

OC TRAVEL

BUILDABLE SPEC DOSSIER



Spiti

BY OC TRAVEL

The buildable specification — a portal built with zero clarifying questions.

A specification a senior team builds from with *zero clarifying questions* — the whole product, decided.

This is the second volume in the slipcase. The first sets the mark, the palette, the type, the voice. This one is the machine, and it is one coherent whole: **ten roles** and their journeys, one golden path and every edge flow, the permissions wall drawn field by field, the data model, every state, every screen, the integrations, the data-transfer story, and the non-functional frame that a portal for HNW international clientele cannot treat as an appendix.

The org chain runs top to bottom (Client Director, Zone Manager, Store Manager), and the two most senior brand roles are the product's sharpest walls: real power over people, and no reach into a single guest. Batch and bulk operations, the property catalog, and the migration all obey one rule: *how many records move never changes which records, or which fields, a role may touch.*

Nothing here is re-decided. The permissions matrix and the data model present the architecture *verbatim in substance*, organized for reading; where this dossier and the architecture files differ, the architecture files win. The one design tension governs every page: *invisible luxury on the guest's face, strict need-to-know underneath*. Discretion is the luxury, and the wall is drawn as a feature — never an omission.

THE TRUTH STANDARD, HELD ON EVERY PAGE

Integration providers named are **real** (FlightAware, Cirium, Postmark/SendGrid, Twilio, Google Maps, PassKit, Auth.js). All in-app flight behaviour is **simulated and labelled simulated**; every count, currency, and threshold is **illustrative and marked**. The worked example, LV Cruise 2027, Paris, is illustrative; no real maison logo or exact palette is claimed.

IN THIS VOLUME

What the last build taught	§1
Roles & permissions — the wall	§2
The capability matrix · ten roles	§2.2
The vendor DTOs	§2.3
PII minimization & masking	§2.4
The adversarial checklist · sixteen reads	§2.5
The data model & the org / catalog spine	§3
States & two lifecycles	§4
The golden path & the delay ripple	Flow 0 · 4
Edge flows & the eight new flows	F1–F6
Ten roles, ten journeys	Maps
The annotated screens	A–J · CD–MSG
The data-transfer story	§DT
Integrations & the non-functional frame	§5 · 6
What "done" means	§7

Two shells, chosen by role. A *spacious* register for guest and manager surfaces, an *efficient* register for OC operations, both from the same design-tokens.css.

RECLAIMING THE NAME

What the last build taught.

The engine worked; the experience did not. Six lessons, paid for in operation — every one carried forward into the good version of the name.

Spiti is the client's own in-market tool name, owned and kept. The prior version was functional but dated; its engine ran, its experience didn't. These lessons were paid for in that build's operation and are stated here not as critique but as the **design principles the new portal honours**. The through-line: the client owns the source, the secrets, the domain, and the data, so no single external account can lock OC Travel or its guests out of their own product.

THEN · A SPREADSHEET

The room block is a first-class object.

A RoomBlock per event, carrying property, room types, rooms_committed, live rooms_used, cutoff, attrition, and rate code, with a designed intelligence dashboard, not a sheet.

→ data model §3 · screens B7

THEN · A FREE-TEXT NOTE

A guest is assigned a specific room, for specific dates, inside the block.

HotelBooking binds an EventGuest to a block with room type, dates, rate, and stay requests — booked *from* the block, decrementing its usage. Precise, not a note.

→ data model §3 · screens C3

THEN · SCATTERED ACROSS EMAIL + TEXT

Messaging lives inside the portal, two-laned.

One thread per guest with a hard partition between the guest_facing lane and OC internal notes — in the product, scoped, audited.

→ data model §3 · screens C8

THEN · NO ACTIVE SUPPORT

The client owns the codebase.

The delivered source is the client's outright, in a repository they control — no vendor lock-in on the code, plain-language handover included.

→ non-functional §6.5

THEN · A PLAINTEXT SMTP PASSWORD IN A COMMITTED .ENV

Secrets never live in the source.

No credential of any kind is ever committed. Secrets live in client-controlled config, rotatable; an .env.example documents, never contains, and RUN.md walks first-time setup.

→ non-functional §6.3

THEN · A DEPARTING AGENCY'S MAILBOX

Auth & email run on a domain OC Travel owns.

Transactional email and sign-in bind to an owned domain and account. Magic-link auth is chosen partly *because* it makes Spiti resilient to a single-mailbox outage.

→ non-functional §6.4

A note on the two Spitis. "The previous build" is the client's earlier, in-market tool of the same name: *old Spiti*. Everywhere else, "Spiti" is the product specified here, *new Spiti*, the good version of a name the house is reclaiming. The name is kept; the fragility is left behind.

The wall is the product.

On the guest's face, nothing shows. Underneath, every read is filtered by tenant and scope, and every field a role sees is a field a rule granted. This is where the design tension becomes engineering: **invisible luxury above, strict need-to-know below.**

Enforcement is **server-side, field-level, deny-by-default** — never a UI hide over a permissive API. A denied field does not render because it never crosses the wire. The four principles at right are the whole doctrine; the matrix, the DTOs, the masking table, and the adversarial checklist that follow are that doctrine, drawn.

PRESENTED FROM `roles-permissions.md` · authoritative there

The only cross-tenant surface in the entire product is the OC Admin portfolio home. Every other role, concierge, approver, store manager, and all vendors alike, is walled to a single brand, and most to a single scope within it.

THE FOUR ENFORCEMENT PRINCIPLES

- 1 **Deny-by-default.** No role sees any field unless a rule grants it. New fields are invisible to every role until explicitly granted. Nothing reads a whole record except OC Admin and Concierge, on their owned scope.
- 2 **Tenant + scope on every query.** Every read is filtered by `brand_id` and the role's scope key (`store_id`, `assignment set`, `owned-guest set`), *in the query itself*. A response never contains a row the role may not see.
- 3 **Field-level projection.** Vendors receive a projected view object (`HotelGuestView`, `TransportGuestView`, `ApproverGuestView`) with only their granted fields. The full Guest record never reaches a vendor; a denied field is absent from their API surface.
- 4 **Time-boxed access is a gate, not a hide.** Vendor sessions carry an `access_window`; outside it, grants evaluate empty server-side. Expiry is enforced at the query layer — an expired login returns nothing, not a stale cache.

DEFENSE IN DEPTH — TWO WALLS

The projection is one wall. Postgres **Row-Level Security** is the second, independent one: even an app-code bug cannot leak a cross-tenant row if RLS holds. The adversarial checklist (§2.5) ships as a passing test suite, not a hope.

Twenty-one capabilities across ten roles. Read down a column for a role's whole world; read across a row for who touches one capability. The eye lands first on how much is **denied**, because the wall is mostly what a role *cannot* do. The two **brand-side roles added since the first build**, **Client Director** and **Zone Manager**, read as a pair of walls: each is nearly all —, lighting up *only* where it manages people or judges quality, never where it touches a guest.

CAPABILITY	OC ADMIN	OC CONCIERGE	CLIENT DIR	ZONE MGR	BRAND APPROVER	STORE MGR	HOTEL	TRANSPORT	RESTAURANT	GUEST*
Create / config events, blocks, catalog, chains	✓	—	—	—	—	—	—	—	—	—
Manage brands / tenants	✓	—	—	—	—	—	—	—	—	—
Appoint / remove Brand Approvers	✓	—	own-brand	—	—	—	—	—	—	—
Appoint / remove Zone Managers	✓	—	own-brand	—	—	—	—	—	—	—
Appoint / remove Store Managers	✓	—	—	own-zone	—	—	—	—	—	—
Nominate guests	✓	✓	—	—	—	own	—	—	—	—
Approve / decline / needs-info	✓	—	—	—	own chain	—	—	—	—	—
Batch approve / decline candidates	✓	—	—	—	own chain	—	—	—	—	—
View nominations (judge-only proj.)	✓	own	—	own-zone	own-level	own-store	—	—	—	—
Enter / confirm flight info	✓	✓	—	—	—	own	—	—	—	self†
Book hotel / dining / transport / gifting	✓	own	—	—	—	—	—	—	—	—
Excel import (scoped)	✓	—	—	—	—	own store	—	—	—	—
Excel export (own projection)	✓	own	—	zone judge	—	own store	—	—	—	—
View full guest CRM / PII	✓	own	—	—	judge-only	own ltd	stay-only	pickup-only	dining-only	self
Message: brand ↔ OC lane	✓	✓	R/W	R own-zone	—	—	—	—	—	—
Message: hotel ↔ OC lane	✓	✓	—	—	—	—	own-event	—	—	—
Message guest (guest-facing)	✓	✓	—	—	—	own	—	—	—	✓
Manage own brand's people surface	✓	—	own-brand	own-zone SMS	—	—	—	—	—	—
View global audit log	✓	—	—	—	—	—	—	—	—	—

✓ full **own** own-scope only **view** read-only **ltd** limited / projected DTO — denied **self** own record only

* Self-service mode only. † Self-service guests enter their own flight. **The brass columns are walls:** lit only where each manages people or judges, never a PII row.

A vendor never receives the Guest record and filters it — they receive a *projection* carrying only granted fields. A denied field is not hidden; it is **absent from the projection**, so the screen literally cannot render it. Each card is one vendor's entire surface.

Hotel HotelGuestView TIME-BOXED · OWN-EVENT · OPENS T-14D ♦ GRANTED display_name* party_size room_type check_in / out rate_code expected_arrival (derived) arrival_terminal special_requests_stay ✗ DENIED — ABSENT FROM PROJECTION passport / ID full DOB flight PNR / full record home address purchase history VIP summary transport gifting budget itinerary beyond own booking	Transport TransportGuestView TIME-BOXED · OWN-ASSIGNMENT · OPENS T-7D ♦ GRANTED display_name* party_size pickup / drop_point pickup_time (derived) flight_number only guest_phone (masked → in-window) accessibility_flag (mobility) route_notes ✗ DENIED — ABSENT FROM PROJECTION passport / ID DOB PNR / full flight hotel room number purchase history VIP summary dietary / medical gifting full itinerary budget	Approver ApproverGuestView OWN-BRAND · OWN-LEVEL · JUDGE-ONLY ♦ GRANTED display_name VIP tier store relationship purchase_summary (justification) store manager's note prior-events count approval chain state + audit slice ✗ DENIED — ADVERSARIAL-CRITICAL passport / ID full DOB flight PNR / times hotel property & room transport home address phone / email dietary / medical gifting itinerary budget
--	--	---

Other vendors — type-keyed

RestaurantView · ActivityView · SecurityView

A GRANT TO ONE TYPE IS NEVER A GRANT TO ANOTHER

Restaurant. Name, party_size, reservation_time, dietary_notes. *Not* security, flights, hotel, transport, purchase.

Activity. Name, party_size, activity_time, accessibility_flag (mobility). *Not* dietary, security, flights, purchase.

Security. real_name **un-masked** (the one vendor that unmask), protection_flags, movement times. *Not* dietary, purchase, gifting.

The approver decides *who comes* — not how they travel.

* display_name renders as "Guest, LV VIP 2" when the security flag is set. The same guest's data reaches different vendors *differently*: the whole doctrine, in one record.

§ 2.4 — PII MINIMIZATION & MASKING THE NEED-TO-KNOW WALLS, AT A GLANCE

This is the heart of "discretion is the luxury." Fourteen PII fields down; twelve roles across. The pattern to read is the diagonal of restraint: full sight collapses to **limited**, to **masked**, to **nothing**, the further a role sits from the guest's care. The **two brand-side roles added since the first build** make the point loudest: the **Client Director** column is — on *every* PII row, and the **Zone Manager** lights up on exactly one, the judge-only purchase summary, identical to the approver's single grant. That wall of dashes is the visual proof of a role with real power over people and zero reach into guests.

PII FIELD	OC	CONC.	CLIENT DIR	ZONE MGR	APPROVER	STORE MGR	HOTEL	TRANSP.	REST.	ACTIV.	SEC.	GUEST
Real name	✓	own	—	jdg	jdg	own	msk	msk	msk	msk	✓	self
Passport / ID no.	✓	own	—	—	—	ent	—	—	—	—	—	self
Full DOB	✓	own	—	—	—	—	—	—	—	—	—	self
Home address	✓	own	—	—	—	own	—	—	—	—	—	self
Phone / email	✓	own	—	—	—	own	—	win	—	—	win	self
Flight PNR / full record	✓	own	—	—	—	ent	—	—	—	—	—	self
Flight timing (arrival / number)	✓	own	—	—	—	own	sub	n+t	—	—	mv	self
Purchase history / VIP summary	✓	own	—	sum	sum	auth	—	—	—	—	—	—
Dietary detail	✓	own	—	—	—	—	room	—	✓	—	—	self
Accessibility (fact)	✓	own	—	—	—	—	✓	mob	—	✓	✓	self
Accessibility (medical detail)	✓	own	—	—	—	—	—	—	—	—	—	self
Security / protection flag	✓	own	—	—	—	—	trg	trg	trg	trg	✓	—
Hotel room number	✓	own	—	—	—	—	✓	drop	—	—	—	self
Gifting record	✓	own	—	—	—	—	dlv	—	—	—	—	itn
Org roster / people (not a guest field)	✓	—	brand	zone	—	—	—	—	—	—	—	—

✓ full sight own own scope limited / windowed subset msk masked / trigger — denied self own record

Codes in limited cells: jdg judge-only projection · sum purchase summary · win in-window only · sub timing subset · n+t number+time · mv movement times · mob mobility fact · ent own-entered · trg mask trigger · drop drop address only. **Masked** renders as "Guest, LV VIP n" when the security flag is set. The bottom row is the one place the two new columns read anything but —, and it is not a PII field at all: it is the org chart. The authority is over *people*, never over guests.

Sixteen questions the wall must answer *no*.

The checklist is not documentation of intent. It is a **passing test suite before v1 ships**. Each line is an adversarial read: an attempt to reach a field a role should never touch. Each must return *no*, and each names the mechanism that makes it so. Nine are below; the seven the two new roles must survive follow overleaf, with the data-operation reads.

→ BECOMES the build's security test suite (§7.2)

The super-user is the last line, and the answer is still *no*: OC Admin is all-seeing on scope but never exempt from the audit. Every access an admin makes writes an AuditEntry. Power is logged, not privileged.

<i>Hotel reaches a passport?</i>	No
not on HotelGuestView — projection + deny-by-default	
<i>Store manager sees another store's nominations?</i>	No
store_id scope on every query	
<i>Driver sees purchase history?</i>	No
not on TransportGuestView	
<i>Approver sees where a guest stays, or their PNR?</i>	No
judge-only DTO; logistics fields absent	
<i>Restaurant sees accessibility medical or security flag?</i>	No
dietary + reservation only; security masks the name, discloses nothing	
<i>Expired vendor pulls last month's rooming list from cache?</i>	No
window enforced at query layer; empty, not stale	
<i>Guest reads another guest's itinerary?</i>	No
scope self, always	
<i>Concierge reads a guest they don't own?</i>	No
owned_guest scope; no portfolio view for concierges	
<i>Super-user exempt from audit?</i>	No
OC Admin actions all write AuditEntries	

Two new roles are two new ways the model had to survive an adversarial read, and both hold. The rule the whole set defends: **widening scope to a brand or a zone never widens field access**. A Client Director sees the whole house's people and not one guest; a Zone Manager judges a whole zone with the approver's exact fields and not one logistic. Below the role reads, the data operations (batch, import, export, message lanes) face the same test: a bulk action never changes *which* records or *which* fields the actor may touch.

THE CLIENT DIRECTOR — POWER OVER PEOPLE, NONE OVER GUESTS

10 *Reach any single guest PII or logistics field?* No

BrandOrgView never joins a guest table; no route returns a guest field

11 *See nomination content or approval decisions?* No

a guest count per event only; verdicts and content are off the DTO

12 *Appoint into another brand, or escalate to OC / vendor roles?* No

writes checked target.brand_id == actor.brand_id; only {approver, zone_mgr}

THE ZONE MANAGER — WIDENED ONE LEVEL, NEVER DEEPER

13 *See another zone's nominations, stores, or managers?* No

every zone query binds WHERE brand_id = ? AND zone_id = ? to the session

14 *Reach passport / PNR / room / transport for a candidate in her own zone?* No

ZoneNominationView is the judge-only projection; logistics never joined

15 *Approve/decline, or appoint a non-store-manager role?* No

not a chain actor; only store_manager, only in her zone (store.zone_id check)

16 *Re-point her own scope via a request parameter?* No

scope_zone_id and brand_id read from the session record on every query, never the request

THE DATA OPERATIONS — BULK NEVER WIDENS

Does batch approval widen an approver's reach? No

batch loops the single-card write; scope + per-candidate lifecycle identical, each audited

Can an Excel export carry a field absent from the caller's screen? No

export serializes the caller's own scoped projection; passport/DOB/PNR/flags are not columns

Can an Excel import move a field the caller cannot write, or land a row out of scope? No

dangerous fields are not import columns; store_id forced to scope; out-of-scope row rejected

Can a hotel read the brand's lane, or a brand read the hotel's lane? No

each lane participant-gated by counterparty; a hotel and a brand never share a thread

Can a Client Director or Zone Manager read PII by opening a brand↔ OC thread? No

a message body is prose; opening a thread never joins a guest record

THE ONE-LINE GUARANTEE

Batch and bulk change how *many* records move in one action; they never change *which* records or *which* fields the actor may touch. The scope and the DTO are the same as the single-record path. An adversarial read of an Excel export is an adversarial read of the caller's screen.

The data model — the spine.

Two decisions define it: the guest CRM persists across events, and flight times are the single source of truth — transport ETA and hotel arrival are derived, never independently written.

PERSISTENCE

One durable person, many events. Service compounds because the profile is the memory.

SINGLE WRITE

One flight write fans out to pickup, arrival, and itinerary. Nobody re-types.

TENANCY

Brand → Event → EventGuest.
No query starts below
`brand_id`.

```

Brand (tenant)
├─ User   role · scope: store_id / owned / vendor-org
├─ Vendor type · org · access_window
└─ Event
    ├── branding_assets · mode_default · approval_rules
    ├── RoomBlock  property · committed · cutoff · rate
    ├── Nomination
    │   └─ NominationCandidate → EventGuest
    └─ EventGuest
        ├── lifecycle_state · vip_tier · mode
        ├── Approval  level · decision → audited
        ├── Flight   actual_arrival ◀ SOURCE OF TRUTH
        ├── HotelBooking  expected_arrival ◀ DERIVED
        ├── Reservation  dining / activity
        ├── TransportAssignment pickup_time ◀ DERIVED
        ├── GiftingRecord · ItineraryItem (some ◀ DERIVED)
        ├── Task · Message  lane: guest_facing | internal
        ├── CostLine  amount · payer
        └─ GuestProfile ← PERSISTS ACROSS EVENTS

GuestProfile (durable person)
  preferences · allergies · accessibility · sizing
  languages · consent · vip_tier · past_events[]

AuditEntry  actor · action · target · ts — spans ALL

```

Read the tree as ownership: everything hangs off a Brand tenant, and the EventGuest is the join where a durable person meets one event and picks up a lifecycle. The three coloured marks are the load-bearing rules — the source of truth, the derived fields, and the profile that persists.

THE THREE RELATIONSHIPS THAT CARRY THE PRODUCT

GuestProfile 1–N EventGuest

The CRM persistence. One person, many events; the profile is the memory that makes service compound rather than restart. GDPR erasure cascades or anonymizes EventGuests per retention rules.

Flight 1–(derives)–
pickup_time
+ expected_arrival +
itinerary times

The single-write propagation. Derived fields have no independent write path. A manual override writes back to Flight (source: manual) so it still propagates from one source. This is the architectural spine of "the car is waiting even when the flight is late."

Brand 1–N Event 1–N
EventGuest

The tenancy + scope chain every RBAC query walks. No query starts below brand_id — the wall from §2 is anchored in the shape of the model itself.

PRESENTED FROM DATA-MODEL.MD

Authoritative there, organized here. Encryption at rest covers passport, PNR, and DOB; the derived fields are computed, never stored as an independent editable value.

Brand

tenant

name · branding_assets{logo, palette, type_lockup, archetype} · approval_rules{l1-3} · mode_default · retention_policy · data_residency_region

Multi-tenancy backbone; no record is tenant-less.

User

brand_id? · org_type (oc/brand/vendor) · role · scope{store_id? / owned_guest_ids? / vendor_org_id?} · auth_method · locale

Scope is the enforcement key on every query.

Vendor

brand_id · type · org_name · contacts[] · access_window{opens, closes} · field_grants (from type)

access_window gates all vendor reads.

Event

brand_id · name · city · venue · start/end · mode · co_branding · approval_chain_instance · capacity · lifecycle_state · template_id?

mode drives the guest-experience toggle.

RoomBlock

event_id · property_name · room_types[] · rooms_committed · rooms_used (derived) · cutoff_date · attrition_threshold · rate_code

A dropped guest's booking releases back here.

Nomination / Candidate

nom: event_id · store_mgr · attachments[] · cand: guest_profile_id · justification · status

Each candidate independent → partial approval native.

GuestProfile

persists

legal_name · display_name · dob? · passport_no? (enc) · home_address? · phone · email · languages[] · vip_tier_default · preferences · allergies[] · dietary[] · accessibility{facts[], medical(restricted)} · sizing · security{protection_flag, name_confidential} · consent{captured_at, scope, gdpr_basis} · past_events[] · notes_internal

The durable person; encrypted at rest for passport/DOB.

EventGuest

event_id · guest_profile_id · lifecycle_state · vip_tier_at_event · mode · waitlist_rank? · dropped_reason? · special_needs_snapshot

A durable person + one event; carries the state machine.

Flight

source

event_guest_id · direction · airline · flight_number · pnr? (enc) · scheduled_arrival · **actual_arrival** (written once) · terminal · status · last_synced_at · source (api/manual)

actual_arrival is the single write that propagates.

HotelBooking

event_guest_id · room_block_id · property_name · room_type · check_in/out · rate_code · requests_stay[] · **expected_arrival** (derived) · late_arrival_flag (derived)

expected_arrival computed, not typed.

TransportAssignment

event_guest_id · vendor_id · pickup_point · drop_point · **pickup_time** (derived = actual_arrival + buffer) · scheduled_flight_id · vehicle_type (derived incl. accessibility) · driver_slot · route_notes · guest_phone (masked until window)

Recomputes on any flight change.

Reservation · Gifting · Itinerary

Reservation: type · vendor · venue · time · party_size · notes · Gifting: item · sizing_ref · delivery_target · status · Itinerary: order · type · title · time (some derived) · location · guest_visible_copy

A delay reflows downstream itinerary items without re-typing.

Message

event_guest_id · thread_id · author · **lane** (guest_facing/internal) · channel · body · sent_at

Hard partition on lane; channel drives notification fan-out.

Task · CostLine · AuditEntry

Task: label · owner · status · due · CostLine: category · amount (illus.) · payer (brand/guest) · status · Audit: actor · action · target · before?/after? · ts · ip?

AuditEntry is immutable append; spans all entities.

```
Brand (tenant)
├─< Zone id · brand_id · name          ◀ new
│   └─< Store id · zone_id · name      ◀ was a flat string
│       └─< EventGuest zone_id ◀ denorm
├─< Property catalog                  ◀ new
│   category · country · city · neighborhood
│   └─< RoomBlock / HotelBooking property_id ◀ FK
└─< Event
    └─ capacity ◀ the guest CAP
    └─ nomination_deadline ◀ new cutoff

User + scope_zone_id · active          ◀ new
Message lane + counterparty_vendor_id
        + counterparty_brand_id        ◀ lanes extended
ImportJob actor · row_count · error_report ◀ new
```

Every addition is additive: no Chapter One relationship is broken, and the two load-bearing decisions (CRM persistence, single-write flight propagation) are untouched. The chain that carries the two new brand roles is Brand → Zone → Store → EventGuest. No query starts below brand_id, and the zone/store layer never grants a logistics field: it only narrows *which* rows a judge-only projection returns.

Zone

new

id · brand_id · name

A first-class geographic group of stores, so “appoint a Zone Manager to a zone” is queryable, not a string convention. A Zone Manager’s scope_zone_id points here.

Store

promoted

id · brand_id · zone_id · name · active

Was a bare string ('lv-dallas'); now a row belonging to exactly one zone. The id keeps the string value, so existing rows resolve unchanged (migration §3.5).

Property

new

id · brand_id? · name · category · country · city · neighborhood · address · active

The hotel catalog, replacing the free-text hotel name. Filterable by the client’s four filters; reusable across events and brands. A null brand_id is a shared house-catalog row.

New fields, in place

events.capacity the guest cap ·
events.nomination_deadline the cutoff · **event_guests.zone_id** denorm from store ·
users.scope_zone_id · **users.active** · **room_blocks.property_id** / **hotel_bookings.property_id**

The cap drives the waitlist (Flow F3a); the deadline closes the nominate route server-side (F3b). property_name is retained as a label/fallback during migration.

Message **lanes+** & ImportJob

messages.lane + hotel_oc / brand_oc · counterparty_vendor_id · counterparty_brand_id – import_jobs: actor · event · row_count · committed_count · error_report · status

A lane is walled to its two parties AND to the specific hotel or brand. The ImportJob keeps the per-row error report retrievable.

BrandOrgView(brandId)

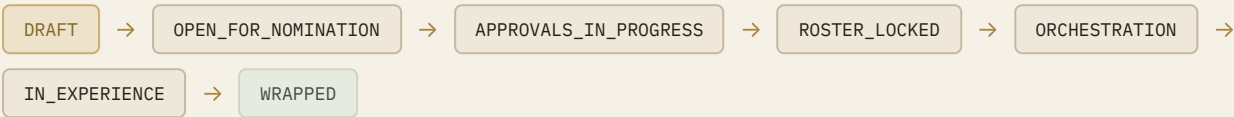
The Client Director’s DTO. Returns the brand’s people roster, the zone/store map, and event *counts*. It never queries guest_profiles or any logistics table; a guest field has no route that reaches it.

ZoneNominationView(brandId, zoneId)

The Zone Manager’s DTO. The ApproverGuestView projection plus store attribution, over event_guests WHERE brand_id = ? AND zone_id = ?. Logistics never joined.

Two state machines run in parallel. The *event* moves through setup phases; each *guest* moves through their own lifecycle independently. That independence is precisely why partial approval and per-guest edge cases are first-class states, not exceptions — an event in `ORCHESTRATION` can hold one guest in `NEEDS_INFO` and another on the `WAITLIST` at the same moment.

THE EVENT LIFECYCLE



OC setup → stores invited → approvers reviewing → approved guests confirming flights → concierge books → day-of live timeline → feedback & reconcile.

THE GUEST LIFECYCLE — THE SPINE



Parallel attributes, not stages: `SPECIAL_NEEDS` (dietary / accessibility / security) travels with the guest and fans out to the right vendor; `DROPPED` frees a slot the waitlist fills.

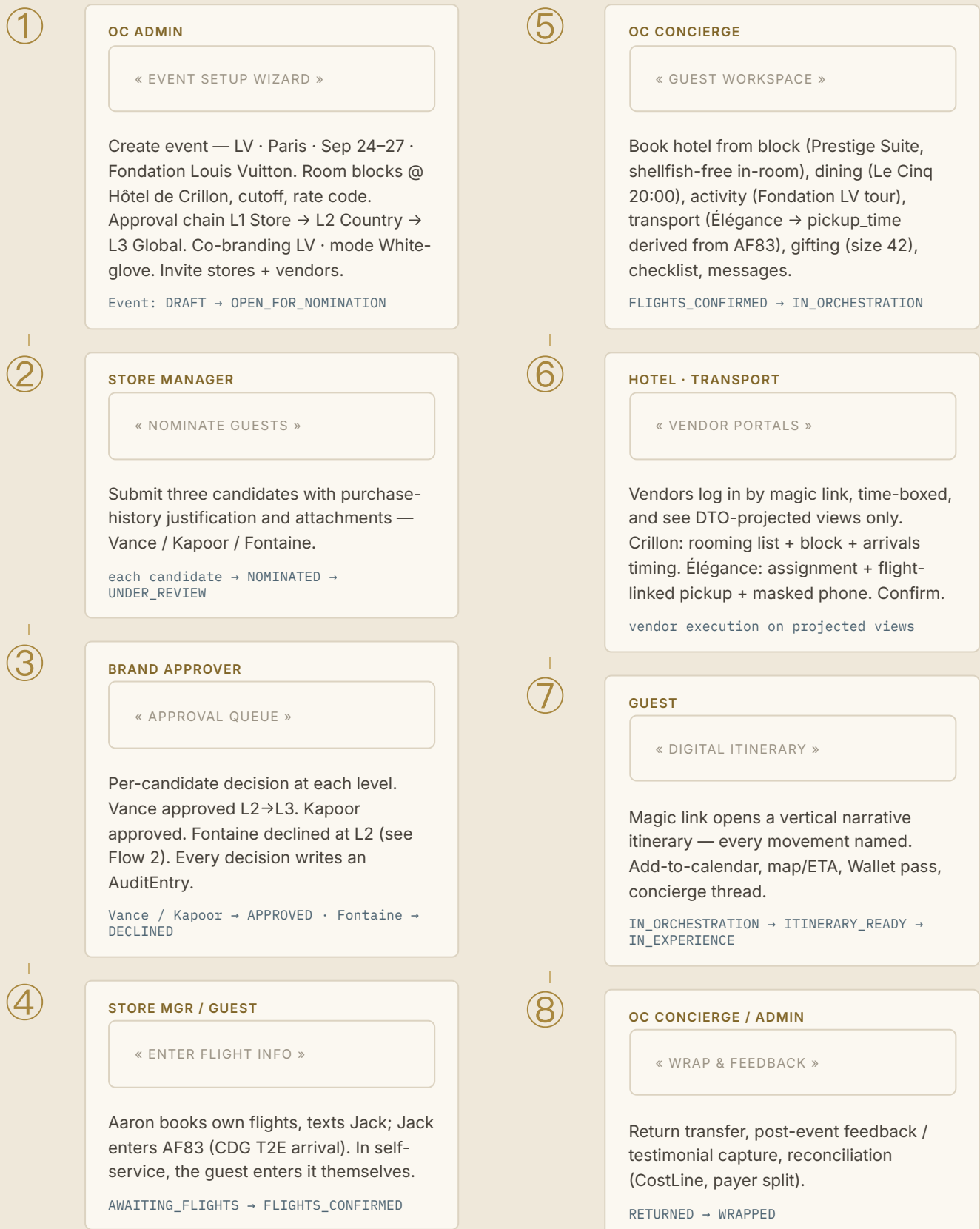
MULTI-LEVEL APPROVAL (§4.3)

Each candidate carries an approval-chain instance: per-brand config, default three levels, `Store` → `Country` → `Global HQ`. Each level is an independent decision: `APPROVE` advances, `DECLINE` terminates, `NEEDS_INFO` routes back to the store and pauses the chain, resuming on resubmit. Every transition writes an `AuditEntry`. Chain length and roles read from `Brand.approval_rules`, so a jeweller client can run two levels next week without a code change.

EDGE CASES, AS DESIGNED STATES

Partial approval per-candidate	Three rows, three states, honestly — the approved guest advances immediately. Flow 1 · B4/D1/E3
Declined / dropped → waitlist	Substitution prompt with ranked waitlist; released room returns to block; both store managers notified. Flow 2 · B9/B7
Missing flight AWAITING → MISSING	Block reason surfaced; amber T-7d → red T-3d; transport gated until flights land. Flow 3 · B6/E3/C5
Delayed flight re-derive	Single-write ripple across four surfaces, shown as one event. Flow 4 · marquee
Special-needs SPECIAL_NEEDS	"Who will see this" affordance shows the wall working; right field to right vendor. Flow 5 · C1

The canonical flow the running app demonstrates, following the LV Cruise 2027 Preview end to end. Each step is a state transition with an actor and a screen. The running cast is illustrative; the flight record is simulated.



One flight moves. Nobody re- types a thing.

This is the architectural spine of "the car is waiting even when the flight is late." A delay is modeled as a **single-write propagation**, not a re-typing exercise. Pickup time, hotel arrival, and downstream itinerary times are *derived* — they have no independent write path.

In steady state, AF83 is scheduled 14:20; pickup derives to 14:50, arrival to 15:35, and the itinerary reflows from there. Then the flight slips.

THE RULE THE ENGINEER ENFORCES

pickup_time, expected_arrival, and downstream itinerary times are **derived, never independently stored or edited**. Editing them means editing the Flight — or an explicit manual override that itself re-propagates from the one source. One write → everything recomputes → nobody re-types.

FALLBACK — NO WRONG VALUE, EVER

If the API is down or the flight isn't found, the system falls back to scheduled_arrival with a visible "awaiting live status" flag — no pickup silently goes wrong. A concierge can manual-override, which propagates identically; private jets and uncovered carriers are handled the same way *by design*. In-app flight behaviour is simulated and labelled simulated.

THE TRIGGER — THE SINGLE WRITE

```
FlightAware / Cirium poll or concierge  
manual override writes:  
Flight.actual_arrival = 15:50 (status:  
delayed)
```

```
one write · the single source of truth · source: api |  
manual
```

↓ fans out

Transport — Éléance

G1/G2

```
pickup_time = f(15:50 + buffer) = 16:20 · row re-times ·  
notify driver
```

"Pickup for A. Vance now 16:20 – flight AF83 delayed."

Hotel — Crillon

F3

```
expected_arrival = 17:05 · late-arrival flag raised ·  
rooming row updates
```

"Mr. Vance · expected 17:05 (flight delayed) · Terminal 2E."

Guest itinerary — Aaron

I1

```
ItineraryItem(s) reflow – the day becomes reassurance,  
not a problem
```

*"Your flight is running late – Alexandre will meet the delayed flight.
Everything else is handled."*

Exceptions panel — OC

B6

```
AuditEntry written · shown as ONE event, not four manual  
edits
```

*"AF83 delayed 90m – pickup, hotel, and itinerary already updated.
No action needed."*

Flow 1 · Partial approval

A nomination is a *set of candidates, each with an independent lifecycle* — there is no nomination-level approval. A mixed result is the default architecture rendering honestly, not a special case.

```
Nomination = { Vance, Kapoor, Fontaine }
├ Vance   APPROVE L2→L3 → AWAITING_FLIGHTS
├ Kapoor APPROVE L2→L3 → AWAITING_FLIGHTS
└ Fontaine NEEDS_INFO @L2 → loops back
```

Three distinct rows, three distinct states, side by side on «Approval Queue» and «Status Board». The approved guest advances the moment the decision lands — no waiting for siblings. Each per-candidate, per-level decision writes its own AuditEntry.

Flow 3 · Missing flight

A state, not just an alert. A guest in AWAITING_FLIGHTS with no Flight record past a threshold auto-flags and blocks orchestration at the transport step — no car is assigned against a guest with no arrival time.

```
AWAITING_FLIGHTS (no record)
↓ T-7d → FLIGHT_MISSING · amber
↓ T-3d → red · transport BLOCKED
↓ Jack enters AF83
→ FLIGHTS_CONFIRMED · unblocks
```

The row shows the block *reason*, "awaiting flight details," never a generic error. Amber → red is a designed urgency ramp on «Exceptions Panel» (B6) and «Status Board» (E3), with the transport step on «Guest Workspace» (C5) visibly gated so the concierge knows *why*.

Flow 2 · Declined / dropped + waitlist

Two entry points into one substitution machinery. Nominators may nominate beyond capacity; over-capacity approved guests sit WAITLISTED, ordered by approval time + VIP tier.

A · decline frees nothing:
Fontaine **DECLINE** → **DECLINED** (Jack notified)

B · a confirmed guest drops:
Vance → **DROPPED**
→ bookings released back to block (rooms_used --)
→ substitution prompt → top waitlist by rank
PROMOTE → WAITLISTED → AWAITING_FLIGHTS
· inherits the freed room-block allocation
· both store managers notified · AuditEntry

On «Waitlist / Substitution» (B9): the freed slot, the ranked waitlist, and a one-action promote with a confirm that shows exactly what moves. Released bookings return to the block, and usage decrements visibly on «Room-Block Intelligence» (B7). Never silent — both managers get a notification.

Flow 5 · Special-needs fan-out

Not an error path — a CRM-driven personalization that demonstrates the wall working. A flag lives on the durable profile and the per-event snapshot, and fans out to *exactly* the vendor that needs it.

Vance — shellfish allergy

→ Le Cinq: dietary_notes ✓ · Crillon in-room ✓
× Éléance: **NOT SENT** — a driver never sees dietary

Kapoor — wheelchair user

→ Crillon accessible room · Éléance accessible car
→ Activity step-free ✓ fact only
× medical detail never sent to any vendor

Security flag [if set]

→ OC + Security: real_name un-masked ✓
× all others: name → "Guest — LV VIP n"

On «Guest Workspace» (C1) the concierge sets a flag once; a "who will see this" affordance shows the fan-out — the operator *sees* the wall working. Need-to-know, shown.

The through-line for every new flow, stated once: **batch and bulk change how many records move in one action; they never change which records or which fields the actor may touch.** Each flow below is a convenience or a scoping rule laid over the Chapter One machine, never a new machine. The per-candidate lifecycle and the RBAC scope are preserved intact.

F1 · Event creation, from the UI

Chapter One created events in the seed only. Now it is a real four-step wizard, and it is where the two gates are *set* so they can be *enforced* everywhere downstream. Actor: OC Admin.

```
1 brand · name · city · venue ·
  dates
2 GUEST CAP · NOMINATION
  DEADLINE
3 room block ← catalog (F6)
4 approval chain · mode
  ↓ POST /events/new
  OPEN_FOR_NOMINATION ·
  AuditEntry
  → nav-reveal (R1) expands the
  event
```

Every field validated (capacity ≥ 1, dates present); a missing brand bounces to the form. Both gates are real from creation.

F2 · Batch approval

Many candidates, one action, still per-candidate. The critical property: batch is a *loop over the single-card decision*, inside one transaction, not a nomination-level verdict. Actor: Brand Approver + OC Admin.

```
FOR EACH selected candidate:
  scope guard ≠ brand → SKIP
  lifecycle guard (terminal) →
  SKIP
  write the SAME Approval row a
  card writes
  advance THIS candidate
  independently
  final level + cap full →
  WAITLISTED
  ONE AuditEntry per candidate
  → "7 approved · 1 waitlisted · 1
  skipped: already declined"
```

Partial approval unbroken, the cap unbroken, one audit row per candidate. The Zone board is read-only judge view: no batch, no single action.

F3 · The cap & the deadline

Set in F1, enforced at approval and at the nominate route. Two server rules, never UI hides.

```
F3a · the cap
final-level approval → count
seats taken
  taken < capacity →
  AWAITING_FLIGHTS
  taken ≥ capacity → WAITLISTED
  (rank)
  a drop frees a seat → promote
  top rank

F3b · the deadline
nominate route: now vs deadline
≤ → form OPEN
> → route CLOSED server-side
a late POST is refused, not
hidden
```

The cap is never overfilled by batch or single approval. Past the deadline the status board reads "Nominations closed."

F4 · Excel import

Creates records *within the caller's scope only*, after per-row validation, and can never move a field the caller cannot write. Actor: OC Admin + Store Manager.

```
rows: display_name, vip_tier, justification
FOR EACH row (in a transaction):
  validate name / tier → numbered error if bad
  store_id FORCED to caller's scope
  other store named → REJECT, never reassign
  passport / DOB / PNR / flag: not columns
  valid → INSERT (UNDER_REVIEW, zone denorm)
→ ImportJob + per-row reject report
```

Partial-file success allowed; “nothing invalid was guessed.” The shipped importer takes CSV pasted rows (opens in Excel) with no new dependency; an `.xlsx` parser is a drop-in behind the same route and guarantees. This is the lane the client’s live records travel at migration (\$DT).

F5 · Excel export

The caller's own screen projection, serialized. It can never contain a column the caller could not see on screen. Actor: OC + Store Manager + Zone Manager.

```
OC Admin    → name, tier, store, status
Store Mgr   → name, tier, status (own store)
Zone Mgr    → name, tier, store, purchase_summary
              (own zone, judge-only – no logistics)
→ spiti-guests-<role>.csv · AuditEntry
```

An adversarial read of the export is an adversarial read of the caller's screen. Every export is audited.

F6 · Property-catalog picker

Free text becomes a reference. A room block now points at a catalog Property, filterable by category, country, city, neighborhood.

```
/properties - filter (server-side WHERE)
table: name · category · location · scope
picker inside /events/new step 3:
choose Property → room block:
  property_id → catalog reference
  property_name → label / fallback
```

One catalog, no free text; the same Crillon row serves an LV event this year and a Cartier event next. No vendor's DTO widens: cataloging changes the hotel field from text to a reference, nothing more.

R_I · Nav-reveal

An event's sub-sections stay hidden until it is opened, backed by server scoping, not a UI trick.

```
default nav (no event): productions · new event ·
  catalog · import · audit · users
open an event → rail EXPANDS:
  command center · guests · movements ·
  exceptions · room block · vendors · budget
leave → collapse. sub-routes 404 without
  an event context (buildNav openEvent)
```

The hide is enforced the same way the data is scoped: the sub-routes redirect without an event in context.

M · THE MESSAGE-LANE MATRIX — A LANE IS A WALL

You see a lane only if you are a participant in it. Reads are participant-gated in the query, never client-filtered; a message you are not a party to never enters the result set.

lane	OC ADMIN	OC CONC.	CLIENT DIR	ZONE MGR	APPROVER	STORE MGR	HOTEL	TRANSP.	REST.	GUEST
guest_facing	R/W	R/W own	—	—	—	R/W own	—	—	—	R/W self
internal	R/W	R/W own	—	—	—	—	—	—	—	—
hotel_oc	R/W	R/W own	—	—	—	—	R/W own	—	—	—
brand_oc	R/W	R/W own	R/W brand	R zone	—	—	—	—	—	—

Deliberate refusals, stated as features: the Brand Approver has *no* lane (a decision surface, not a correspondent); a hotel and a brand never share a thread (OC is the hub, each spoke its own wall); the guest sees only *guest_facing*, absolute. A message body is prose, so opening a `brand_oc` thread never joins a guest record.

MAPS

Ten roles, ten journeys.

One event, one cast, followed by everyone, top of the house to the received guest. Each journey names the moment, what they see, where it delights, and the wall: the field deliberately absent, and why.

ABOVE THE WALL

Marielle sees across every production; the concierge remembers last year's Barolo; the guest is simply received.

AT THE WALL

The approver judges but never sees the room; the driver gets a name, a flight, a phone that unlocks 24h out.

THE CAST, ILLUSTRATIVE

LV Cruise 2027, Paris · Sep 24–27 · a worked example, not real client data.

OC Admin / Operations — *the super-user, audited*

Shell: efficient (desktop-rich). Scope: none — the only cross-tenant role, and every access is logged. Her goal: stand up the event, invite the right stores and vendors, and watch every arrival and exception on one screen, trusting the walls to hold.

MOMENT	WHAT SHE SEES	WHERE IT DELIGHTS	THE WALL
Portfolio home	Health tiles per event, all brands: LV Cruise — 27 confirmed / 3 awaiting / 1 exception; Cartier Geneva; Aston Martin Monaco.	<i>The exception count is the only number in accent — the eye goes straight to trouble.</i>	The only portfolio-wide surface in the product. Every other role is walled to one brand.
Create event → DRAFT	Five-step wizard; co-branding preview re-skins the shell to LV live as she picks; mode set to White-glove.	<i>She watches sign-in and the guest itinerary become an LV experience before a guest is invited.</i>	—
Invite stores + vendors → OPEN_FOR_NOM	Vendor roster with access_window per vendor: Crillon T-14d, Éléance T-7d; both close T+3d.	<i>Windows are set once and expire themselves — she never has to remember to revoke a driver's login.</i>	She grants access; she never enters guest PII for vendors. The DTO projection decides what each vendor sees — she cannot over-share.
Day-of command center IN_EXPERIENCE	Today's movements — every guest, every vendor, timed; exceptions panel live.	<i>AF83 slips 90m → she sees the ripple as one event, not four people re-typing.</i>	She sees everything by design — but every access writes an AuditEntry. Audited, not exempt.

OC Concierge / Account Manager — *the invisible private secretary*

Shell: efficient. Scope: owned guests + assigned events. His goal: book every room, dinner, car, and gift; know each guest's allergies and preferred suite before they ask; keep the itinerary reflowing when a flight moves.

MOMENT	WHAT HE SEES	WHERE IT DELIGHTS	THE WALL
My guests various	His Paris roster only: Vance — CONFIRMED, Kapoor — ORCHESTRATION; not another concierge's guests.	<i>The board is his world, calm not cluttered.</i>	No portfolio view. He cannot read another concierge's guests — owned_guest scope on every query.
Guest workspace ORCHESTRATION	Full CRM: VIP Tier 1, prefers a corner suite, allergic to shellfish, drinks Barolo, size 42, English/French, 3 past LV events.	<i>Last year's memory is already on the page — service compounds, it doesn't restart.</i>	Full PII — he <i>is</i> the private secretary — but only for guests he owns.
Arrange transport	Pickup CDG T2E, drop Hôtel de Crillon, pickup time shown <i>derived</i> from the flight — greyed, "auto-adjusts to AF83."	<i>He never types a pickup time. The system computes it and re-times on delay — the promise made literal.</i>	—
Special-needs fan-out	Sets Kapoor's accessibility fact once; system routes an accessible room to Crillon, an accessible vehicle to Éléance.	<i>He records the fact once; the right vendors get exactly the mobility fact and nothing medical.</i>	The medical detail reaches no vendor. Transport gets "wheelchair-accessible vehicle," not why.

Brand Approver — *Global (L3) & US Country (L2), judge-only*

Shell: spacious (judge-only). Scope: own brand, own level. Their goal: decide *who* comes — judge each nominee on relationship and purchase history, fast, on defensible grounds, with a clean record. They decide the guest list; they never touch logistics.

MOMENT	WHAT THEY SEE	WHERE IT DELIGHTS	THE WALL
Approval queue UNDER_REVIEW	Justification cards: <i>Vance — VIP Tier 1, Dallas, "top collector, 3 prior shows, relationship since 2019," nominated by the Dallas store.</i>	<i>The card is a dossier, not a form row — everything to judge, nothing to distract.</i>	No logistics PII on the card. No passport, no flight, no hotel, no room, no address, no itinerary.
Decide per candidate → APPROVE/...	Three clear actions per card + a note field; decision advances to the next level or terminates.	<i>Approving one and declining another in the same nomination is one screen, not a workaround.</i>	Cannot see other levels' private deliberation beyond the decision + reason surfaced.
Multi-level handoff	Chain view: L1 submitted, L2 approved (actor + timestamp), L3 pending.	<i>The chain is a clean audit trail — "who decided what and when," exactly as asked.</i>	—
Approved roster post-decision	Read-only roster: names + tiers + status, no logistics.	<i>She can reassure the house the guest list is right — without seeing where anyone sleeps.</i>	No PII beyond what approval required. Logistics is a different wall.

Store Manager / Nominator — *works from the sales floor, on a phone*

Shell: spacious, mobile-friendly. Scope: own store. His goal: get his top clients invited, track approval without chasing anyone, tell them the good news, and enter the flight so OC takes over — the bridge between the relationship and the machine.

MOMENT	WHAT HE SEES	WHERE IT DELIGHTS	THE WALL
My nominations	Only his Dallas nominations across LV events; approval states at a glance.	<i>His store's world only — clean, personal, mobile.</i>	Hard wall: Dallas cannot see Miami's or Beverly Hills' nominees. <code>store_id</code> scope, every query.
Nominate → NOMINATED	Guided nomination: pick client from his store's CRM, write the purchase-history justification, attach a note.	<i>Writing "top collector, 3 prior shows" feels like advocating, not filling a form.</i>	—
Enter flight → CONFIRMED	A focused flight-entry card for his approved guest: airline, number, arrival, terminal, PNR optional.	<i>The single act that hands the guest to OC — he feels the baton pass.</i>	He sees his guest's flight because he entered it; not vendor detail, budget, or OC internal notes.
Reassure his client	Read-only itinerary view for his own guest.	<i>He can say "you're all set, a car meets your flight" — because he can see it's true.</i>	Read-only. He reassures; he doesn't orchestrate or edit OC's bookings.

Missing-flight nudge: if the flight isn't entered by T-7d the guest row goes amber — *"Flight details needed for Aaron Vance"*; T-3d it goes red. In-app + email/SMS. Designed urgency, not a nag (Flow 3).

The org chain now reads **Client Director → Zone Manager → Store Manager**, continuous top to bottom. These two roles are the sharpest walls in the product: maximum authority over *people*, and, for the Director, zero reach into *guests*. Same LV Cruise 2027 example, two named leaders added to the cast: *Irène Lacroix*, LV Client Director, and *Priyanka Rao*, Zone Manager for North America East (LV Miami), who sees Miami's candidates and never Dallas's.

Client Director — *the org atlas, never a guest board*

Shell: spacious. Scope: own `brand_id`, org-management only. DTO: `BrandOrgView`. Her goal: decide who at the house holds power (which people are Brand Approvers, which are Zone Managers) and read the scale of what is in flight, without ever touching a single guest.

MOMENT	WHAT SHE SEES	WHERE IT DELIGHTS	THE WALL
My brand /director	Four calm tiles — Approvers · Zone Managers · Stores · Events — and an <i>Events at a glance</i> table: name, city, dates, stage, and a guest count ("Cruise 2027 · Paris · 12 guests").	<i>An org atlas, not a guest board. The one accent number is the event count: the scale in flight, at a glance.</i>	No guest anywhere. The count is a single integer: no name, no attribute, no derivable PII. <code>BrandOrgView</code> never joins a guest table.
The org chart /director/people	Three grouped tables — Approvers (by level), Zone Managers (by zone), Store Managers (by store · zone, for the record) — each with name, email, active/suspended.	<i>The whole house's chain of command on one page.</i>	She sees the org tree; never a nomination, a verdict, or candidate content. She appoints the people who judge; she never judges.
Appoint a person /people/new	A short form: role (<i>only</i> Approver or Zone Manager), name, email, and a level or a zone; a magic-link invitation is sent.	<i>Two roles, one form, one house — she stands up her leadership in seconds.</i>	Only two roles, only her brand. A Store Manager she cannot appoint (that is the Zone Manager's job); any other role or another brand's person is refused server-side and audited.

Zone Manager — *the approver's judge-view, widened one level and no deeper*

Shell: spacious, judge-only. Scope: own `brand_id` **and** own `scope_zone_id`. DTO: `ZoneNominationView`. Her goal: judge candidate quality across her zone's stores (a level above one store, a level below the chain) and staff the store managers who report into her zone.

MOMENT	WHAT SHE SEES	WHERE IT DELIGHTS	THE WALL
Zone nomination board /zone/nominations	Judge-only dossier cards across <i>every store in North America East</i> : display name, VIP tier, the nominating store , the store manager's justification, purchase summary, prior-shows count, state — read-only.	<i>The approver's dossier, widened to her whole zone, carrying the store attribution the approver's card doesn't.</i>	No logistics, ever, and no decision button. No passport, flight, hotel, transport, itinerary. She oversees; she does not approve or decline.
My stores & managers /zone/stores	Her zone's stores, each with its Store Manager(s) and contact; a <i>Manage</i> action per store.	<i>The people surface she owns — clean, scoped to her zone.</i>	Only her zone's stores; Dallas (North America Central) never enters her result set.
Appoint a store manager /stores/:s/people	A short form scoped to a store in her zone; a magic-link invitation is sent.	<i>She staffs her own zone's stores — the chain of authority stays honest.</i>	Only <code>store_manager</code> , only for a store whose <code>zone_id</code> equals hers. A cross-zone attempt is refused server-side and audited.

Hotel — the Hôtel de Crillon

HotelGuestView · time-boxed T-14d → T+3d.

Rooming list	display_name, party_size, room_type (Prestige Suite), dates, rate LV-CRUISE-27, special_requests_stay (shellfish-free; accessible room for Kapoor).
Arrivals timing	expected_arrival (derived, timing only), terminal, late-arrival flag when AF83 slips.
The wall	No passport, no PNR, no address, no purchase, no transport, no itinerary. The full record never crosses the wire — a projected view.

Transport — Éléance Paris

TransportGuestView · time-boxed T-7d → T+3d; phone unmask in-window.

Assignment	display_name, pickup CDG T2E, drop (address), pickup_time (derived), flight_number, vehicle (accessible for Kapoor), route_notes.
Live delay	Pickup row re-times itself; a notification fires — the car waits even when the flight is late.
Contact	+33 4821 until T-24h, then unmasked in-window. Discretion with a clock on it.
The wall	No PNR, no room number, no dietary, no purchase. Name + flight + pickup + phone is the entire surface.

Guest — the received arrival

Shell: spacious, mobile-first, editorial — never looks like software. Scope: self, always. His goal: be looked after. He never signs into a "system."

Itinerary opens	<i>"Thursday – AF83 lands 14:20 CDG T2E · Alexandre from Éléance meets you at Arrivals · Hôtel de Crillon, Prestige Suite · 20:00 dinner, Le Cinq." Every line names the real thing.</i>
A flight moves	<i>The day reflows into reassurance: "Your flight is running late – Alexandre will meet the delayed flight. Everything else is handled."</i>
Message concierge	<i>"Tomás – can we do a later dinner Friday?" → "Of course. Moved to 21:00. – T"</i>

TWO MODES, ONE TREATMENT

In **white-glove** (default) the guest receives a beautiful itinerary and never touches an admin surface. In **self-service** (per-event toggle) Spiti adds guided choices to the front, one considered question per screen, curated hotel/dining/activity cards, never a bare form, and both modes end at the same hero itinerary. The toggle changes what the guest *does*, never how they are *treated*.

OTHER VENDORS — THE MODEL IN ONE GUEST

Vance's shellfish allergy reaches *Le Cinq* and never *Éléance*. Kapoor's accessibility *fact* reaches the Crillon and Éléance; the *medical detail* reaches neither. A security-flagged guest's name masks to hotel, transport, restaurant, and activity — and unmask only to OC and the security detail. Need-to-know, shown, not asserted.

THE ANNOTATED SCREENS

Zero clarifying questions.

Every screen names its purpose, role, shell, exact fields from the role's grants, actions, and all four states — empty, loading, error, edge.

A1 P1

Sign-in / entry

One co-branded door for every role; resolves to the right shell after auth. A vendor's or guest's magic link carries the event → LV skin.

FIELDS email/phone → magic link; optional SSO for brand HQ. No password.

"Welcome. Enter your email and we'll send you a secure link."

B2 P1

Event Command Center

The state of one whole production at a glance — guest-state summary, today's movements, exceptions, room-block usage.

STATES empty (setup checklist) · per-panel retry, page still usable.

B3 P1

Event Setup Wizard

Stand up an event end to end — five steps: basics, room blocks, approval chain, co-branding & mode, invites.

STATES draft · inline validation ("Add a room block before inviting the hotel") · publish confirm.

B6 P1

Exceptions Panel

Everything that needs a human, in one place. The delay ripple shows as *one* line, not four edits.

"AF83 delayed 90m — pickup, hotel, itinerary already updated. No action needed."

B4 · B7 · B9 P1

Board · Block · Waitlist

B4 per-guest status board, mixed states honest. **B7** room-block intelligence — usage, cutoff, attrition. **B9** waitlist / substitution, one-action promote with confirm.

COPY "18 of 25 rooms held · cutoff in 6 days · rate LV-CRUISE-27."

C1 P1

Guest Workspace — CRM

The durable memory; where service compounds. Full PII — the concierge is the private secretary. Passport/DOB masked-by-default with reveal (encrypted).

EDGE "who will see this" fan-out affordance shows the wall (Flow 5).

"Prefers the corner suite · Barolo in-room · allergic to shellfish · size 42 · 3 prior LV events."

C2 · C5 P1

Flights · Transport

The source of truth and the derived car. Manual override re-propagates; pickup_time read-only, "auto to AF83." Gated if FLIGHT_MISSING.

"Élégance Paris · CDG T2E → Hôtel de Crillon · pickup auto-times to AF83."

C3 · C4 · C6–C8 P1

Hotel · Dining · Gifting · Tasks · Messages

Book from block; curated dining/activities; gift sized from CRM; the per-guest run-list; the two-lane thread with a hard partition between guest-facing and internal.

COPY "Dinner · Le Cinq · Friday 20:00 · party of 2 · no shellfish."

D1–D4 P1

Approver — judge-only

Every screen renders ApproverGuestView — logistics PII is absent from the projection and cannot render. Queue, nomination detail, chain view, read-only roster.

"VIP Tier 1 · Dallas · top collector · 3 prior shows · relationship since 2019." Never a bare form row.

E1–E5 P1

Store Manager — mobile

Own store_id only. My nominations, nominate, status board, enter flight (the baton pass), read-only "my guest."

"Thank you — OC Travel takes it from here."

F1–F3 P1

Hotel Portal

Renders `HotelGuestView` only — the screen literally cannot show a passport. Rooming list, block status, arrivals timing. Time-boxed.

STATES empty · loading · **expired** — "This event's access window has closed." · masked-guest.

"Mr. Vance · expected 17:05 (flight delayed) · Terminal 2E."

G1–G2 P1

Transport Portal

Renders `TransportGuestView` only. Assignment schedule + single-assignment detail; phone masked until T-24h then in-window.

STATES empty · delayed (pickup re-times + notification) · expired · masked-phone.

"A. Vance · CDG T2E → Hôtel de Crillon · pickup 16:20 · AF83."

H1–H3 P2

Other vendors

Type-specific DTO, where a grant to one type is never a grant to another. Restaurant (dietary), Activity (mobility fact), Security (real_name un-masked, the one that un-masks).

"Party of 2 · Friday 20:00 · no shellfish."

I1 P1

Digital Itinerary — the hero

The whole trip as a beautiful vertical narrative; the file that makes a guest feel a private secretary wrote it. No nav rail, no tables, no system chrome. Held to the Aesop/Cereal register and WCAG 2.2 AA.

STATES ready · pre-ready (graceful holding, never an empty scaffold) · flight-delayed (reflows to reassurance) · offline (cached, readable at an airport) · editorial skeleton.

"Thursday. AF83 lands 14:20, CDG Terminal 2E. Alexandre from Éléance Paris will be waiting at Arrivals. Hôtel de Crillon, Prestige Suite. 20:00, dinner at Le Cinq."

I2 · I3 P1

Day · Messages

One day expanded; the only "system" a white-glove guest touches — and it reads like texting a private secretary.

I4–I6 P1 toggle

Self-service

Guided welcome (consent captured plainly), curated choosers (hotel from block, dining, activities — editorial cards, never a dropdown), confirm & flights. Cannot pick outside the curated set.

"The Prestige Suite, overlooking the Place de la Concorde."

J1–J6 P1/P2

Cross-cutting & edges

Notifications center (lane-aware, channel fan-out); and the four edges as named states: missing-flight (J2), delay ripple (J3), partial approval (J4), waitlist substitution (J5), special-needs fan-out (J6).

GUARANTEE a denied field is absent from the projection — the screen cannot render it. Every P1 names purpose, role, shell, fields, actions, states.

GLOBAL STATES CONVENTION

Loading = content-shaped skeletons, never a spinner.
Error = a calm, specific message + a recovery action.
Empty = a composed invitation to the first action. Degraded = scheduled data + "awaiting live status." AA on every screen.

Every field named here is drawn from a DTO or a scoped query in the shipped code, so a denied field is not merely hidden — it is absent from the projection. Each surface carries its `wall` callout: the need-to-know boundary stated on the screen itself, as a designed feature.

CD1 Ch.2

Client Director — My brand

An org atlas plus the scale of every event as a count. Four stat tiles (Approvers · Zone Managers · Stores · Events) and an *Events at a glance* table with a guest **count**, never a name.

FIELDS name · city · dates · stage chip · guest_count (one integer). **DTO** BrandOrgView.

“The people who hold power at the house — and the scale of what is in flight. Never a guest, never a nomination.”

ZM1 · ZM2 Ch.2

Zone Manager — board

Six oversight tiles, then a judge card grid, two-up. Each card: name, tier, nominating store, the store manager's note, purchase summary. **No action buttons.**

FIELDS display_name · vip_tier · store_name · prior_events · purchase_summary · state. **DTO** ZoneNominationView — logistics never joined.

“You judge, you do not decide. And the card carries no logistics: the approver's dossier, widened to your zone and no deeper.”

RV1 Ch.2

Le Cinq — Reservations

The restaurant/experience portal, now running — the sibling of the hotel and transport portals. A covers list: guest, party, time, venue, and the one type-keyed note.

FIELDS display_name (masked on flag) · party_size · reservation_time · venue · one of dietary (restaurant) or access (experience). **DTO** RestaurantGuestView.

“Exactly what the kitchen needs: covers, time, and the dietary brief. Nothing more.”

CD2 · CD3 Ch.2

Org chart · Appoint

Three grouped tables (Approvers by level, Zone Managers by zone, Store Managers for the record) with Suspend/Reinstate. CD3: a form offering *only* Approver or Zone Manager, own-brand.

STATES empty group “None appointed yet” · suspended chip · server refusal (forged role / cross-brand) 403 + audit.

ZM3 · ZM4 Ch.2

Stores · Appoint SM

Her zone's stores + managers with a Manage action; a form scoped to a store in her zone (the store id is validated before the form renders).

WALL only store_manager, only her zone. A :store outside the zone renders a plain 403, never the form.

RV2 Ch.2

Single reservation

One booking on one card — the whole surface for that booking.

STATES empty (within window) · **expired** closed-window notice · masked guest “Guest, VIP 1” · not-found redirects to the list.

“No route from here to a passport, a flight, a room — those fields are not on your projection.”

The screens are designed cards, not baked photographs. The running app's real HTML is engineer-generated via the shipped tool; the dossier carries these annotated cards in the book's own Screens convention, the honest on-bar way to spec a surface.

OA1 · OA2 Ch. 2

Houses · one brand

Each brand as its own surface with its event count, replacing the flat grouping. A grid of brand tiles (archetype, mode, Events + Guests-in-flight); the tile links to `/brands/:id`.

WALL each brand is a tenant; the only place that sees across houses. No brand reads another's data.

OA3 · OA4 Ch. 2

Property catalog · Add

The filterable hotel catalog that replaces free text — filter row (category · country · city · neighborhood, server-side WHERE); a table with a scope chip. OA4 adds one catalog row, audited.

STATES "No properties match" · empty catalog → only the Add action · filters retain their value.

OA5 Ch. 2

Event-creation wizard

Four numbered step cards — the event; capacity & deadline (each with an inline note on enforcement); room block from the catalog; approval chain & mode. Flow F1.

"The cap and the deadline are real from step two. The guest cap drives the waitlist the moment approvals exceed it; the deadline closes the nominate surface server-side. Neither is a label."

OA6 Ch. 2

Batch approval

On the existing approver queue: a checkbox per card, a sticky "N selected" action bar with the three actions + a shared note, and an honest result panel. Flow F2.

EDGE mixed result names every skip with its reason. **ABSENT** on the Zone board (read-only judge view).

OA7 Ch. 2

Import guests

A CSV textarea (display_name, vip_tier, justification), Validate & import, and a result screen. Flow F4. The scope note adapts to role.

"Passport, DOB, PNR, and security flags are not import columns — a spreadsheet round-trip can never move them. Every import is audited."

STATES all valid · partial (rejected rows by number + reason, "nothing invalid was guessed") · empty paste returns without writing.

MSG1–3 Ch. 2

The message lanes

MSG1: Client Director writes `brand_oc` (tab strip, thread, composer). MSG2: Zone Manager reads it, no composer, · *read-only*. MSG3: OC hub gains the `hotel_oc` tab beside guest-facing / internal.

"A lane is a wall. You correspond with OC Travel, and only OC Travel. Opening this thread never reveals a guest record."

THE CHAPTER TWO STATE CHECKLIST

Every new screen ships with its designed states. **Empty**: composed copy, never a blank grid. **Loading**: counts resolve first (the fast fact), then tables fade in — never a spinner over a blank page. **Refusal**: vendor windows return a dignified closed-window notice; adversarial refusals (cross-brand appoint, cross-zone appoint, forged role) return a plain server message the legitimate UI never surfaces, permanent, not a retry prompt. **Edge**: masked names on security-flagged guests; partial-import per-row reasons; batch results honest about every skip; nav sub-sections present only with an event in context.

§ DT

Their curated world in. Their people, by their own hand.

The move onto Spiti is designed to live Spiti's soul: the curated, non-sensitive world is brought in for the client; the sensitive live records they migrate themselves, on their own server, the discreet way. The split is the product's privacy posture applied to its own onboarding.

STUDIO SEEDS

Venues, hotels, restaurants, brands, zones — the real curated inventory. No guest records. No PII.

CLIENT MIGRATES

Live guest records, by the client's own engineer, on the client's own server, through the scoped Excel import.

THE HARD LINE

The sensitive data never sits on a studio bench, not even during the move.

The client is moving off the old platform onto Spiti. This is the one Chapter Two concern that earns its own section, because the migration itself is designed the product's way: **the curated, non-sensitive world is brought in for them; the sensitive live records they move themselves, discreetly.** The split is not an accident of tooling. It is the need-to-know posture applied to day zero.

DT.1 · WHAT THE STUDIO SEEDS — NO PII

The Spiti install opens with the client's own world already inside it, so the first day feels like a move-in, not a cold start.

The Property catalog — the client's actual venues, experiences, hotels, restaurants, filterable by category, country, city, neighborhood (F6).

The org structure — the client's real brands and zones, and the store map under them.

This is curated reference data, not personal data. It carries **no guest records and no PII**. The catalog picker in the event wizard already lists the client's properties; the org chart already shows their houses and zones.

DT.2 · WHAT THE CLIENT MIGRATES — ON THEIR SERVER

The live guest records do not travel in the delivered seed. The client's own engineer migrates them, on the client's own server, through the Excel import (F4), documented in `DEPLOYMENT.md`. This is the discreet way, for three reasons:

The sensitive data never passes through the studio. A seed with real PII would put HNW records on a studio bench. The people who should hold that data are the client, on their own infrastructure.

The import is already the scoped, validated, audited lane for exactly this. Passport, DOB, PNR, and security flags are not import columns — even the migration cannot move them through a spreadsheet.

It matches how the product treats the data forever. Onboarding behind the same need-to-know wall, rather than a bulk PII handover, is that posture on day zero.

DT.3 · THE SPEC OBLIGATIONS THIS CREATES

DEPLOYMENT.md is a named deliverable

It documents the migration for the engineer: apply the schema migration (§3.5), confirm the seeded catalog and org, then run the guest import per event on their server, reading back the `ImportJob` error report until every row is clean.

The seed is auditable as PII-free

The delivered seed contains catalog, org, and illustrative demo guests only; the client's real guests appear only after their own import runs on their own server.

The import template is the migration contract

The columns the engineer fills are exactly the fields a brand-side caller may write. The template cannot carry passport, DOB, PNR, or security flags, so the migration inherits the product's field-level wall.

THE ONE-LINE FRAME FOR THE CLIENT

We bring your curated world into Spiti for you, and you bring your people into Spiti yourself, on your own server, through the same discreet, validated door the product uses every day. Nothing sensitive ever sits anywhere it should not, not even during the move.

This is data residency and PII minimization applied to onboarding itself: the client's live records stay in their own data-residency region, and the studio never holds them. Audit covers the migration like any sensitive access — every import writes an `AuditEntry` with actor, scope, and row count.

Flight status — the signature touch

FlightAware AeroAPI (v1) · Cirium (deeper international) · behind a `FlightStatusProvider` adapter

Poll per active flight on a cadence that tightens near arrival (every 30 min at T-24h, every 5 min at T-2h — *illustrative*). Each poll writes `Flight.actual_arrival` + status *once*; the propagation job fans out (Flow 4).

Fallback, three layers. (1) Manual override: a concierge sets arrival/status by hand (`source: manual`), propagates identically, covers private jets and uncovered flights. (2) Graceful degradation: API down falls back to scheduled plus a visible "awaiting live status" flag, no pickup silently wrong. (3) Provider abstraction: Cirium ↔ FlightAware swappable, a future provider drops in.

Notifications

Postmark / SendGrid (email) · Twilio (SMS + WhatsApp) · web push

Triggered by state transitions — APPROVED, FLIGHTS_CONFIRMED, ITINERARY_READY, "driver en route," the delay ripple. Templates carry co-branding + the concierge register, guest-facing lane only.

Calendar & maps

.ics per `ItineraryItem` · Google Maps / Mapbox

Add-to-calendar per item; routing and itinerary map tiles for transport and the guest itinerary.

Exports — PDF & Wallet

Headless Chromium / PDF service · Apple PassKit · Google Wallet

Server-side render for itinerary booklets, rooming lists, manifests — composed to the luxury bar, not a table dump. Wallet passes (.pkpass + Google) signed server-side.

Auth & SSO

Auth.js / Clerk / WorkOS · optional SAML/OIDC

Magic-link / passwordless for guests and vendors; optional enterprise SSO for brand HQ teams. Sessions carry role + scope + (for vendors) `access_window`.

Optional brand CRM/POS hook future

An adapter so purchase-history justification can be *pulled* into a nomination rather than typed — a per-brand inbound connector. v1 accepts typed / attached justification.

THE TRUTH STANDARD, RESTATED

Providers named above are real. All in-app flight behaviour in the delivered portal is **simulated realistically and labelled simulated**. Cadences and buffers are illustrative; the client confirms real values at build. Start on one flight provider behind the adapter interface, so the second is a config swap, not a rewrite.

For HNW international clientele, discretion and data protection *are* the luxury. The frame below is drawn from the phased plan and extended with the operational lessons the client's own experience makes non-negotiable.

6.1 RBAC & need-to-know — two walls

Server-side field projection (DTOs) *and* Postgres Row-Level Security — two independent walls, defense in depth. Deny-by-default. The adversarial checklist (§2.5) is a passing test suite before v1, not a hope.

6.2 Time-boxed vendor access

The `access_window` gate at the query layer; expired = empty result, not stale cache. Phone unmask only in-window; re-granting requires an audited admin action. *Illustrative*: hotel T-14d, transport/other T-7d; both close T+3d; phone at T-24h.

6.3 Secrets management — the anti-pattern forbidden

No secret of any kind is ever committed to source. No SMTP password, no API key, no database URL, no signing key — in no committed `.env`, ever. Secrets live in client-controlled config, injected at runtime; the repo ships an `.env.example` that documents but never contains a credential; every secret is rotatable config-only; `RUN.md` walks first-time setup step by step.

6.4 Auth & email on an owned domain — the resilience lesson

Transactional email and sign-in run on **a domain OC Travel owns outright**, with OC-owned SPF/DKIM/DMARC, billed to OC. Magic-link auth is chosen for resilience — no single external mailbox outage can lock everyone out; the fallback is another sender on the *same owned domain*. Optional SSO federates to the brand's own IdP for their staff.

6.5 No vendor lock-in — the ownership principle

The client owns **source, secrets, domain, and data**. Source in a repo they control; secrets in their accounts; auth/email on their domain; data in their region-pinned Postgres, exportable, no trapping format. Providers sit behind adapters, so any one can be swapped without a rewrite.

6.6 GDPR / CCPA / privacy by design

Consent captured on `GuestProfile` with lawful basis recorded. Right-to-erasure as cascade/anonymize per retention policy. PII minimization enforced by the DTO projection — a vendor never receives PII it doesn't need. Paris/EU involvement makes this real, not theoretical.

6.7 Retention, deletion & data residency

Per-brand retention policy; the CRM persists across events by default but is subject to configurable retention + explicit erasure. Audit log retained per compliance window. Region-pinned managed Postgres — EU guest data can stay in-region (`Brand.data_residency_region`).

6.8 Audit trail

`AuditEntry` on every state transition, sensitive-field access, and permission grant/expiry. Immutable append. OC Admin reads the global log; an approver reads only their chain slice. The super-user is audited, not exempt.

6.9 Accessibility — WCAG 2.2 AA

Both shells meet AA. The guest itinerary (the hero) meets AA including reduced-motion, contrast, and screen-reader reading order. Vendor/ops surfaces are keyboard-navigable. Designed into every screen's states, not retrofitted.

6.10 Multi-language — i18n from day one

Locale on `User` + `GuestProfile`; the guest itinerary and notifications localize; the concierge register is preserved per language via copy keys, *not* machine-translated — the voice survives translation.

6.11 • 6.12 Encryption & performance

Passport / PNR / DOB encrypted at rest; TLS in transit; signed, expiring URLs for exports. Boards are server-side-paginated with indexed tenant/scope queries; the itinerary is mobile-first, fast at an airport, and cached to read offline.

Phase I is done when every line below is true.

The golden-path scope is locked. Phase 1 ships when the seven conditions at right hold — not "mostly," and not "with TODOs." Enterprise-grade means real working code, real architecture, real persistence.

PHASES 2-3 — SPECCED, ADDITIVE

Command-center depth, other-vendor portals, the full room-block dashboard, SSO, budget, self-service depth, CRM intelligence, reusable templates, analytics, and the brand CRM/POS inbound hook are **fully specced in this dossier** — so they are additive, never a rewrite. The model, the wall, and the flows are built once and extended.

THE SINGLE ACCEPTANCE PRINCIPLE

The two walls hold as a test suite, the flight ripple is a real branch, every edge is a designed state, and the client owns source, secrets, domain, and data from day one. A cold reader stands the portal up on their own accounts from RUN.md alone.

✓ **Every golden-path role works end to end** on shared event data, at the luxury bar at both faces.

✓ **The two walls hold** — DTO projection + Postgres RLS; the §2.5 adversarial checklist passes as a test suite.

✓ **Flight propagation is live** — FlightAware adapter + manual-override fallback + the single-write ripple to transport, hotel, and itinerary (simulated, labelled).

✓ **Every edge case is a real branch**, not a toast — partial approval, waitlist substitution, missing flight, delay ripple, special-needs fan-out.

✓ **Both modes run the golden path** — white-glove default; self-service toggle.

✓ **Audit writes on every transition** + every sensitive access.

✓ **Ownership is real** — secrets in client-controlled config with an `.env.example`; auth + email on an OC-owned domain; source/secrets/domain/data are the client's; RUN.md walks a cold first-time setup.



Spiti

BY OC TRAVEL

THE VOLUME

The Buildable Spec Dossier for Spiti, the event-concierge portal by OC Travel — volume two of the slipcase, and one coherent whole. Ten roles and their journeys, the golden path and every edge flow, the RBAC wall drawn field by field, the data model and its org and catalog spine, the annotated screens, the data-transfer story, the integrations, and the non-functional frame.

Presented from the architecture (the roles matrix, the data model, the flows, the phased plan) organized for reading and never re-decided. Where this dossier and the architecture differ, the architecture wins.

FACES & THE TRUTH STANDARD

Set in Cormorant Garamond, Newsreader, Inter, and IBM Plex Mono — all open-licence (SIL OFL 1.1) and self-hosted; the dossier renders with no runtime font fetch, and wears the same `design-tokens.css` skin as the running portal.

Integration providers named are real; all in-app flight behaviour is simulated and labelled simulated. Every count, currency, threshold, and buffer is illustrative and marked. The worked example, LV Cruise 2027, Paris, is illustrative; co-brand references demonstrate the system, and no real maison logo or exact palette is claimed.

■ MADE BY KALDUNE