# NACHO INTEGRATION LAYER — Interface Control Document

### How the POS and the LOS become one system

| | |
|---|---|
| **Document** | `INTEGRATION_version1.0.0` |
| **Version** | **1.0.0** — the baseline. See §7. |
| **Issued** | 2026-09-01 |
| **Companion documents** | `POS_version1.0.0` · `LOS_version1.0.0` . **All three are one specification.** |
| **Audience** | Both halves of the development team, together |
| **Stage table version** | **11.4.1**, published 2026-08-20 |

## Why this document exists

⭐ **Roughly a third of the defects in a two-halved system like this live in the seam rather than in either half.** Both teams build correctly to their own document, both are right, and the product is still broken — because they each made a reasonable assumption about the other side and the assumptions did not match.

This document exists so that neither team has to assume anything.

> ⭐⭐ **The single most expensive failure mode in a POS/LOS programme is both sides implementing the same logic and then disagreeing at runtime.**

Everything here is written to prevent that one thing. Every shared concern has **exactly one owner** (§1). Every value that crosses has **exactly one shape** (§3). Every borrower-facing word derived from a lender-side state comes from **exactly one published table** (§2).

## How to read it

| § | What it covers | Read it if you are… |
|---|---|---|
| **1** | ⭐ The ownership matrix — one system of record per concern | Everyone, first |
| **2** | ⭐⭐ The published stage table — the most important contract in the product | Everyone |
| **3** | The push contract: POS → LOS | Both teams |
| **4** | The event contract: LOS → POS | Both teams |
| **5** | ⭐ Compliance boundaries that cross the seam | Both teams, and compliance |
| **6** | ⚠️ Conflicts still to resolve before code | Product, leadership |
| **7** | Change control for the seam specifically | Everyone |

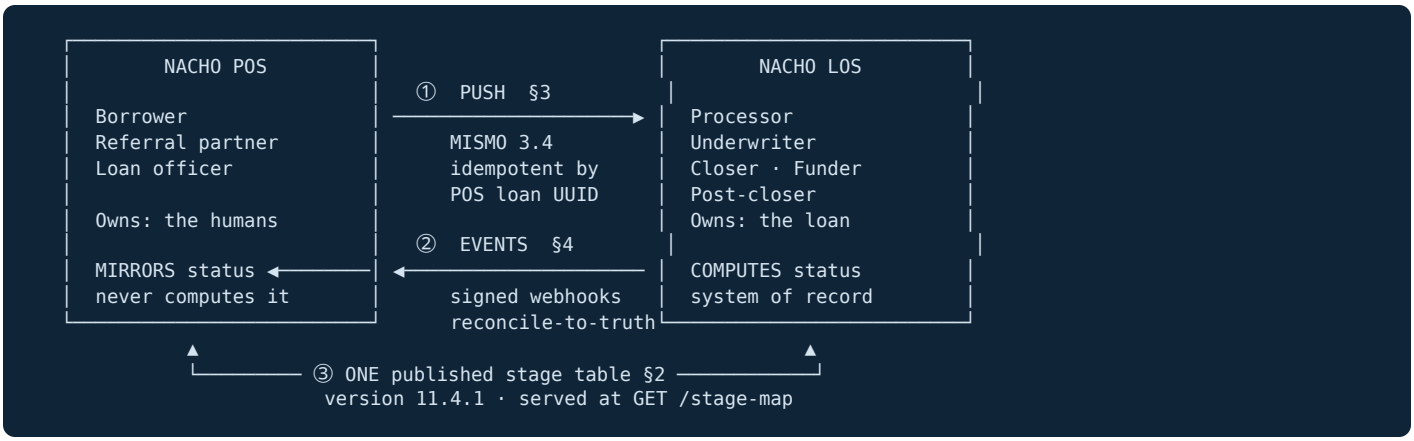
## The three rules of the seam

**1 · One owner per fact.** If you are writing code that computes a value the other half also computes, **stop.** One of you is wrong, and §1 says which.

**2 · The POS mirrors; it never computes.** The POS renders the stage the server told it. It does not derive a stage of its own from any combination of fields. §2 exists to make this possible.

**3 · Reconcile to truth, not deltas.** When the two halves disagree, the answer is never to apply a difference. It is to re-read the authoritative side and adopt it. §4 specifies this.

### The seam in one picture




# 1 · The ownership matrix

⭐ **Rule of thumb: the POS owns the humans. The LOS owns the loan.**

Every row below has **exactly one** system of record. If both halves need a fact, one of them reads it and the other one owns it — never both.

| Concern | System of record | Note |
|---|---|---|
| **Borrower identity and authentication** | **POS** | ⛔ **The LOS never authenticates a borrower.** |
| **Referral partner identity and authentication** | **POS** | The partner is a real authenticated seat, on the POS side |
| **Application capture (the URLA experience)** | **POS** | The POS owns the *capture*; the LOS owns the **authoritative record after handoff** |
| ⭐ **Loan status / milestone** | **LOS** | **Authoritative.** The POS mirrors and presents. **It never computes a stage.** |
| ⭐ **Underwriting decisions** | **LOS** | The POS decides nothing. This is a compliance boundary as much as an architectural one. |
| **Conditions** | **LOS** | The POS renders them as plain-language needs-list items |
| **Documents** | **POS captures → LOS stores** | ⭐ **The LOS is the document system of record.** |
| **Disclosure generation and delivery** | **LOS** | ⚠️ See §6 — this is an open conflict, resolved provisionally |
| ⭐ **The TRID six-piece trigger timestamp** | **POS detects → LOS owns the deadline** | See §5.1. The single most nuanced row in this table. |
| **Borrower status communications** | **POS** | Driven off LOS events |
| **Referral-partner experience** | **POS** | Entirely POS-side |
| **AI borrower assistant** | **POS** | ⭐ **Reads loan state; writes nothing to the loan.** |
| **AI operations assistance** | **LOS** | Suggests; never decides |
| **Pricing** | **The pricing engine** | ⭐ The POS shows **estimates only**, never authoritative pricing. The LOS lock desk is the system of record for the locked rate. |
| **Rate lock — the request** | **POS or LOS** | Either may originate it |
| ⭐ **Rate lock — the confirmation** | **LOS** | Only `lock-confirm` makes a lock real |
| **Fee templates and the 2015 itemisation** | **LOS** | Feeds the LE, ICD, CD and the wire |
| **Attribution — loan officer and partner** | **POS** | Bound at first touch, resolved server-side, **follows the loan not the person** |
| **Consent records — contact (SMS, email)** | **POS**, on the **Person** | Revoking texts revokes them for the human across every file |
| **Consent records — E-SIGN and credit** | **POS**, on the **Loan** | Per-loan, always |
| **The audit trail** | ⭐ **Both, independently** | Each half keeps its own append-only trail. Neither is a subset of the other. |
| **Audit trail for disclosures** | **The document vendor, today** | ⚠️ Long-term ownership is an open decision — §6 |
| **Tenant and branch structure** | **LOS** | The POS reads it |
| **Seat, role and capability model** | **LOS** | ⭐ One shared model. The POS enforces the same capabilities on its own surfaces. |

## 1.1 ⭐ The two rows most often got wrong

### Loan status

**The LOS computes it. The POS renders what it was told.**

This is not a style preference, and the product has already been bitten by it once. When both sides could decide what a milestone meant, a file at the *Approved / Suspended* milestone told the borrower **"You are approved. A short list of items finishes it."** A **suspended** file said that.

§2 is the fix, and it is structural rather than procedural: the POS's borrower-facing words are **derived from the published table by construction**, so a stage with no row **goes silent** rather than inventing a sentence. That is the safe direction to fail in.

### The AI assistant

⭐ **The borrower assistant reads loan state and writes nothing to the loan.** It cannot change a stage, clear a condition, set a fee, or record a consent. Every one of those is an act by a licensed or authorised human, and the assistant's inability to perform them is a **design property, not a current limitation.**

## 1.2 What "one owner" means in practice

| Situation | Wrong | Right |
|---|---|---|
| Both halves need the borrower's name | Each stores its own copy and they drift | The LOS holds the authoritative record post-handoff; the POS reads it back |
| The POS wants to show "in underwriting" | Derive it from `submittedAt` and `decisionAt` | Read `status` and look it up in the published table (§2) |
| The LOS needs to know the partner on the file | Re-derive from a query parameter | Receive it in the push payload (§3), where the POS resolved it server-side |
| Both need to know when the application legally began | Each computes six-piece completeness | ⭐ The POS detects it; the timestamp crosses as a first-class event; **the LOS owns the deadline** (§5.1) |

# 2 · ⭐⭐ The published stage table

**This is the single most important contract in the product.** It is the reason a borrower and a processor can never be told two different things about the same loan.

| | |
|---|---|
| **Version** | **11.4.1** |
| **Published** | 2026-08-20 |
| **Owner** | ⭐ **The LOS.** The milestone belongs to the LOS; the POS mirrors and presents it. |
| **Served at** | `GET /stage-map`, **with its version** |
| **Rule** | It is **one published, versioned table owned in one place** — ⛔ **not translation logic hardcoded in two codebases.** |

## 2.1 Why it is a served table rather than a shared constant

Because a shared constant becomes two constants the first time either side ships independently.

This build had exactly that: the LOS spine carried one copy of the borrower's words, and the POS's localisation file carried a second, Spanish copy. **They agreed on a sentence that was wrong in both languages.**

⭐ **The POS takes its borrower-facing words FROM this table by construction, not by discipline.** Delete a row and the stage **goes silent to the borrower** rather than inventing a sentence.

## 2.2 The four rules a row can carry

| Rule | Meaning |
|---|---|
| `mirror` | An ordinary borrower-facing stage. Shown on the tracker; a milestone change generates a borrower message. |
| `item-only` | ⭐ Not surfaced under its own name. Surfaced as **the specific item needed.** |
| `not-surfaced` | Not surfaced to the borrower at all. **The tracker holds at the last visible stage and no message is generated.** |
| `compliance-controlled` | Compliance-controlled messaging only, and **all borrower automation halts — synchronously and confirmed** (§5.3). |

## 2.3 The table, version 11.4.1

| LOS milestone | Stage key(s) | Sys | Rule | What the borrower sees |
|---|---|---|---|---|
| Application Started | `started` | POS | `mirror` | **"Application started"** — *Your application is saved. Pick up where you left off any time.* |
| Pre-application — 1003 incomplete | `submitted` | POS | `mirror` | **"Application submitted"** — *We have your application and your loan officer has been notified.* |
| Application Taken | `app-taken` | LOS | `mirror` | **"Application received"** — *All six required pieces are in. Your disclosures are on the way.* |
| Processing | `processing` | LOS | `mirror` | **"We're reviewing your file"** — *A processor is building your file and ordering services.* |
| ⭐ Initial Submission → Approved / Suspended | `initial-sub`, `approved` | LOS | `mirror` | **"In underwriting"** — *Your file is with an underwriter. We will tell you the moment there is anything you need to do.* |
| Resubmitted / Condition Submission | `resubmission` | LOS | `mirror` | **"Finishing your conditions"** — *What you sent is being reviewed against your conditions.* |
| Clear to Close | `ctc` | LOS | `mirror` | **"Approved — scheduling your closing"** — *Everything is signed off. Closing is being scheduled.* |
| Docs Out (Sent) | `docs-out` | LOS | `mirror` | **"Your closing documents are out"** — *Closing documents have been sent to the settlement agent.* |
| ⭐ Closed / Funded | `closed`, `funded` | LOS | `mirror` | `funded` → **"Funded"** — *Your loan has funded. Welcome home.* · `closed` → **"Signed — funding is next"** — *You have signed. Funding is next.* |
| Post Closing / Purchased | `post-closing`, `purchased` | LOS | **`not-surfaced`** | ⛔ **Nothing.** The tracker holds at the last borrower-visible stage. |
| ⭐ Suspended | `suspended` | LOS | **`item-only`** | ⛔ Never the word "suspended". **The specific item needed**, and nothing else. |
| ⭐ Withdrawn / Denied | `withdrawn`, `denied` | LOS | **`compliance-controlled`** | A fixed notice — see §2.6 — and **all automation halts.** |

**Every row carries an English and a Spanish label and blurb.** Spanish is a launch-tier requirement ( `POS_version1.0.0` M1), and the translation lives in **this table**, not in a separate localisation file that can drift from it.

## 2.4 ⭐⭐ The three rows that exist to prevent a specific harm

### "In underwriting" — two milestones, one borrower row

**Initial Submission** and **Approved / Suspended** map to **one** borrower row, and that row says **"In underwriting."**

⭐ **The word "approved" does not reach a borrower until Clear to Close.** This row is the direct fix for the live defect in which a **suspended** file told the borrower they were approved. Collapsing the two milestones is not a simplification — it is the control.

### Suspended — the item, never the state

⚠️ **"Suspended" is a normal operational word and an alarming borrower word.** The mapping table is where that judgment gets made **once**, rather than in whichever screen renders it.

The borrower sees **the specific item needed.** They never see the state.

⛔ **Do not invent suspension language.** The POS specification contains zero occurrences of the word "suspend"; this row is the only instruction that exists anywhere for what a suspended borrower is told, and nothing beyond it should be written.

### Closed vs Funded — one row, one override

The two milestones share a row, **and the `closed` key overrides the label**, because ⭐ **a file that has closed but not funded is not funded.** On a refinance of a primary residence a three-day rescission period sits between them.

⭐ **The override lives here, in the published table — not in a screen.** That is the pattern for every such exception.

### Post Closing / Purchased — deliberately silent

This build used to surface these as ordinary borrower blurbs — *"Wrapping up"* and *"Complete."* **Both are gone.** The borrower is not told. **Nothing new is said and nothing is invented.**

## 2.5 The status field on the seam

The loan object exposes `status`. When the lender state is Suspended it additionally exposes:

```
suspension: { reason, owner, since }
```

⭐ **These names are the contract the POS reads.** They are not internal shorthand and **they do not get renamed** without a MAJOR version bump on this document and on both halves (§7).

## 2.6 The compliance-controlled notices, verbatim

These are fixed text. ⛔ **They are not templates to be improved.**

**Withdrawn** — *"Application withdrawn" / "Solicitud retirada"*

> Your application has been withdrawn. Nothing further is owed, and no automated updates will be sent on this file.
>
> *Su solicitud fue retirada. No se debe nada más y no se enviarán actualizaciones automáticas sobre este expediente.*

**Denied** — *"Application denied" / "Solicitud denegada"*

> A decision has been issued. Your Adverse Action notice explains it and your rights. No automated updates will be sent on this file.
>
> *Se emitió una decisión. Su aviso de Acción Adversa la explica junto con sus derechos. No se enviarán actualizaciones automáticas sobre este expediente.*

⭐ Note what these notices do **not** do: they do not explain the reason. **The Adverse Action notice does that**, and it is a formal document with legally specified content ( `LOS_version1.0.0` §9.4). The POS points at it and says nothing more.

## 2.7 Versioning this table

| Change | Bump |
|---|---|
| A blurb reworded, no rule change | **PATCH** — `11.4.2` |
| A new row added | **MINOR** — `11.5.0` |
| ⭐ A row's **rule** changes, a key moves between rows, or a field name on the seam changes | **MAJOR** — `12.0.0` , **and a MINOR bump on both** `POS_version` **and** `LOS_version` |

**Both halves read the version.** A POS that receives a stage-map version it does not recognise must **degrade to silence on unknown rows**, not guess — same failure direction as a missing row.

# 3 · The push contract: POS → LOS

## 3.1 What crosses, and when

| Trigger | What is sent | Shape |
|---|---|---|
| `application.submitted` | The full loan file plus documents | ⭐ **MISMO 3.4 (ULAD)**, the industry-standard XML dataset |
| Subsequent borrower edits | Field-level updates | The same schema, partial |
| New document uploaded | The document, its classification, and the condition it satisfies | Binary + metadata |
| Consent recorded | The consent record, with its **exact text and version** | JSON |
| Rate-lock **request** | Intent and timestamp | JSON |

## 3.2 ⭐ Idempotency — keyed by POS loan UUID

**SM-01.** Every push carries the **POS loan UUID** as its idempotency key. A retry with the same UUID **updates**; it never creates a second file.

⚠️ This matters more than it sounds. Without it, a webhook retry during a network blip creates a duplicate loan file, and duplicate loan files in a mortgage system produce duplicate credit pulls, duplicate disclosures and duplicate TRID clocks.

## 3.3 ⭐ The document handoff — the rule that removes the loudest complaint

**D8, in the POS document.** Every document arrives in the LOS **already:**

1. **Classified** — pay stub, W-2, bank statement, identification.
2. **Named to convention** — the LOS's convention, not the borrower's filename.
3. **Attached to the condition it satisfies.**

⭐ **The test for this whole contract:** *a processor should never have to ask the borrower for something the borrower already gave us.* That single failure is the loudest complaint borrowers have about every product on the market, and it is **entirely a plumbing problem.**

## 3.4 What must never be re-keyed

These arrive from the POS and are **never retyped** on the LOS side. Each is a mapping chain ( `LOS_version1.0.0` §6.2):

| Value | Where it flows |
|---|---|
| **Property address** | USPS verify → appraisal order → title order → flood → CD → legal/exhibit → Deed of Trust → MERS |
| **The 236 URLA fields** | The 1003, AUS, DTI, conditions, disclosures |
| **Borrower identity and contact** | The Loan Summary, all correspondence |
| **Income and employment** | Income calculation → 1003 → AUS → DTI → conditions |
| **Assets and their source** | Underwriting, reserves, cash-to-close |
| **Attribution** — loan officer, referral partner, campaign | Reporting, partner scorecards, compensation |
| ⭐ **The six-piece trigger timestamp** | The TRID clock — see §5.1 |
| **Consents** — with exact text and version | The consent ledger on both sides |

## 3.5 ⭐ The `LosAdapter` interface

**SM-08.** The LOS sits behind an interface with **two implementations**, so that changing the LOS is a new implementation rather than a rewrite.

```
interface LosAdapter
  ├── pushApplication(mismo34, idempotencyKey)     → losLoanId
  ├── pushUpdate(losLoanId, partialMismo34, key)   → ack
  ├── pushDocument(losLoanId, blob, metadata, key) → documentId
  ├── pushConsent(losLoanId, consentRecord, key)   → ack
  ├── requestLock(losLoanId, intent, timestamp)    → requestId
  ├── getLoan(losLoanId)                           → loan  (status, conditions, fees…)
  └── subscribe(eventTypes, endpoint)              → subscription
```

**Two implementations at launch:** the interim LOS in use today, and NACHO LOS.

⭐ **CM-08 requires the reverse too.** A licensee may arrive with their own front end, so the seam is **an open contract in both directions** — not a private channel between two halves that only work together.

⚠️ **Do not leak vendor response shapes past the adapter.** The moment business logic branches on a specific LOS's field naming, the interface has stopped being an interface.

### 3.6 Multi-tenancy across the seam

**SM-11.** Tenant identity crosses on **every** call, and both halves enforce it independently.

⛔ **The POS must never rely on the LOS to scope a tenant, and the LOS must never rely on the POS.** Two independent enforcements is the point — a single enforcement is a single point of failure on the one boundary where a failure is a data breach.

⭐ **Credentials are tenant-scoped** (CM-03). Agency credentials — DU, LPA, FHA Connection, MERS — are issued **per lender**. A platform holding one central set cannot be licensed at all.

---

## 4 · The event contract: LOS → POS

### 4.1 The event set

**SM-02.** The POS subscribes; the LOS publishes. Every meaningful state change is a typed event.

| Event | Carries | What the POS does |
|---|---|---|
| `stage.changed` | New stage key, timestamp, actor | Looks the key up in the published table (§2) and renders accordingly. ⛔ **It does not interpret the key itself.** |
| `condition.added` | Condition, its type (PTA/PTD/PTC/PTF), plain-language text | Adds a needs-list item **in the borrower's language** |
| `condition.cleared` | Condition ID | Removes the item; tells the borrower it was accepted |
| `document.received` | Document ID, status | Updates the needs list |
| `document.rejected` | Document ID, **the reason** | ⭐ A specific, kind re-request — never "conditions outstanding" |
| `disclosure.sent` | Which disclosure, delivery timestamp | Surfaces it for signature |
| `disclosure.signed` | Signer, timestamp | Advances the signing room |
| `lock.confirmed` | Rate, expiry, plan code | Updates the borrower's lock status and the partner's locked/floating view |
| `lock.expiring` | Days remaining | Cadenced warning to borrower **and** loan officer |
| `valuation.completed` | Valuation type, delivery obligation | Triggers the Reg B delivery path |
| `decision.issued` | Approve / Suspend / Deny | See §5.3 — the halt |
| `closing.scheduled` | Date, location, participants | The closing screen |
| ⭐ `automation.halt` | Reason (denied / withdrawn) | ⭐⭐ **Synchronous and confirmed** — see §5.3 |

## 4.2 ⭐⭐ Reconcile to truth, not deltas

**SM-04, and it is the rule most likely to be got wrong by a competent engineer.**

When the POS and the LOS disagree about a loan's state, the answer is **never** to compute the difference and apply it. It is to **re-read the authoritative side and adopt what it says.**

**Why delta application fails here:** events can arrive out of order, be duplicated, or be missed during an outage. A system that applies deltas accumulates drift silently and each individual step looks correct. A system that reconciles to truth **cannot** drift, because it never carries state forward that it did not just verify.

**Concretely:**

- On any ambiguity — an unknown event, an out-of-order sequence, a version mismatch — the POS calls `getLoan()` and **adopts the returned state wholesale.**
- The daily reconciliation sweep (SM-06) does this for **every** open loan and **reports what it healed**, not merely that it ran.

## 4.3 Delivery guarantees

| Requirement | Rule |
|---|---|
| **SM-03** | ⭐ **Signed webhooks with replay protection.** Signature verification, a nonce, and a time window. |
| — | **At-least-once delivery**, with **de-duplication at the consumer.** Exactly-once does not exist across a network; de-duplication is how you get its effect. |
| — | **Out-of-order events reconciled to true state** (§4.2), never applied blindly in arrival order. |
| **SM-05** | ⭐ **Dead-letter queue surfaced to a human.** A failed event that only lands in a log is a lost event. |
| **SM-06** | **Daily reconciliation sweep with drift alerting.** |
| — | **Exponential-backoff retries** with an idempotency key on every call. |

## 4.4 ⭐ The one event that is not fire-and-forget

Every event above is asynchronous **except** the automation halt (SM-10). It is specified in §5.3 and it is the only place in this contract where the LOS **waits for a confirmation** before considering the act complete.

## 4.5 What the POS may derive, and what it may not

| The POS may | The POS may not |
|---|---|
| Render the borrower-facing label from the published table | ⛔ Compute a stage from any combination of fields |
| Translate a condition into plain borrower language | ⛔ Decide whether a condition is satisfied |
| Show that a lock is confirmed and when it expires | ⛔ Decide that a lock is effective |
| Show a fee and what authorised it | ⛔ Decide whether a fee is permitted before ITP — **that gate is enforced on both sides independently** (§5.2) |
| Show that a valuation is available | ⛔ Decide that the Reg B delivery obligation is satisfied |
| Show the partner a milestone | ⛔ Show the partner anything financial — the partner view is a **separate projection**, not a filtered loan |

⭐ **The last row is the one with the sharpest architectural consequence.** The partner-facing view is built from a **separate, smaller object.** Financial fields are not hidden from the agent — **they were never in the payload that was sent.** This is what makes the partner-scoped AI assistant safe against prompt injection: a retriever cannot reach data that is not in its index.

# 5 · Compliance boundaries that cross the seam

⚠️ **These four are the ones where a seam defect is not a bug — it is a violation.** Each is enforced on **both** sides independently, because a control that exists on only one side of a network boundary is a control that fails whenever that boundary does.

## 5.1 ⭐⭐ The TRID clock — split ownership, single deadline

**SM-09.** This is the most nuanced row in the ownership matrix, and it is worth stating three ways so it cannot be misread.

| Responsibility | Owner | Why |
|---|---|---|
| **Detect that the six pieces are complete** | **POS** | The six pieces land there. That is where "an application now exists" becomes knowable. |
| **Timestamp that moment** | **POS** | The instant is the fact; it must be captured where it happens |
| **Carry the timestamp across the seam** | ⭐ **A first-class event**, not a field on a batch | It is the input to a legal deadline |
| **Generate and deliver the disclosures** | **LOS** | Through the document vendor |
| ⭐ **Own the three-business-day deadline** | **LOS** | — |
| **Log the whole sequence** | **Both** | Neither trail is a subset of the other |

> ⭐⭐ **Whoever owns delivery owns the deadline.** That sentence is the entire point of the split.

**The six pieces**, restated so neither team has to look them up: name · income · Social Security number · property address · estimated property value · loan amount sought.

⭐ **The pre-trigger warning (K20)** fires on the **POS** side: the system knows when the borrower is one field away from creating a legal application and tells the loan officer, **so the clock is started deliberately rather than by accident.**

⚠️ **Business-day arithmetic is defined once.** TRID contains two different definitions of "business day." They are codified — in one place — and **both halves use the same implementation.** Two independent implementations of a date rule will disagree, and the disagreement will be invisible until an examination.

## 5.2 ⭐⭐ The pre-ITP fee gate — enforced twice, on purpose

**Both halves enforce this independently.** It is the one duplication this document explicitly *requires*, and the reason is that a fee can be created on either side.

> **Before Intent to Proceed, exactly one fee type may exist: a bona fide, reasonable credit-report fee.** No appraisal fee, no application fee, no lock fee, no processing deposit. §1026.19(e)(2)(i).

> **Documents may be invited but never required before Intent to Proceed.** The needs list may appear; it may not block. §1026.19(e)(2)(iii).

| Side | Enforcement |
|---|---|
| **POS** | K21 — the fee model **rejects** a non-credit fee before ITP. K22 — the needs-list model **cannot set** `required` before ITP. |
| **LOS** | CO-07 — the same rejection, at the same data layer, on the fee sheet. |

⭐ **Intent to Proceed is its own timestamped act** (K24, CO-08). Never pre-checked. Never inferred from silence. **Never riding on the e-signature of the Loan Estimate** — a borrower may receive an LE and choose not to proceed, and that must be a representable outcome.

## 5.3 ⭐⭐ The adverse-action automation halt — the only synchronous event

**SM-10.** This is the single most important line in this document after the stage table.

When a file is **denied** or **withdrawn**, the POS's borrower automation must stop — and ⭐ **the LOS must know that it stopped, synchronously and confirmed.**

⛔ **An asynchronous "we published the event" is not sufficient.** The failure mode is a cheerful marketing message reaching a borrower who was denied an hour ago. **That is a fair-lending problem, not a bug.**

**The sequence:**

```
LOS: decision.issued (denied | withdrawn)
     │
     ├──► POS: automation.halt  ──────────── SYNCHRONOUS
     │         · stop every campaign, nudge, milestone card, partner update
     │         · stop the listing-agent stream
     │         · stop the assistant from generating anything new
     │         └──► ACK  ("halted", timestamp, what was stopped)
     │
     ▼
LOS: records the confirmed halt on the file
     │
     └──► only now does the milestone change complete
```

⭐ **If the acknowledgement does not arrive, the LOS raises it as an incident** — a task on the loan officer and an alert to compliance. It does not silently proceed and it does not retry indefinitely in the background while a borrower receives messages.

**Related rules that must not be missed:**

- ⛔ **Withdrawn ≠ denied** (CO-20, M9). Different obligations, different notices. A borrower who walked away was not declined.
- The published stage table's `compliance-controlled` rule (§2.6) supplies the **only** words the borrower sees, and they do not explain the reason — **the Adverse Action notice does that.**
- ⭐ The **counteroffer clock** (K25, CO-18) means a file can *become* an adverse action 30 days after a counteroffer nobody accepted. **That transition fires this same halt**, and it is the one everybody forgets because the file still feels alive.

## 5.4 The CD three-day scheduling block

**CO-10.** The Closing Disclosure must be received by the borrower **at least three business days before consummation.**

- The **LOS** owns the CD and its delivery timestamp.
- The **POS** owns the borrower's closing scheduler (E7).
- ⭐ **The block is enforced on both sides:** the POS cannot offer a closing slot that would violate the window, and the LOS refuses to advance the milestone if one somehow were.

⚠️ Electronic delivery carries **received-timestamp presumption rules**. The presumption used is defined once, in the LOS, and the POS reads the resulting permissible date range — it does not compute its own.

## 5.5 Consent and revocation across the seam

| Consent | Lives on | Crosses as |
|---|---|---|
| **Contact — SMS, email** | The **Person**, POS side | ⭐ Revocation propagates to POS, CRM, assistant and partner stream **within seconds** (K12) |
| **E-SIGN** | The **Loan**, POS side | Pushed to the LOS with exact text and version |
| **Credit authorisation** | The **Loan**, POS side | Pushed; ⛔ **reuse across concurrently open files is counsel's decision, not a developer's inference** |
| **Partner update stream** | Its own record | ⭐ **Separately revocable**, independent of the borrower's other consents (K16) |
| **Listing-agent disclosure** | Its own record | Its own checkbox, its own revocation, scoped to one transaction, expiring at funding or termination (I24) |

⭐ **Every consent record carries the exact text shown, its version, the timestamp, the IP and the device — never a boolean.** The test is an examiner asking *"prove this borrower consented to this text on this date"*, and the answer must be **one query**, on either side of the seam.

---

# 6 · ⚠️ Conflicts to resolve before code

⭐ **These are named, owned and open.** They are in this document rather than in either half's because they cannot be resolved by one team — resolving them unilaterally is precisely how the two halves end up disagreeing at runtime.

## 6.1 ⭐⭐ Conflict 1 — who sends disclosures

**Three sources give three answers.**

| Source | Position |
|---|---|
| **The Director of Operations**, verbatim | *"Currently the LO sends initial disclosures and revised LEs while the ICD is sent by the Closer — my team/processor sends the Revised LE if needed and closer sends ICD. **ALL are sent in the LOS — not the POS.**"* |
| **The prior NachoLOS blueprint** | The loan officer sends initial disclosures from the POS / LO Portal |
| **The POS specification** | Assumed a POS-side delivery experience, with the TRID trigger timestamped there |

**Recommendation, for the decision-maker to accept or reject: follow the Director of Operations.** She is describing what a licensed shop does today under an existing compliance regime, and ⭐ **disclosure delivery is not a place to introduce novelty for architectural tidiness.**

**The provisional split, already written into §5.1** — and it satisfies all three concerns: the **POS detects and timestamps** the trigger; the **LOS generates, delivers and owns the deadline**; **both log**; and the timestamp crosses as a first-class event.

**Decision owner:** CLEAR. **Blocks:** the disclosure screens on both sides, and SM-09.

## 6.2 Conflict 2 — the long-term owner of the disclosure audit trail

Today the document vendor holds the authoritative disclosure trail. The LOS reconciles to it (AU-03).

**The open question:** does that remain true, or does the LOS become the system of record with the vendor as a subordinate source?

**Why it matters now rather than later:** it determines whether the one-action examination pull (CO-25) can be satisfied from CLEAR's own systems, or whether it will always require a vendor export. That is an operational dependency with commercial implications under §1.6 licensing.

**Decision owner:** CLEAR, with compliance counsel. **Blocks:** nothing immediately — AU-03 (reconcile to the vendor's trail) is correct under either answer.

## 6.3 Conflict 3 — the brokered channel

⚠️ **Absent from the prior blueprint entirely**, leaving roughly **one loan in eight** with no home in the system.

The seam question specifically: a brokered loan is **disclosed and handed off.** That means the POS→LOS push, the event subscription and the borrower's status experience all terminate somewhere different, and **nobody has specified where.**

**Decision owner:** CLEAR. **Blocks:** PR-09, and the brokered path through this entire document.

## 6.4 Conflict 4 — hard-credit-pull reuse across concurrent files

A borrower may run more than one live application at once — refinancing two rentals, or a purchase alongside a refinance (B22).

**A single hard pull may be reusable across files opened close together. The authorisation record is per-loan regardless.**

⛔ **The reuse window is counsel's call, and it must not be inferred by a developer.**

**Decision owner:** compliance counsel. **Blocks:** B25's implementation detail only — the per-loan authorisation record is settled and can be built now.

### 6.5 Conflict 5 — the review rule that keeps §1 true

⭐ **This is not an open question; it is a standing process commitment**, recorded here because it is what prevents §1 from decaying.

> **Any pull request that computes a stage, a status, a tolerance bucket, a business-day count or a compliance deadline in the POS must be reviewed against §1 and §2 before merge.**

The rule exists because the failure it prevents is invisible in code review otherwise. A function called `deriveStatus()` in the POS looks perfectly reasonable, passes its tests, and is a violation of the single most important contract in the product.

**Suggested enforcement:** a lint rule or a CODEOWNERS entry on the relevant modules, so this is mechanical rather than a thing somebody has to remember.

### 6.6 Summary — what is blocked on whom

| # | Conflict | Owner | Blocks |
|---|---|---|---|
| 1 | ⭐⭐ Who sends disclosures | CLEAR | Disclosure screens both sides, SM-09 |
| 2 | Disclosure audit-trail ownership | CLEAR + counsel | Nothing immediately |
| 3 | ⭐ The brokered channel | CLEAR | PR-09 and its whole path |
| 4 | Hard-pull reuse window | Counsel | B25 detail only |
| 5 | The seam review rule | Both engineering teams | ⭐ Nothing — **adopt it now** |

---

# 7 · Change control for the seam

### 7.1 ⭐⭐ The rule that makes the seam different

The POS and LOS documents version independently. **This one does not.**

> ⭐⭐ **A change to anything in this document forces at least a MINOR version bump in BOTH `POS_version` and `LOS_version`.**

Because by definition, a change to the seam affects both halves. A seam change that appears in only one half's changelog is exactly how one team ships against a contract the other team has not read.

### 7.2 Version rules for this document

| Bump | When | Also requires |
|---|---|---|
| **PATCH** — 1.0.**x** | Wording clarified. **No contract changed.** | Nothing |
| **MINOR** — 1.**x**.0 | A new event type; a new field on an existing payload; a new row in the ownership matrix | MINOR bump on both POS and LOS |
| **MAJOR** — **x**.0.0 | ⭐ An owner moves in §1; a field is renamed or removed; an event's meaning changes; the stage table's rules change | MINOR bump on both POS and LOS **and** a joint review before merge |

**The stage table carries its own version** (§2.7), served alongside it. The two version numbers are independent: this document can go to 1.1.0 without the stage table moving from 11.4.1.

## 7.3 Backward compatibility, and the direction to fail in

| Situation | Required behaviour |
|---|---|
| The POS receives a **stage key it does not recognise** | ⭐ **Go silent on that row.** Hold the tracker at the last known-visible stage. ⛔ Do not guess a label. |
| The POS receives a **stage-map version it does not recognise** | Adopt the table it was served, and **treat unmapped keys as** `not-surfaced` |
| The LOS receives a **push field it does not recognise** | Store it, ignore it, and log it. ⛔ Never reject the whole payload for one unknown field. |
| Either half receives an **event type it does not recognise** | Acknowledge, log, and trigger a `getLoan()` reconciliation (§4.2) |
| A **required** field is missing | **Reject with a named reason.** This is the one case where failing loudly is correct — a missing required field is a defect, not a version skew. |

⭐ **The pattern: unknown things degrade to silence; missing required things fail loudly.** Silence is safe for a borrower-facing label. Silence is dangerous for a compliance input.

## 7.4 The joint review

⭐ A **MAJOR** change to this document requires **both teams in the same review** before merge. Not a notification, not a changelog entry either team can miss — a review.

This is a small amount of process guarding the failure this entire document exists to prevent.

## 7.5 Changelog

| Version | Date | What changed |
|---|---|---|
| **1.0.0** | 2026-09-01 | First delivery. The ownership matrix, the published stage table at version 11.4.1, the push and event contracts, the four compliance boundaries that cross the seam, and the five open conflicts — named rather than resolved unilaterally. |

## 7.6 ⭐ The checklist before either half ships anything that touches the seam

1. Does this change a row in **§1 (ownership)**? → MAJOR, joint review.
2. Does this compute a stage, status, tolerance bucket, business-day count or compliance deadline **in the POS**? → ⛔ **Stop.** See §6.5.
3. Does this add or change an **event** (§4)? → MINOR here, MINOR both halves.
4. Does this touch the **TRID clock**, the **pre-ITP fee gate**, the **automation halt**, or the **CD three-day block** (§5)? → ⭐ Both halves must be re-verified, together, with the dedicated test suites ( `POS` MAINT-04, `LOS` MAINT-03).
5. Does this rename a field on the wire — including `status` or `suspension.{reason,owner,since}`? → ⭐ MAJOR, joint review.
6. Is a new stage key involved? → Add the row to the **published table** (§2), bump its version, and let both halves read it. ⛔ **Never hardcode the label in a screen.**

---

*NACHO INTEGRATION PRD · version 1.0.0 · issued 2026-09-01 · generated by `node assemble.mjs` — do not hand-edit.*