WORKING PAPER

Absence of Errors Is Not Evidence of Completeness

Six elementary questions that a fiscal product cannot leave implicit when a webshop is a source

Kylian de Groot

15 August 2026

ABSTRACT

A fiscal product that books webshop revenue without a human watching the feed has to answer a question that looks too simple to write down: how do we know that what arrived is everything that happened? The usual answer is operational. The webhooks are green. The sync has run for weeks. The error log is empty. That answer is not a proof. It is the claim that completeness is observable from the stream itself.

It is not. A missing delivery leaves no hole in the observation. It leaves nothing. The empty error log is consistent with a complete feed and with a dropped feed. Those two worlds are indistinguishable to the receiver. A product that treats them as the same world will produce a ledger that looks finished while it is not. The merchant notices. The product does not.

This paper therefore does one thing. It states six questions. It answers each of them by arithmetic on finite sets of integer cents and integer identifiers. It assumes no property of any commerce platform, no webhook specification, and no implementation. The only primitives are sets, addition in $\mathbb{Z}$, and comparison of two independently obtained totals. Where a claim is used later, it is proved first.

The answers are: a push stream cannot prove its own completeness; the proof is a source-side closed statement reproduced locally; the hardest such statement is a payout that also appears on the bank; a monotone identifier makes a gap nameable within one reconciliation cycle; a sync fault must not reach the ledger; and a reconnection is healthy only when the silent interval has been backfilled and the statement closes, not when the token works again.

These facts are what Winstwaker’s development is not allowed to forget. They are written here so that a silent shop cannot be mistaken for a complete shop.

KEYWORDS. completeness; reconciliation; at-most-once delivery; webhooks; payout identity; staging; reconnection; product invariants.

1. Intention

Winstwaker must turn webshop orders, refunds and payouts into append-only fiscal facts. That step looks like import. It is not. Import that is believed because it did not raise an error is a different function from import that is believed because an independent total has been reproduced. Anyone who has counted a till at the end of the day has met the phenomenon. A fiscal product that connects a commerce platform, books revenue, and lets agents “check the period” meets it every day, at amounts that change what is owed, and in a shape that does not look like a bug: the books still add up on the page.

The intention of this paper is therefore not to describe an ideal API integration, and not to describe Winstwaker’s internals. The intention is to write down, without a gap, the answers to six questions that the product’s development will otherwise answer by habit:

1. Can a webhook stream be complete?
2. What is a proof of completeness?
3. Which anchors are there, in order of hardness?
4. How is a gap detected, and within how much time?

5. May a sync fault touch the ledger?
6. What is a reconnection, formally?

Each question is simple. Each wrong answer is a bug that does not look like a bug: the connection is “up”, the queue is empty, and a foreign shop has been silent for eighteen days. Winstwaker cannot be developed on that tolerance. The paper exists to remove the tolerance. If a later design choice contradicts an equality below, the equality wins.

No number in the body is imported from a codebase. Every number is computed in the section that uses it. A companion check may refuse a silent edit of those integers; it is not a source. The shape of Question 6 is a production incident: one shop repaired, two shops silent, empty error logs, a merchant who had to write that the figures were wrong. The integers of that question are the classroom witness of the same shape. Names of merchants and platforms are not used. They are not needed.

2. What is not assumed

The following are *not* taken as given.

- That a live connection is a complete connection. Question 1 exhibits two source sets that produce the same observation.
- That “it has worked for weeks” is evidence. Question 1 shows that a long empty error log is the same object as a short one.
- That a webhook specification supplies exactly-once delivery. Question 1 distinguishes at-most-once, at-least-once, and a closed source statement.
- That a count of received orders is a completeness proof. Question 3 computes a substitution that preserves the count and breaks the sum.
- That a successful OAuth dance restores the feed. Question 6 computes an interval on which the source has events and the receiver has none, after the token works again.
- That an incomplete ingest may be posted and later “fixed in place”. Question 5 records that the ledger of the companion paper is append-only.

IEEE acknowledgements, platform retry policies, and operational runbooks are cited only as related writing. They are not used as lemmas. If a sentence cannot be replaced by a comparison of two integers, it is marked as intention, not as claim.

3. Language

3.1 Events

Work throughout with a finite source set E. Each event $e \in E$ has an identifier $\mathrm{id}(e) \in \mathbb{Z}$ and an amount $\mathrm{amt}(e) \in \mathbb{Z}$, in cents. Identifiers in a single shop’s order stream are distinct. One euro is the integer 100. No decimal point is used in a calculation.

A *period* I is a finite interval of time. Write $E_I = \{e \in E : e \text{ occurred in } I\}$. The source count and source sum on I are

$$N_I := \#E_I, \qquad Y_I := \sum_{e \in E_I} \mathrm{amt}(e).$$

Both are integers. Their pair (N_I, Y_I) is a *source statement* for I.

3.2 Observations

A receiver holds an observed set $O \subseteq E \cup X$, where X is a (possibly empty) set of duplicates or late copies. The error log ε is a finite set of reported faults. The pair (O, ε) is an *observation*. The local count and local sum on I are taken from the distinct identifiers in O:

$$n_I := \#\{\mathrm{id}(e) : e \in O,\ e \text{ occurred in } I\}, \qquad y_I := \sum \mathrm{amt}(e) \text{ over those distinct identifiers.}$$

An observation is *quiet* when $\varepsilon = \varnothing$.

3.3 Delivery

A *push channel* attempts to deliver each $e \in E$ to the receiver. Three modes are named, not assumed as properties of any vendor.

- *At-most-once:* each event is delivered zero or one times. A drop is not reported.
- *At-least-once:* each event that the channel accepts is delivered one or more times. A drop of an event the channel never accepted is still not reported.
- *Exactly-once:* each event is delivered one time. This is not a mode of an unreliable channel. It is the conjunction of at-least-once delivery with a closed source statement that names the events.

The body uses at-most-once as the quiet-drop case, which is the case that looks like health.

3.4 Staging and ledger

The receiver's durable copy of O is *staging*. The append-only journal of the companion paper (de Groot 2025) is the *ledger*. A posting is a map from a staging event to a ledger fact. Question 5 asks when that map may fire.

3.5 Completeness, as an equality

Period I is *complete* when

$$n_I = N_I \qquad \text{and} \qquad y_I = Y_I.$$

That equality is the only completeness predicate in the paper. A quiet observation is not a predicate on E_I.

4. Question 1

Can a webhook stream be complete?

4.1 Two sources, one observation

Let the source set on an hour I be

$$E_I = \{11, 12, 13, 14\}$$

with amounts

$$\mathrm{amt}(11) = 199,\ \mathrm{amt}(12) = 250,\ \mathrm{amt}(13) = 101,\ \mathrm{amt}(14) = 300.$$

The source statement is computed as follows.

$$199 + 250 = 449, \qquad 449 + 101 = 550, \qquad 550 + 300 = 850.$$

$$N_I = 4, \qquad Y_I = 850.$$

Let the channel drop identifier 13 and report nothing. The observation is

$$O_I = \{11, 12, 14\}, \qquad \varepsilon = \varnothing.$$

The local statement:

$$199 + 250 = 449, \qquad 449 + 300 = 749.$$

$$n_I = 3, \qquad y_I = 749.$$

The quiet observation $(O_I, \varnothing)$ is exactly the observation that would have been produced by a different source set $E'_I = \{11, 12, 14\}$ with $N'_I = 3$, $Y'_I = 749$, and a channel that dropped nothing.

Lemma 1 (silence is ambiguous). If a channel does not report drops, then for every observation $(O, \varnothing)$ there exist at least two source sets that produce it: $E = O$, and $E = O \cup \{e^*\}$ for any event e^* the channel dropped. The receiver that sees only $(O, \varnothing)$ cannot name which source set it is in.

Proof. The observation is a function of the delivered subset. The undelivered subset does not appear. An empty error log does not mention it. Two preimages are exhibited above: E_I with four events and E'_I with three. They are not equal. They produce the same quiet observation.

4.2 At-least-once does not close the set

Suppose instead the channel delivers identifier 11 twice and still drops 13. After distinct-identifier collapse,

$$n_I = 3, \qquad y_I = 749$$

as before. Duplication is a problem of idempotency. It is not a proof that $\{13\}$ was empty. At-least-once plus idempotency gives at-most-one *effect* per delivered identifier. It does not give a list of the identifiers that should have been delivered.

4.3 “It has worked for weeks”

Let the same drop occur on day 1, and let days 2 through 21 deliver every subsequent event. The error log on day 21 is still empty. The observation of day 1 is still the three-event set. A long quiet period is the same object, as a proof, as a short quiet period: both are $\varepsilon = \varnothing$. Duration does not add an element to O.

4.4 Answer

No. A webhook stream cannot prove its own completeness. Completeness is a statement about E_I, and E_I is not a function of (O_I, ε). Push is an acceleration: it shortens the time until a delivered event is visible. It is not a source of truth.

> **What Winstwaker’s development must not do.** It must not treat a quiet webhook, a green connection, or a run of quiet weeks as evidence that staging equals the source. Those are observations of the channel. The source is a different set.

5. Question 2

What is a proof of completeness?

5.1 Two statements, one equality

A proof that period I is complete is a reproduction of the source statement. The source is asked for (N_I, Y_I). Staging computes (n_I, y_I). The period is complete when both coordinates match.

On the numbers of §4.1 the source says $(4, 850)$ and staging says $(3, 749)$.

$$4 \neq 3, \qquad 850 - 749 = 101.$$

The missing identifier is not required to *name* incompleteness. The pair of integers is enough to refuse the period. Naming the gap is Question 4. Refusing the period is this question.

5.2 The bank parallel

A cash book that has recorded every till movement the cashier remembers is not a closed till. The closed till is the cash book against the counted drawer. The drawer is an independent statement of the same integers.

A webshop feed is the cash book. The source statement is the drawer. A commerce connection without a closing of (n_I, y_I) against (N_I, Y_I) is a cash book without a count. The books can look complete for the same reason a cash book can: every line that is present is a real line. The missing line is not a wrong line. It is an absent one.

5.3 A period that does close

Let the source be $E_I = \{11, 12, 14\}$ with the amounts of §4.1, and let the channel drop nothing. Then $N_I = 3$, $Y_I = 749$, $n_I = 3$, $y_I = 749$. Both coordinates match. Completeness on this I is proved. Completeness on a larger interval that contains identifier 13 is not. A closed hour is not a closed day.

5.4 Answer

A completeness proof is a source-side closed statement — a count and a sum — reproduced in staging. Nothing else in this paper is called a proof. In particular, $\varepsilon = \varnothing$ is not a proof. It is the hypothesis of Lemma 1.

> **What Winstwaker's development must not do.** It must not mark a period complete because ingest finished, the cursor advanced, or the last page of a listing had no successor. Those are statements about the walk. They are not (N_I, Y_I).

6. Question 3

Which anchors are there, in order of hardness?

Hardness is the number of independent parties that would have to be wrong for a false close to stand. An anchor that can be checked against a party the product does not control is harder than an anchor the product computes from its own log.

6.1 Payout against the bank

Let the orders of I be the four amounts of §4.1, totalling 850. Let there be one refund of 50 and one fee of 25. The payout identity is

$$850 - 50 = 800, \qquad 800 - 25 = 775.$$

The platform's payout is 775 cents. The bank credits 775 cents. Three totals, two institutions, one integer.

If identifier 13 is missing in staging, the local reconstruction is

$$749 - 50 = 699, \qquad 699 - 25 = 674.$$

$$674 \neq 775, \qquad 775 - 674 = 101.$$

The bank did not drop identifier 13. The payout still contains it. The disagreement is the missing order, in cents. This is the hardest anchor in the paper: the ledger, the platform, and the bank must all print 775, or the period is not closed.

6.2 Period sum against the source

The source statement $Y_I = 850$ against $y_I = 749$ is a two-party check. It does not need the bank. It does need the source to answer a closed query for I. It is weaker than §6.1 because a source that omits 13 from both its feed and its statement would close: then $N_I = 3$, $Y_I = 749$, and staging reproduces both, so Question 2 is satisfied by a lie that is internally consistent. The bank payout would then fail §6.1, which is why the payout sits above the period sum.

The hierarchy guarantees independence only where a second party exists. Revenue that settled through a platform payout has the bank. Manual orders, and bank transfers that never enter the platform, have no such party. For those streams the period sum is the hardest anchor in this paper, and it leans on the honesty of the same source that produced the feed. That is not a fault of the hierarchy. It is the bound on what the hierarchy can prove.

6.3 Period count

Suppose staging, in a different fault, receives four identifiers but substitutes the amount of identifier 11: 99 instead of 199. Then $n_I = 4 = N_I$, and the sum is

$$99 + 250 = 349, \qquad 349 + 101 = 450, \qquad 450 + 300 = 750.$$

$$n_I = N_I, \qquad 750 \neq 850, \qquad 850 - 750 = 100.$$

A count-only close would accept this period. A sum close would not. Counts are a lower bound on hardness: they catch missing identifiers and do not catch amount substitution.

6.4 The empty error log

$\varepsilon = \varnothing$ is a one-party check. The party is the receiver. Lemma 1 says it has at least two preimages. It is not an anchor. It is listed so that it cannot be inserted above the others by habit.

6.5 Order of hardness

1. Payout identity against the bank: $\sum \text{orders} - \sum \text{refunds} - \sum \text{fees} = P = B$.
2. Period sum $y_I = Y_I$.
3. Period count $n_I = N_I$.
4. Quiet observation $\varepsilon = \varnothing$.

A product may use a weaker anchor as a *hint*. It may close a period only on an anchor that is at least as hard as the period sum. Revenue that settled through a platform payout must close on the payout identity. Streams with no second party close on the period sum, and the product must not pretend that close is independent.

> **What Winstwaker's development must not do.** It must not rank a quiet log above a payout, and it must not treat a matching count as a matching sum. A payout that does not reconstruct from orders minus refunds minus fees is an open period, even if every webhook in the log was a 200. Where no payout and no bank line exist, it must not call the period-sum close an independent proof.

7. Question 4

How is a gap detected, and within how much time?

7.1 Monotone identifiers

Let the processed identifier set on a shop be $P \subset \mathbb{Z}$, and write $x = \max P$ when P is non-empty. Let the source report that the maximum identifier assigned in I is y. If identifiers are issued in increasing order without holes at the source, the candidate interval is $\{x + 1, \dots, y\}$ when $y > x$, and the *named gap* is

$$G = \{k \in \mathbb{Z} : x < k \le y\} \setminus P.$$

On the observation of §4.1, $P = \{11, 12, 14\}$ and $y = 14$. Then $x = 14$, and

$$\{k : 14 < k \le 14\} = \varnothing$$

does not name the hole. The maximum of P is not the right left-hand endpoint when P itself has holes. The correct left endpoint is the least integer that was skipped. Write $m = \min(\mathbb{Z}_{>0} \setminus P)$ after translating identifiers so that the shop's first identifier is 1, or, without translation, take the missing set inside $[\min P, \max P]$:

$$G = \{k \in \mathbb{Z} : \min P \le k \le \max P\} \setminus P.$$

Here $\min P = 11$, $\max P = 14$,

$$\{11, 12, 13, 14\} \setminus \{11, 12, 14\} = \{13\}.$$
$$G = \{13\}.$$

Identifier 13 is named. Its amount, from §4.1, is 101. The period sum already refused the close ($850 \neq 749$). The monotone scan names the row.

Naming is not concluding. Platforms skip identifiers for cancelled checkouts and test orders: the integers are issued, the events are not in E_I. Then G is non-empty and the source statement still closes. Let the source issue $11, 12, 13, 14$ and let 13 be a cancelled checkout, never an order. Staging holds $P = \{11, 12, 14\}$, so $G = \{13\}$ as above, while

$$N_I = 3, \qquad Y_I = 749, \qquad n_I = 3, \qquad y_I = 749.$$

Question 2 is satisfied. G is a candidate. A gap in the numbering is a question to the source, not a conclusion. An alarm that fires on G alone false-fires on every skipped checkout.

If the source later reports $y = 15$ and 15 has been processed, $P = \{11, 12, 14, 15\}$, the same interior formula still yields $G = \{13\}$. A new maximum does not heal an interior hole, and does not decide whether 13 was an event.

7.2 When identifiers are not monotone

If identifiers are not an interval of integers — opaque tokens, per-shop namespaces that reset, several shops interleaved without a namespace, or a source that skips integers on purpose — then G as a set difference of integers is either undefined or larger than the missing events. Completeness then falls back to Question 2: the source statement (N_I, Y_I). A product that cannot name a missing *event* is not excused from refusing the period. It is excused from treating a missing *integer* as that event until the source has answered the question.

7.3 Maximum unobserved time

Let τ be the interval between two source-statement checks. A gap may exist. It may not exist *unobserved* for longer than τ. That bound is a product invariant, not a quality target.

On a push-only design, the next check never comes, and $\tau = \infty$. Lemma 1 then says a drop can remain unnamed for the life of the connection. On a design that asks the source for (N_I, Y_I) every τ, the drop of identifier 13 is flagged at the first check after the hour I.

The invariant does not say that staging is complete at every moment. It says that incompleteness is a visible state before the next close is attempted.

7.4 Answer

A gap in a monotone identifier interval is a candidate: the integer set difference between what was issued and what staging holds. It becomes a missing event only when the source says the identifier belonged to E_I. Otherwise completeness is the failure of $n_I = N_I$ and $y_I = Y_I$. In both cases the maximum unobserved time for an unanswered candidate, or for a failed close, is one reconciliation cycle.

> **What Winstwaker's development must not do.** It must not wait for a merchant to report that the figures are wrong, and it must not alarm on a skipped integer without asking the source whether that integer was an event. A named candidate or a failed source statement is a work item at the first cycle that can see it. A gap that survives a cycle unasked is a product fault, not an operational inconvenience.

8. Question 5

May a sync fault touch the ledger?

8.1 Two stores

Staging is mutable operational state: cursors, retries, observed sets, quiet logs. The ledger is append-only fiscal state. A correction in the ledger is a new opposite fact, not an edit (de Groot 2025). Therefore an event that is posted from an incomplete O_I becomes a fact that cannot be made not to have been posted. The missing event, when it later arrives, is a second fact. The pair is not the single fact the source issued.

If identifier 13 is posted never, and identifiers 11, 12, 14 are posted as sales, the ledger's revenue for I is 749. When 13 arrives later, a new posting of 101 makes 850. That is the right sum and the wrong history if a return for I was already prepared on 749. The companion paper's Question 9 then applies: the amendment is the difference of two filings, not a silent edit of the first.

The clean procedure is earlier. Let posting be allowed only from a period that has already closed under Question 2 (and, where a payout exists, under §6.1). Then a drop of 13 leaves staging incomplete, the period unclosed, and the ledger untouched. The repair is a backfill into staging, a close of $(4, 850)$, and then a posting of four facts, once.

8.2 The zero posting of an incomplete hour

On the observation of §4.1,

$$n_I = 3 \neq 4 = N_I.$$

The number of ledger facts created from I is required to be 0 until equality holds. After backfill of identifier 13,

$$n_I = 4, \qquad y_I = 749 + 101 = 850 = Y_I,$$

and posting may fire. The ledger then sees four facts totalling 850, which is the source statement, which is the only statement the ledger is allowed to receive from a webshop.

8.3 Answer

No. A sync fault must not touch the ledger. Import is staging. Booking is a later map, and it fires only on a closed period. This is the join with the companion paper: the immutable ledger receives only input that has already been proved complete.

> **What Winstwaker's development must not do.** It must not post an order because the webhook was well-formed. Well-formed is a property of a row. Completeness is a property of a period. The ledger stores facts, not rows.

9. Question 6

What is a reconnection, formally?

9.1 A broken interval

A credential is a predicate on time: valid or not. Let it be valid on $[0, t_b)$, invalid on $[t_b, t_r)$, and valid on $[t_r, \infty)$. Write $J = [t_b, t_r)$ for the *broken interval*. During J the source still accepts orders. The push channel, lacking a valid credential, delivers nothing and reports nothing that the receiver classifies as a gap in E_J. The observation on J is

$$O_J = \varnothing, \qquad \varepsilon = \varnothing$$

or, worse, ε contains authentication faults that stop once a new token is issued, and are then cleared.

A reconnection at t_r is a new valid credential. It proves that the receiver can speak to the source *now*. It does not prove that $E_J \subseteq O$. By Lemma 1, the quiet observation on J is consistent with $E_J = \varnothing$ and with a non-empty E_J that was never delivered.

9.2 Health is a close, not a token

Define *healthy on J* by the predicate of §3.5, after a backfill:

$$\mathrm{Healthy}(J) :\iff n_J = N_J \ \wedge \ y_J = Y_J.$$

OAuth success is not a conjunct. It is a precondition of asking the source for (N_J, Y_J) and for the listing that fills O_J. A connection whose token works and whose $\mathrm{Healthy}(J)$ is false is a live connection to an incomplete staging set.

9.3 Three shops, one administration

A merchant runs three shops on one platform, booked in one administration: a home shop H and two foreign shops A and B. A credential repair is applied to H at t_b. H resumes. A and B do not. The receiver's quiet log is empty. The home shop's feed is live. The administration looks complete: every row that is present is a real row, and no error is on the page.

Let J be eighteen days. Let the source statements on J be

$$\begin{aligned} N_J^H &= 30, \quad Y_J^H = 9000, \\ N_J^A &= 12, \quad Y_J^A = 2808, \\ N_J^B &= 7, \quad\ Y_J^B = 796. \end{aligned}$$

Staging, after the repair of H, holds

$$\begin{aligned} n_J^H &= 30, \quad y_J^H = 9000, \\ n_J^A &= 0, \quad\ y_J^A = 0, \\ n_J^B &= 0, \quad\ y_J^B = 0. \end{aligned}$$

Shop H closes on J. Shops A and B do not:

$$0 \neq 12, \qquad 0 \neq 7.$$

The missing count is $12 + 7 = 19$. The missing sum is

$$2808 + 796 = 3604.$$

The administration's observed revenue on J is 9000. The source's revenue on J is

$$9000 + 2808 = 11808, \qquad 11808 + 796 = 12604.$$

$$12604 - 9000 = 3604.$$

The books are short 3604 cents. The error log is empty. The live shop is healthy. Completeness of the administration is the conjunction of the three shop predicates. A conjunction is false when one conjunct is false. Here two are false.

At t_r the tokens of A and B are restored. OAuth succeeds. Staging is still

$$n_J^A = 0, \qquad n_J^B = 0$$

until a backfill of J is run. If health is defined as "the token works", the connection is declared healthy at t_r with 19 events still missing. That declaration is the fault. The merchant's later letter — that the figures are wrong — is the detection method of a product that set $\tau = \infty$ on A and B.

After backfill, suppose A and B reproduce their source statements: $n_J^A = 12$, $y_J^A = 2808$, $n_J^B = 7$, $y_J^B = 796$. Then $\mathrm{Healthy}(J)$ holds for all three shops, the administration's sum is 12604, and posting (Question 5) may fire on J. Not before.

9.4 Reconnection, as a procedure

A reconnection is the triple

1. a new valid credential at t_r,
2. a backfill of $J = [t_b, t_r)$,
3. a close of $\mathrm{Healthy}(J)$.

The third step is the definition of done. The first step is a means. A product that stops after the first step has performed an authentication. It has not performed a reconnection.

9.5 Answer

A reconnection is a backfill of the broken interval plus a close of the source statement on that interval. A working token is a precondition. It is not the proof.

> **What Winstwaker's development must not do.** It must not call a shop healthy because OAuth succeeded, and it must not repair one shop of a multi-shop administration without closing the others on the same J. The silent shop is the incomplete shop. The live shop does not speak for it.

10. What the six answers require of Winstwaker

The questions are simple. The product consequences are not optional.

1. **Do not believe a quiet stream.** Webhooks accelerate delivery. They do not close a period.
2. **Close against a source statement.** A period is complete when $(n_I, y_I) = (N_I, Y_I)$. A finished cursor is not that equality.
3. **Rank anchors by hardness.** Payout against the bank, then period sum, then period count, then the error log. Never the reverse.
4. **Bound the unobserved gap.** A named hole is a question to the source, not a conclusion. A failed close is a work item within one reconciliation cycle. $\tau = \infty$ is a product fault.
5. **Stage, then book.** The ledger receives a webshop event only from a closed period. A well-formed webhook is not a posting licence.
6. **Reconnect by closing J.** After every broken credential, backfill $[t_b, t_r)$ and require $\mathrm{Healthy}(J)$. A new token is not health. A repaired home shop is not a repaired administration.

These six sentences are the development rule. The paper is the proof that they are not style. Each is the negation of a procedure that produces a ledger which looks complete and is not. The merchant letter is what remains when the product sets τ to infinity; it is not a detection method the product is allowed to rely on.

11. Related writing

The companion paper (de Groot 2025) treats the map from integer cents to a whole-euro return. The present paper treats the map into those cents from a webshop. The two maps meet at Question 5: the ledger is append-only, so an unproved ingest is not a fact that can be quietly replaced. Question 9 of the companion paper is what remains if Question 5 is ignored.

Lamport, Shostak and Pease (1982) and the two-generals observation that precedes them record that an unreliable channel cannot confirm delivery to both parties. The present paper does not use that literature as a lemma. Lemma 1 is the same elementary fact written on four identifiers: a drop that is not reported is not visible. The fiscal consequence is the only addition — the quiet drop is booked as completeness unless a source statement is asked.

Double-entry cash control is older than any of this. Pacioli (1494) already treats the journal as a conservation law. The till-count at the close of the day is the source statement of a cash drawer. A commerce payout printed on a bank statement is the same object with a different institution on the other side.

Platform retry policies and webhook signatures are operational hygiene. They reduce the size of J and the frequency of drops. They do not change Lemma 1. A signed quiet log is still a quiet log.

12. Conclusion

Six questions, all of a kind that is easy to skip. The answers, all computed:

	Question	Sides	Integers
1	Can a push stream prove completeness?	no	quiet $\{11, 12, 14\}$ fits $N = 4$ and $N = 3$
2	Proof of completeness?	$(n, y) = (N, Y)$	$3 \neq 4$, $749 \neq 850$
3	Hardest anchor?	payout = bank	$850 - 50 - 25 = 775$; missing 13 gives 674
4	Named gap, bound τ?	candidate $\{13\}$	ask the source; skipped checkout still closes $749 = 749$
5	Sync fault into the ledger?	no	post 0 facts until $n = 4$, $y = 850$
6	Reconnection = OAuth success?	no	H: $9000 = 9000$; A: $0 \neq 12$; B: $0 \neq 7$; admin $9000 \neq 12604$

Winstwaker’s development needs these answers in writing because the questions look too simple to deserve writing. The cost of leaving them implicit is a complete-looking administration whose foreign shops have been quiet since the last repair. The paper is the proof obligation, fully expanded, so that a quiet log can be refused as evidence.

References

de Groot, K. (2025). *When summation and rounding fail to commute*. Working paper.

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, *4*(3), 382–401.

Pacioli, L. (1494). *Summa de arithmetica, geometria, proportioni et proportionalita*. Venice.

Appendix. Calculation ledger

Every sum and set difference used in the body, in one place. Each line is recomputed from the two terms on the right.

Amounts of the four-event hour

Identity	Expansion
$199 + 250 = 449$	$200 + 249$
$449 + 101 = 550$	$449 + 100 + 1$
$550 + 300 = 850$	$550 + 300$
$449 + 300 = 749$	$450 + 299$; $850 - 101 = 749$

Identity	Expansion
$4 - 3 = 1$	missing count, §4.1
$850 - 749 = 101$	missing amount, identifier 13

Payout identity

Identity	Expansion
$850 - 50 = 800$	refund
$800 - 25 = 775$	fee; payout and bank
$749 - 50 = 699$	local, refund
$699 - 25 = 674$	local, fee
$775 - 674 = 101$	payout hole equals identifier 13
$99 + 250 = 349$	substituted amount, §6.3
$349 + 101 = 450$	
$450 + 300 = 750$	
$850 - 750 = 100$	count matches, sum does not
$N_I = 3 = n_I, Y_I = 749 = y_I$	source omits 13 from feed and statement; Question 2 closes; payout still $674 \neq 775$

Named gap

Identity	Expansion
$\{11, 12, 13, 14\} \setminus \{11, 12, 14\} = \{13\}$	interior hole
$\mathrm{amt}(13) = 101$	from §4.1
$G = \{13\}$ and $749 = 749$	13 is a cancelled checkout, not an event; candidate, not a close failure

Three shops on J

Identity	Expansion
$12 + 7 = 19$	missing orders, shops A and B
$2808 + 796 = 3604$	$2808 + 800 - 4$
$9000 + 2808 = 11808$	home plus shop A
$11808 + 796 = 12604$	plus shop B
$12604 - 9000 = 3604$	administration shortfall
$0 \neq 12$	shop A not closed
$0 \neq 7$	shop B not closed
$30 = 30$	shop H closed; does not speak for A, B

Posting gate

Identity	Expansion
$749 + 101 = 850$	backfill of identifier 13
$n_I = 4 = N_I$	after backfill
$y_I = 850 = Y_I$	after backfill; posting may fire