

When Completed Is Not Complete

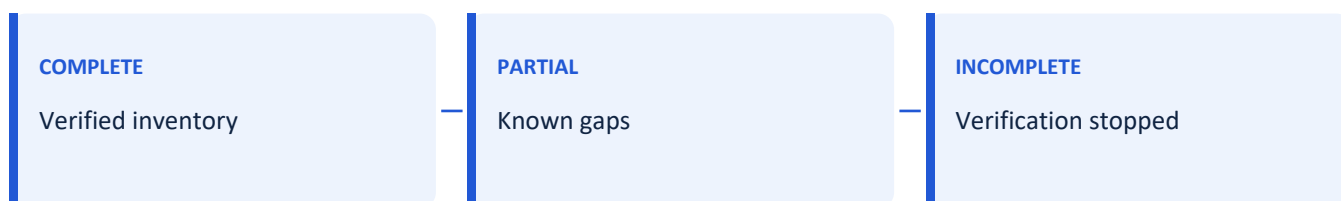
The last mile of an asynchronous API integration

EXECUTIVE CASE STUDY

AMY V

14 OCTOBER 2026

A completed batch is a useful signal, but a consumer still has to establish what came back, what is missing, and whether the result is ready to use. An independent Olostep reference implementation turns those questions into explicit decisions.



The result is only as trustworthy as the explanation of what remains unknown.

Independent portfolio writing prepared for Hamza and the Olostep team.
Not commissioned by, affiliated with, or endorsed by Olostep.

A status is not a dataset.

A developer submits a set of URLs, waits for a batch to finish, and sends the returned text into a downstream workflow. The integration looks successful. Yet there are several different things the application might have proved: that the service finished processing, that every expected item was accounted for, or that every source produced useful content. Treating these as one claim makes a convenient demo and a fragile explanation.

I built the Olostep Integration Reliability Lab to keep those claims separate. It follows one narrow journey: known URLs become a batch; the client polls, enumerates completed and failed items, retrieves Markdown, and exports an evidence report. Olostep documents the underlying batch lifecycle. The additional outcome model and recovery rules are decisions made by this independent client, not new provider guarantees. [1, 2]

Three outcomes, three next steps

Client outcome	What it means	What the reader should do
Complete	Enumeration and reconciliation finished; all content passed the local check.	Evaluate whether the content meets the downstream task.
Partial	Enumeration finished, but an item failed or content remains unresolved.	Inspect the affected identities before accepting a subset.
Incomplete	The client could not finish determining the result.	Preserve evidence and investigate or resume.

The local content check is deliberately narrow: a nonempty Markdown string, without an unresolved hosted-content condition. It does not establish extraction accuracy, factual correctness, or relevance. A report can therefore be complete under the integration policy while still needing a separate content-quality review.

Define what success means before teaching a developer how to print it.

That distinction belongs in documentation as well as code. It gives a developer language for discussing a result with support, a teammate, or a customer without reducing every imperfect outcome to the same generic error.

Recover without creating uncertainty.

The most consequential retry happens before the client has a reliable identifier. If a creation request loses its response, the service may have accepted the work. This lab makes one creation attempt and exposes an ambiguous outcome instead of automatically submitting another batch. A local request ID would not make a repeated creation safe unless the endpoint supplied corresponding idempotency guarantees. [2]

After acceptance, the CLI saves the batch ID, execution mode, and original item identities before polling. A checkpoint lets the client resume reads against existing work. It does not remove the crash window between remote acceptance and local persistence. The documentation makes that remaining uncertainty explicit.

Situation	Client policy
Creation accepted	Save the batch ID and original identity manifest.
Creation outcome unknown	Reconcile before attempting another creation.
Polling budget expires	Keep the resume ID and incomplete evidence.
Transient read failure	Retry within the attempt limit and remaining deadline.
Repeated cursor or wrong identities	Stop; do not present the inventory as verified.
Usage is null	Keep usage unknown; do not infer zero cost.

Reconcile identities, not positions

In the recorded live run, page-2 arrived before page-1. Both identities matched the submitted manifest. Matching by position would have assigned content incorrectly; matching only counts would not have detected a different pair of items. The first URL was also normalized with a trailing slash. Stable custom IDs avoided dependence on either order or raw URL equality. [3]

This is a small observation with a useful teaching consequence: show the original identities alongside the returned inventory. Developers should see why the client trusts the mapping, rather than simply receiving a green completion indicator.

A useful sample makes its limits visible.

The project contains 42 offline tests and nine reproducible scenarios. A fresh local run on October 14, 2026 passed all 42 tests. Those tests exercise client behavior, including partial completion, bounded retries, checkpoint recovery, cursor loops, and unknown usage. They are not evidence that Olostep experienced those failures. [2, 4]

A separate, previously recorded live check used one batch containing two public URLs. It observed four polling reads, real cursor traversal, two nonempty Markdown retrievals, and no retries or failed or unresolved items. Batch information returned null usage fields. That establishes compatibility for this limited lifecycle; it supplies neither a billing measurement nor a production reliability benchmark. The walkthrough remains an offline demonstration. [3]

What this suggests for developer documentation

The most useful companion to an endpoint reference is a guide that helps readers make decisions after their first successful request. For this workflow, that means naming partial results, preserving a resume path, checking item identities, and explaining when a retry would increase uncertainty. The existing reference remains the source for the API contract; the companion teaches a consumer policy around it.

I would start with this bounded tutorial and ask an unfamiliar developer to explain a partial result and recover from an interrupted read. Their hesitation would guide the next revision. Faster onboarding and lower support demand are plausible goals, not outcomes measured by this project. The immediate contribution is concrete: working examples whose explanations, tests, and evidence agree.

SOURCES & REVIEW PATH

1. [Olostep batch workflow - public documentation reviewed October 14, 2026.](#)
2. [Reference implementation and recovery model - reviewed revision 0d989ef.](#)
3. [Bounded live-compatibility record - October 14, 2026.](#)
4. [Offline test suite and verification record.](#)