own AI assistant at the MCP endpoint darash-api.publifye.pro/mcp (or use the **darash** command line — for instance, *darash verse kjv "John 1:1"* fetches a verse). It gives you the full research surface: the 59 translations, Strong's Hebrew and Greek, the 446,544 cross-references, the 13 dictionaries, morphology, search, word studies, and gematria. (It also includes authoring access to Junifye, Darash's publishing companion.) For almost everyone, this is Darash.

The one thing Access does **not** include is the keyless ELS discovery — for that, see the Engine below.

## 3.2 Darash Engine — run the whole thing yourself

The keyless ELS discovery belongs to the **Darash Engine** — for two reasons. It is heavyweight: a full ELS scan loads many gigabytes of letter-index data and wants real CPU and memory. And it is licensed separately: the everyday Access key does not carry ELS at all. The Engine is a license to run the complete engine on your own infrastructure (self-hosted or fully managed), built to scale to tens of thousands of concurrent users, with the Junifye publishing server bundled in — and it is the one edition where ELS is switched on. If your work is the Torah-code research Darash is known for, this is the edition you need. It is priced per deployment, by quote.

## 3.3 Darash Agent — a ready-made assistant

The **Darash Agent** is the no-setup option: a ready-made AI assistant that lives as a private bot **on Telegram**, locked to you. You simply message it — ask about a verse, a word, a cross-reference — and it does the research, and can even draft documents and books

for you through Junifye. There is nothing to connect or configure; after you subscribe, a short guided chat sets up your bot. It is Darash for someone who wants the answers without touching the tools. Its the pricier of the two, Darash Access vs Agent, as its more extensive to set up and uses multiple times more resources - and we have limited seats available as of now. Price and availability may change in the future though.

## 3.4 The website

You will also see **darash.publifye.pro** — the home of Darash on the web. It has example lookups, hosts the ELS poster gallery for churches and study groups, and is where you subscribe to any of the three editions (there is also a free tier with a daily allowance). Think of it as the front door, not the workshop: the real research happens through your AI, the CLI, or the Agent.

The next chapter walks through a few examples in plain language. Four chapters then open up the ELS engine — what it holds, how to see a finding rather than only score it, the unattended blind probe, and the WRR tools that test the method itself. The final two chapters are the technical reference your assistant uses to connect to Darash and drive it.

# Chapter 4

# Worked Examples

A few short walk-throughs, in plain language, to show the kind of thing Darash makes easy.

## 4.1 A word study: more than ”peace”

Open almost any English Bible and šālôm שָׁלוֹם[H7965] is simply ”peace.” Ask Darash for the word itself and a fuller picture appears: šālôm means **completeness, soundness, welfare, wholeness** — it is built from a root meaning *to be whole, to make complete* (H7999). So when Scripture pronounces *shalom* over someone, it is not only the absence of conflict; it is everything being made whole. Darash shows you the word, its range, and the more than two hundred places it stands — and a verse you thought you knew opens up.

## 4.2 Following a cross-reference

Read a verse and ask Darash what Scripture ties to it. From a single line it will surface the other passages — the quotation it draws on, the prophecy it answers, the later echo. You follow the thread the text itself laid down, rather than one a teacher chose for you. A study that used to need a shelf of reference books becomes a short conversation.

## 4.3 Seeing the grammar

Ask Darash for the morphology of a verse and it returns each word’s grammar — part of speech, tense, voice, mood, person. Again and again this resolves a question English alone cannot settle: a verb is *passive* (something is being done **to** the subject) where the translation made it active, or a command is plural where English ”you” is silent. The meaning was always there in the original; Darash simply makes it visible.

## 4.4 A keyless ELS scan (Engine edition)

On the Darash Engine, point Darash at a Torah passage and ask it to scan — without naming a target. It reports the equidistant letter sequences that are present, with their skip distance and position. This is the honest form of the method: because you did not tell it what to look for, a result is not something you went fishing for. Darash is deliberately

sober about what this means — it reports what is measurable and leaves the weighing to you. Yet, you yourself also need be pushing the LLM in order for it to recognize the visual ELS and not just the statistical ones, as placement is as important if not more important compared to statistics, which an LLM will have a tendency to compare against - and so this is a weakness in any LLM. This is important to be aware of.

# Chapter 5

# The ELS Suite in Full

The ELS side of the Engine is not one tool but a workshop. This chapter is the map of it.

Everything here runs over the Koren Torah — **304,805 letters**, the exact text used in the WRR 1994 *Statistical Science* paper, carried with a verified SHA-256 so you can prove the letters have not moved under you before you compute anything on them.

## 5.1 Finding codes

- **els_discover** — the keyless entry point: scan and report what is present *before* you have named anything
- **els_search** — one Hebrew word at every skip in the Torah, with position, skip and verse for each occurrence
- **els_scan** — a grid read in all eight directions, ranked longest word first, since longer words are rarer by chance. Ranking is a heuristic for what to inspect, not a significance score.
- **els_sentences** — consecutive meaningful words running the same direction, grouped into multi-word sequences
- **els_verse_codes**, **els_verse_theme**, **els_verse_signal** — what a particular verse carries, and whether a named set of words clusters around it beyond chance
- **els_study** — the heavy one: a whole *set* of words across all 152,402 skips with cylindrical wrapping, hunting the skips where several of them converge

## 5.2 Where a code lands

This is the part most tools skip, and usually the part that matters most.

- **trace_els_code** — walks every letter of a code through the Torah and returns the surface words and verses each letter lands on, with a per-verse roll-up. The headline answer to ”which verses does this code actually hit?”

- **els_proximity** — WRR-style closeness between two codes, including the bounding-rectangle area when the text is wrapped at that skip
- **els_skip_position_cluster** and **els_skip_coherence_map** — whether a skip is doing something across the whole Torah, or only at one lucky position

Placement is not decoration. A word encoded across twelve consecutive tribal offerings, or bookending the verses that frame a redemption arc, is saying something no z-score will tell you. Read the verses a code hits before you weigh the number.

## 5.3 Grading a find

- **els_pvalue** — a permutation test on placement: generate random Hebrew words of the same length and measure how often they land in that book too
- **els_pvalue_surface** — the sharper question: are the surface words the code passes through semantically related to the concept, more than random words at the same skip?
- **els_thematic_score** — per verse, does the real Torah encode vocabulary thematically tied to that verse's plain meaning at a rate above vocabulary- and length-matched shuffled Torahs?
- **corpus_status** and **corpus_top_verses** — the Torah-wide distribution of that score, so one verse's number can be read as a percentile rather than in a vacuum

Skip 1 is plain text, not a code, and the tools say so rather than letting it quietly inflate a result.

## 5.4 Scanning wide

- **els_corpus_scan** — one skip, drilled into the broader inflected corpus of roughly 18,000 word forms, against a hundred shuffled controls
- **els_themed_scan** — Torah-wide vocabulary containment, with a heatmap and drill-down by scan and cell id
- **els_distill_theme** — pull the content words out of verses or raw Hebrew (nouns, verbs, adjectives kept; particles, prepositions and articles dropped) to build the vocabulary a scan will use

- **els_palindromic_mirrors** — positions where a term reads as a code both forward and backward from the same anchor letter, the Torah treated as a closed loop so mirrors near either end still count
- **els_geometric_mirror_scan** — places where the *geometry* of a code's placement matches the *meaning* of the word encoded

## 5.5 Honest by construction

Each of these reports its method alongside its result: what was shuffled, what was held fixed, how many controls were run, and what the number does and does not license you to say. Statistical rarity is where an investigation starts, not where it ends. The engine measures and shows; the weighing stays with you.

# Chapter 6

# Seeing a Finding

Statistics score rarity. Placement carries the meaning. Darash draws its findings so you can see where each letter lands in the verse — not just how unlikely the code is.

## 6.1 The grid

Wrap the Torah's letters at a chosen width and the text becomes a sheet you can look at. **els_grid** builds one; **els_grid_get**, **els_grid_info**, **els_grid_lookup**, **els_grid_words** and **els_grid_unique** read it back — what is at this cell, which words appear in this grid, which of them appear in no other. **els_grid_image** renders the same grid as a picture instead of a table.

## 6.2 The overlay

**els_verse_overlay** is the one to reach for when a finding needs to be *seen*. It renders a verse — or a range of them — as an SVG figure in which each encoded code is colour-marked at its real letter positions in the surface text, with leader lines running out to a legend that names the word encoded.

This is the figure that makes a finding legible to someone who does not read Hebrew, and it is where placement stops being an abstraction: you can see that a word threads through *these* words of *this* verse, and judge whether that is saying something.

## 6.3 The view store

Rendered figures are kept in a public view store and served at their own URL, so a figure can be linked, shown in a study, or put on a wall. **view_list** and **view_info** audit what is live; **view_set_ttl** and **view_delete** manage the rest.

By default a figure expires after 72 hours — right for exploration, wrong for anything you intend to keep. **view_freeze** pins one permanently: it is persisted and rehydrated on restart, so the URL stays live indefinitely. **view_unfreeze** puts it back on the clock. The rule of thumb is simple — the moment a figure is publication-worthy, freeze it, because the default will otherwise quietly take it away.

These are the figures behind the ELS poster gallery, and they are what you reach for when a study needs a picture rather than another paragraph.

# The clearest example — a Gospel Code in Leviticus 11:32

At skip 6, four Hebrew letters read vertically through the verse. Each lands on a surface word that matches its meaning:

מ on בְּמַיִם (in/through water) א on יוּבָא (shall be brought) ד on עַד (until) ט on וְטָהֵר (and shall be cleansed)

The arc: water → brought → until → cleansed. The surface text and the hidden letters trace the same immersion pattern.

# Other codes at the same verse

Exodus 15:25 at skip 11: The code ישורל reads vertically through the verse where Moses casts the tree into bitter water. The first three letters spell ישו (yesh) — the beginning of Yeshua — landing on וַיִּמְתְּקוּ (made sweet), שָׁם (there), and וְשָׁם (and there). The code then crosses into verse 26 where the Lord says: I am YHWH your healer.

Leviticus 11:32 at skip 6 (crossing into verse 33): Another code bridges the cleansing of washable vessels in verse 32 (evening, cleansed) to the breaking of earthen vessels in verse 33 (vessel, which). The clean vessel is washed; the clay vessel is broken.

# Chapter 7

# The Blind Probe

The blind probe is the keyless method run at scale, unattended, for as long as you let it.

## 7.1 What it does

Instead of asking ”is this word in there?”, the probe works through a Hebrew-lemma source skip by skip and records what forms as an equidistant sequence — accumulating results you never asked for. You start it, and it keeps going: **els_blind_probe_start** and **els_blind_probe_stop**, with **pause**, **resume** and **reset** for the long middle, **set_concurrency** to decide how hard it leans on the machine, and **status** for where it has reached.

## 7.2 Reading what it found

- **els_blind_probe_top** — the top lemmas at one specific skip
- **els_blind_probe_search**, **els_blind_probe_lemma**, **els_blind_probe_by_verse** — query the accumulated output by term, by lemma, or by where it lands in the text
- **els_blind_probe_list_completed** — which skips have actually been finished, so you know what your query covered
- **els_blind_probe_investigate** — one composite call that returns everything needed to validate a single finding: give it a lemma, a skip and a source, and it assembles the evidence rather than making you gather it across a dozen calls

## 7.3 Why it is built this way

The value of the blind probe is exactly that nothing was named in advance. When you supply a target and go looking, a find proves very little — you would probably have found *something*. When the machine reports what is there before anyone chose a word, the searcher has been taken out of the search. That is the honest form of the method, and it is why the results are worth arguing about.

The other half is the cost. This is the heaviest thing Darash does: gigabytes of letter-index data resident in memory, real CPU for hours rather than seconds, and a completion

list you should check before trusting a query. That expense is precisely why the probe lives in the Engine edition and not on a shared hosted key.

## 7.4 Reading the results well

Two habits are worth forming. First, always look at **where** the hits land — across the whole Torah, not one chapter — before deciding whether a result is interesting; a marker stamped on each of twelve consecutive consecrations is a pattern, not a coincidence of formula. Second, treat the statistics as the opening of the question rather than its closing. Rarity describes; placement interprets.

# Chapter 8

# WRR — Testing the Method

WRR is Witztum, Rips and Rosenberg — whose 1994 paper in *Statistical Science* is the reason equidistant letter sequences are taken seriously enough to be argued about at all. Darash carries the same Torah text they used and a small set of tools that speak their metric directly.

## 8.1 Prove the text first

**wrr_text_check** hashes the embedded Koren Torah so you can confirm, before computing anything, that the letters are the ones the paper worked from. It is an unglamorous tool and the most important one here: every ELS result in Darash is a claim about a specific 304,805-letter string, and a result computed on a subtly different text is not a result at all.

## 8.2 Measure the way they measured

- **wrr_distance** — computes σ, the cylindrical compactness between two ELS placements, as defined in the paper's own appendix
- **wrr_load_sample** — the sample data, so the method can be exercised on a known case
- **wrr_study_coherency** — a cross-layer coherency score for a verse-anchored study, asking whether the layers of a finding agree with one another rather than each standing alone

## 8.3 Why this matters

The point is falsifiability. Same text, same metric, published method — so whatever you conclude, someone who disagrees can run it and say why. Darash exists for people who would rather check than be told, and that has to include checking Darash. The WRR tools are where the engine hands you the instruments to do exactly that.

# Chapter 9

# Connecting Darash to Your AI

This chapter and the next are written for the AI assistant — though a technical reader will follow them easily.

## 9.1 What MCP is

**MCP** (Model Context Protocol) is an open standard for giving an AI access to external tools. A service exposes a set of *tools*; an MCP-capable assistant can list them and call them during a conversation. Darash is such a service.

## 9.2 The endpoint

Darash's engine speaks MCP at darash-api.publifye.pro/mcp — a JSON-RPC 2.0 endpoint. There are two ways to authenticate:

- **Sign in with your browser (preferred).** On the first call your MCP client opens a Publifye login; you approve access once, and the session refreshes itself afterwards — there is no key to copy or store.
- **API key.** Send your key as the 'X-API-Key' header on the same endpoint.

You get access by subscribing to **Darash Access** (or the Agent or Engine) at darash.publifye.pro; a free tier with a daily allowance lets you try it first. All access is metered with daily quotas — generous for everyday study and shown in your account — so heavy automated use should plan around limits.

## 9.3 Adding it to your assistant

Most MCP clients take a small piece of configuration naming the server and its URL — typically a single command that adds the endpoint above as an HTTP MCP server. Once Darash is registered, the assistant calls 'tools/list' to discover every tool, and 'tools/call' to run one. From then on it can research Scripture mid-conversation — no further setup. (Connected to a hosted key you get the full research surface; the ELS tools appear only when the assistant is pointed at a Darash Engine.)

## 9.4 The shape of a call

Every tool follows the same JSON-RPC form:

```
{"jsonrpc":"2.0","id":1,"method":"tools/call","params":{"name":"get_verse","arguments":{
"bible":"kjv","ref":"John 1:1"}}}
```

and returns a JSON result carrying the source data. The next chapter lists the tools worth knowing, grouped by what they do.

# Chapter 10

# The Tool Reference

The engine's tools, grouped by purpose. Names and arguments are stable; an assistant can always call 'tools/list' for the authoritative, current set — and 'get_health' for the live counts of translations, cross-references and lexicon entries, which grow as data is added.

## 10.1 Reading the text

- **get_verse** — one verse or a small range in a chosen translation. '{"bible":"kjv","ref":"John 3:16"}'
- **get_chapter / get_book** — a whole chapter, or a whole book.
- **list_bibles / get_bible** — list, or fetch a passage from, the 59 translations.
- **list_books** — the 66 books, lean by default; pass detail=full for every alias.
- **compare_verses** — the same reference across several translations together.
- **export_bible / export_bible_book** — a whole translation, or one book, as structured JSON via a single-use download link.
- **verse_to_position / position_to_verse** — convert between a reference and its absolute letter position, the bridge between reading and the ELS layer.

## 10.2 The original languages

- **get_strongs** — the full lexical entry for a Strong's number, gematria value included. '{"number":"H7965"}'
- **search_strongs / search_strongs_definition / reverse_strongs** — find a Strong's number from a word, from wording inside a definition, or from an English gloss.
- **strongs_in_verse** — every Strong's-tagged word in a verse.
- **get_morphology** — word-by-word grammar: part of speech, tense, voice, mood, person.

- **get_related_strongs / get_synonyms / synonym_stats** — neighbouring words in the same semantic field, and how that field is distributed.
- **etymology_tree** — the derivation and "see also" links walked several levels deep in one call.
- **hebrew_pictographs** — what each of the 22 consonants originally depicted, spelled out across a word.

## 10.3 Lexicons and word study

- **list_dicts / list_dict_topics** — the 13 dictionaries, and what each covers.
- **lookup_dictionary / multi_dict_lookup** — look a term up in one, or in many at once.
- **word_study / verse_study** — an assembled study around a word or a verse, several lookups folded into one call.
- **word_frequency** — where and how often a word occurs.
- **hapax_list** — the words that appear exactly once in the whole Bible.
- **co_occurrence** — the verses where two Strong's numbers stand together.

## 10.4 Cross-references and search

- **get_cross_refs** — the references Scripture ties to a verse, from the 446,544-link set.
- **search** — keyword search across the text.
- **semantic_search** — the three-layer engine: verses, then an expanded meaning cluster, then the Hebrew of that cluster looked for as Torah codes.
- **search_gematria** — words and verses sharing a numeric value.

## 10.5 Service

- **get_health** — what this instance is, what data it holds, and the live counts.
- **cache_status / cache_control** — inspect and steer the caches the heavy scans rely on.
- **submit_feedback** — send a correction or a note back to the maintainers.

## 10.6 ELS — keyless discovery (Darash Engine only)

These belong to the **Darash Engine** — a separately-licensed capability the everyday Access key does not carry, so they appear in ‘tools/list‘ only when the assistant is connected to an Engine. They are also the heavy end of Darash: a serious scan wants Engine-class memory and CPU. Four earlier chapters cover them properly; this is the index.

- **Finding** — els_discover, els_search, els_scan, els_sentences, els_sentences_get, els_study, els_status, els_verse_codes, els_verse_theme, els_verse_signal
- **Placement** — trace_els_code, els_proximity, els_skip_position_cluster, els_skip_coherence_map
- **Grading** — els_pvalue, els_pvalue_surface, els_thematic_score, corpus_status, corpus_top_verses
- **Wide scans** — els_corpus_scan, els_themed_scan, els_distill_theme, els_palindromic_mirrors, els_geometric_mirror_scan
- **Grids and figures** — els_grid, els_grid_get, els_grid_info, els_grid_lookup, els_grid_words, els_grid_unique, els_grid_image, els_verse_overlay
- **The view store** — view_list, view_info, view_freeze, view_unfreeze, view_set_ttl, view_delete
- **The blind probe** — els_blind_probe_start, stop, pause, resume, reset, status, set_concurrency, search, lemma, by_verse, top, list_completed, investigate
- **WRR** — wrr_text_check, wrr_distance, wrr_load_sample, wrr_study_coherency

So the surface an assistant sees has two sizes: the research tools above on a hosted key, and the full set — well over ninety tools — once an Engine adds the ELS suite. Either way, each tool returns structured JSON with its source data, so whatever the assistant reports, you can trace it back to the verse, the lemma, or the grid it came from. That is the whole point of Darash: not answers to be trusted, but sources to be checked.

## How this was made

This is the author's own work. It was composed with **junifye** and an AI assistant, drawing its Scripture and original-language work (Greek, Hebrew, cross-references) from **Darash** (Hebrew *darash*, "to seek, inquire, study"), a tool and reference work for the Bible in its original languages.

A gift of support — Stripe

*Scan to read this book online*