

Evidence export is one of those features that sounds secondary until you actually have to defend a match, retrace a decision, or hand work from one team to another. Then it becomes the feature. In the context of the open source “Wikidata + Google Knowledge Graph MCP” server and CLI, evidence export is the practical bridge between a promising entity match and a match that another person, or another system, can inspect with confidence.

That distinction matters more than people admit. Plenty of tools can return a candidate entity. Far fewer can show why that candidate surfaced, what facts supported the decision, what remained uncertain, and where a human reviewer should hesitate. The project’s design points directly at that problem. It is built to let agents search Wikidata, read selected facts, and link local records to Wikidata QIDs with inspectable evidence and explicit uncertainty when evidence is insufficient. Once you understand that design goal, the CLI’s batch and evidence export capabilities stop looking like convenience features and start looking like the center of the workflow.

The project sits in a very specific lane. It is read only. It does not edit Wikidata, Google, or user data. It is not official Wikimedia or Google software, and it is not an export of the Google Knowledge Graph. That restraint is healthy. It keeps the tool focused on retrieval, comparison, and decision support instead of drifting into uncontrolled enrichment or silent data mutation. When I evaluate tooling for identity resolution, that boundary is usually a good sign. Systems that separate evidence gathering from record editing are much easier to trust.

Why evidence export belongs in the CLI

A server can answer a single question well enough in an interactive session. A CLI has to survive repeated use, late night reruns, partial failures, and scrutiny from people who were not present when the command ran. That is why evidence export belongs at the command line level.

With an MCP client such as Claude Code, Cursor, or Codex, an agent can query the server and reason over what comes back. That is useful during exploration. But operational work tends to happen in batches. A catalog team may have hundreds of local names to reconcile. A newsroom might need to verify a set of public figures before publication. A data engineering team may want a reproducible artifact they can archive with the run. Evidence export gives those teams something durable.

Durable does not just mean saved to disk. It means interpretable after the fact. If a local record was matched to a Wikidata QID through deterministic resolution logic, the exported evidence should let a reviewer understand whether the outcome was something like `AUTO_MATCH`, `HOLD`, `AMBIGUOUS`, or `NO_CANDIDATE`, and why. Those labels are not cosmetic. They describe the state of certainty in a way that matters for downstream decisions.

This is one of the places where the project’s bounded search approach helps. By default, it returns 3 candidates, and at most 5, instead of dumping a large raw result set on the user. That boundedness makes evidence export more legible. You are not exporting a haystack and calling it transparency. You are exporting the specific candidate set the resolver considered, along with the facts that were judged relevant. That is the right trade off for a lot of production work. Very large candidate dumps can feel comprehensive, but they often make review slower and less reliable.

What “evidence” means here

Evidence in this project is not a vague aura of confidence. It comes from concrete retrieval and comparison steps. The server supports search, entity inspection, related lookups, resolution, and status checks through documented MCP tools such as `kg_search`, `kg_entity`, `kg_related`, `kg_resolve`, and `kg_status`. The fact retrieval side is especially important because selected facts can include ranks, qualifiers, and references on request.

That last detail is where evidence export becomes genuinely useful instead of merely decorative. A flat statement that an entity has some property is often not enough for matching or review. On Wikidata, rank can signal preferred versus deprecated status. Qualifiers can narrow the scope of a claim. References can help a reviewer understand whether a statement is merely present or also supported. You do not need every possible property every time, but when a match is close, those extras can decide whether the record is ready for auto acceptance or should be held back.

In practical terms, good evidence export in a CLI for an MCP for Wikidata should preserve the shape of that reasoning. Not just “candidate X exists,” but “candidate X matched because these specific facts aligned, these others were absent or uncertain, and the resolver therefore produced this explicit outcome.” The project’s own emphasis on inspectable evidence suggests exactly that style of use.

The role of deterministic outcomes

The deterministic resolution outcomes are one of the strongest aspects of the design. A lot of entity resolution tools blur uncertainty behind a score that looks objective but leaves too much open to interpretation. This project instead documents explicit outcomes such as `AUTO_MATCH`, `HOLD`, `AMBIGUOUS`, and `NO_CANDIDATE`.

That is valuable for evidence export because it gives the exported result a stable vocabulary. If an operations team sees `AUTO_MATCH`, they know the resolver found enough support under its rules to make a direct link. If they see `HOLD`, the implication is different. The system found something worth attention, but not enough to proceed without review. `AMBIGUOUS` tells a reviewer that more than one candidate plausibly fits. `NO_CANDIDATE` tells them that the search did not surface a suitable match under the bounded search and deterministic criteria in use.

Those labels also help when results are revisited months later. In my experience, the hardest part of auditing old entity links is not discovering what was chosen. It is discovering what was considered and why uncertainty was or was not tolerated at the time. Deterministic outcomes improve that situation because they force the tool to say what state the match ended in.

Evidence export is where bounded search proves its value

The default behavior of returning 3 candidates, with up to 5, deserves more credit than it usually gets. People often assume that more candidates mean better matching. That can be true during research, but it can be counterproductive in routine CLI runs, especially when the purpose is to export evidence that another team must review.

A bounded candidate set keeps the evidence package proportionate. Reviewers can compare a short list of plausible entities without drowning in noise. It also nudges users to formulate better search inputs and better local comparison logic instead of outsourcing the problem to sheer volume. When a tool sprays twenty or fifty weak candidates into an output file, the burden simply shifts downstream.

There is, of course, a trade off. Bounded search can miss a valid candidate that would appear further down a larger ranking. That is real. But the project's design seems to favor precision and inspectability over exhaustive retrieval, and for many operational matching tasks that is a reasonable choice. Especially because the system can produce explicit uncertainty rather than forcing a weak match. A clean NO_CANDIDATE or HOLD is often safer than a speculative auto link.

Where Google Knowledge Graph fits, and where it does not

The project can optionally use the Google Knowledge Graph Search API as a cross check. That needs to be framed carefully. The cross check is based on exact identifier joins, specifically /m/ corresponding to Wikidata property P646 and /g/ corresponding to P2671. The project treats agreement between Google and Wikidata as provider concordance, not proof of identity.

That is exactly the right stance. In matching work, a second provider can strengthen confidence that two systems are talking about the same thing, but it is not independent truth in any absolute sense. The moment people mistake cross provider agreement for proof, they start automating decisions they should still review.

This matters for searches people may phrase as MCP for google knowledge graph and wikidata, MCP for wikidata, or MCP for google knowledge graph. Those phrases suggest a blended capability, but the actual design is more disciplined than that. Wikidata is the core source. Google support is optional. No account or API key is required for Wikidata. The Google side is supplementary, and the project is explicit that it is not a dump or mirror of Google's graph. For evidence export, that means any Google concordance should be preserved as one signal among others, not elevated above the rest.

How evidence export changes batch resolution work

The CLI's batch support is where the feature becomes operational. A single search result is easy to eyeball in a client. Hundreds of records are not. Evidence export lets a team run a batch, capture the candidate set and selected facts considered for each record, and then review the outcomes systematically.

The best use case is not blind automation. It is staged decision making. Records that land in AUTO_MATCH can be queued differently from those in HOLD or AMBIGUOUS. Records with NO_CANDIDATE can be pushed into a research lane instead of polluting the main matching queue. If the exported evidence is structured consistently, a reviewer can scan for patterns. Perhaps one source system lacks enough distinguishing metadata. Perhaps a naming convention consistently triggers ambiguity. Perhaps certain classes of entities need richer selected facts before they can be matched safely.

That sort of learning only happens when the evidence leaves the ephemeral interaction window and becomes an artifact. I have seen teams improve match quality dramatically once they stop looking only at final links and start looking at the evidence package behind the difficult cases. The batch export gives them material to inspect, sort, sample, and discuss.

A practical pattern for using the CLI

The exact command syntax is not documented in the verified context, so it would be irresponsible to invent it. But the documented capability is enough to describe a sound working pattern. In practice, the CLI is most useful when it is treated as a pipeline stage rather than a one off utility.

A sensible flow looks like this:

1. Run a batch resolution against local records
2. Export inspectable evidence for each attempted match
3. Separate outputs by deterministic outcome such as AUTO_MATCH or AMBIGUOUS
4. Review selected facts, including ranks, qualifiers, and references when needed
5. Archive the evidence artifact alongside the run metadata

That flow works because it respects the boundaries of the tool. The CLI gathers and exports evidence. Humans or downstream systems decide what to do with it. The archive step is not glamorous, but it matters. If you later need to explain why a local record was linked to a given QID, having the evidence artifact from the actual run is far better than reconstructing the case from memory.

The subtle importance of selected facts

One common mistake in entity resolution is to retrieve either too little or too much. Too little, and every person with a common name starts to look identical. Too much, and the exported evidence becomes so heavy that reviewers stop reading it carefully. The project's support for selected fact retrieval is the practical middle path.

Because selected facts can include ranks, qualifiers, and references on request, users can shape the evidence to the case at hand. If the local record has a strong date anchor or an occupation clue, you can imagine focusing retrieval on the parts of the Wikidata record that matter for disambiguation. If the local record is sparse, pulling references or qualifiers may still not resolve the ambiguity, but it at least gives the reviewer more than a bare label and identifier.

This is where experienced judgment matters. Not every record deserves maximal detail. [Wikidata MCP](#) Exporting references for every straightforward match may create unnecessary overhead. But for the close calls, that extra structure can make the difference between a justified hold and an avoidable manual chase.

Evidence export is also about saying "not enough"

The project's language about explicit uncertainty is more important than it sounds. Many data workflows are biased toward completion. A script is expected to return something. A dashboard wants a percentage. A manager wants a linked record count. Under that pressure, tools often smuggle uncertainty into overconfident matches.

Evidence export can resist that pressure, but only if it records insufficiency clearly. A HOLD outcome should not read like a failed AUTO_MATCH. It should read like the correct answer for the evidence available. AMBIGUOUS should preserve the set of plausible candidates, not bury them beneath a single guessed favorite. NO_CANDIDATE should tell the operator that the bounded search and selected evidence did not support any match, which is useful information in its own right.

I like this design because it matches the reality of reference data work. Some records are simply too thin, too noisy, or too overloaded with name collisions to resolve automatically. A trustworthy CLI does not hide that. It exports it.

Operational checks and status visibility

The presence of a kg_status tool may look minor next to search and resolution, but it plays a quiet role in trustworthy export workflows. Before you run batches and save evidence artifacts, you want some indication

that the service is available and behaving as expected. Operational clarity matters because a broken or partially working dependency can produce thin evidence that looks legitimate until someone notices gaps.

This is especially relevant when using the optional Google cross check. Since the Google component is optional, a team should be able to distinguish between a run that included provider concordance and one that relied only on Wikidata. Evidence artifacts are much easier to interpret when the operational state behind them is clear. Again, the verified context does not specify the exact output format, so the responsible point is simply this: status visibility belongs near evidence export, because exported artifacts are only as trustworthy as the conditions under which they were produced.

What this means for MCP workflows more broadly

Wikidata now has its own documented MCP context as well, described as standardized tools for LLMs to explore and query Wikidata programmatically via the Wikidata API and Wikidata Query Service. That broader context matters because it shows where this project fits. It is not trying to be every possible interface to Wikidata. It occupies a more focused niche around search, selected fact retrieval, and deterministic linking with inspectable evidence.

That specialization is useful. General purpose query tools are excellent for exploration and custom analysis. A linking oriented MCP with bounded search and explicit outcomes is better suited to repeatable reconciliation work. When you add CLI batch operations and evidence export, you get something that can slot into real data operations without pretending that all ambiguity can be optimized away.

For teams comparing approaches, this is worth keeping in mind. If your main task is open ended graph exploration, a general Wikidata MCP may be the better first stop. If your main task is linking local records to QIDs and preserving the justification for each decision, the evidence export capabilities in a dedicated CLI become much more compelling.

What good exported evidence should help you answer

When I review match artifacts, I want the export to answer a short set of hard questions quickly. Not with marketing language, but with the actual record of what happened.

A strong evidence export should make these questions easy to answer:

1. What candidate entities were considered under the bounded search
2. Which selected facts were retrieved for those candidates
3. What deterministic outcome the resolver produced
4. Whether optional Google concordance was present through exact id joins
5. Where uncertainty remained, if the match was held or marked ambiguous

Notice what is absent from that list. There is nothing about hidden scores, mystery weighting, or unverifiable confidence language. The emphasis is on **Knowledge Graph MCP identifier** inspectability. If another analyst cannot reconstruct the logic from the artifact, the export is not doing its job.

The value of restraint

One of the strongest qualities of this project is its restraint. No account or API key is needed for Wikidata access. Google support is optional. Search is bounded. Fact retrieval is selective. Resolution outcomes are

explicit. The tool is read only. Each of those choices reduces a different class of operational risk.

That restraint serves evidence export directly. Smaller candidate sets are easier to review. Selective facts are easier to interpret. Read only systems are easier to trust in regulated or high scrutiny environments. Optional concordance is easier to contextualize than mandatory blending. Deterministic outcomes are easier to route and audit than opaque scores.

There is a temptation in graph tooling to equate breadth with quality. In practice, quality often comes from the opposite direction, from tools that know what they are for and stop there. For entity linking work around Wikidata, especially in environments where every match may need to be explained later, evidence export in the CLI is one of those disciplined features that pays off long after the first successful run.

Where it earns its keep

If you only ever search a single entity interactively, evidence export may feel like extra plumbing. The minute your work scales, or your matches need review, or your organization asks why a local record was linked to a specific QID, the feature earns its keep.

It earns it by preserving candidate context instead of only final decisions. It earns it by carrying selected facts, including rank, qualifiers, and references when needed. It earns it by making uncertainty explicit through outcomes like HOLD, AMBIGUOUS, and NO_CANDIDATE. It earns it by treating Google agreement as concordance rather than proof. And it earns it by fitting naturally into batch workflows where reproducibility matters as much as retrieval.

That combination is what makes the CLI side of this MCP project interesting. Not because it promises magic, but because it supports the less glamorous discipline of evidence first entity resolution. In this corner of data work, that is usually the difference between a demo and a dependable tool.