APPOINTFRAME

AppointFrame Architecture & Developer Extension Guide

System boundaries, data flows, invariants, branch topology and safe extension seams

Revised 13 September 2026

Current structured build: Generic reference/product branch and Live deployment branch. Covers schema through Generic migration 035 / Live migration 033.

Handoff objective: a developer unfamiliar with the codebase should be able to trace a feature, identify its owning seam, change it without duplicating business rules, test it, and produce a safe branch-specific deployment patch.

Contents

1. Purpose and product boundary
2. Architecture at a glance
3. Branch topology and branch-specific capability
4. Runtime startup and composition
5. Repository ownership map
6. Admin presentation architecture
7. HTTP and domain/data flow
8. Database facade and migration model
9. Scheduling context: Service + Offering + canonical time
10. Explicit appointment creation and batch semantics
11. Recurring schedule architecture (V2)
12. Group Session architecture
13. Session Package entitlement, reservation and consumption
14. Notifications and side effects
15. External calendar intake
16. Secure client and Staff access
17. Reporting, history and auditability
18. Migration and historical maintenance
19. Security, diagnostics, backups and background services
20. Performance/scannability patterns
21. Extension decision tree
22. Architecture review checklist
23. Appendix A - migration heads
24. Appendix B - Generic WhatsApp Utility defaults

1. Purpose and product boundary

AppointFrame is a modular monolith for administrator-first scheduling and operational continuity in recurring-session businesses. Its durable center is not “a booking page”; it is the relationship between a real appointment, its scheduling context, package entitlement, Staff workload, notifications, notes, reports and historical evidence.

- In scope: availability, appointments, Group Sessions, recurring schedule rules, Session Packages, operational Email/WhatsApp, Session Notes, calendar intake, reporting, migration, audit and narrow role-scoped self-service.
- Out of scope by design: general CRM/sales pipeline, POS/inventory, payroll/accounting, marketplace discovery, broad marketing automation or unrestricted anonymous booking.
- The system is intentionally self-hostable and incremental: Node.js + SQLite + server-composed HTML + vanilla browser modules.

2. Architecture at a glance

```
Browser / secure role surface
 -> views/admin/* + public/assets/app/*
 -> lib/server/*-routes.js (auth, CSRF, request/response)
 -> domain/service seam or LedgerDatabase facade
 -> lib/database/* responsibility modules
 -> SQLite

Background / provider work
 -> ApplicationLifecycle
 -> notification / reminder / notes / calendar / migration services
 -> database + provider
 -> Audit Trail + diagnostic.log
```

The architecture rule is simple: HTTP modules should not become business-rule stores, database modules should not know about Express/browser assets, and browser modules should not invent validation that the server does not recheck.

3. Branch topology and branch-specific capability

Concern	Generic	Live
Product site / Help	Present under product-site/	Not deployed as the Generic product site
Client reschedule	Supported by Generic policy and secure manage route	Blocked by Live policy
WhatsApp appointment reminder	Confirm + Cancel Quick Reply + Reschedule Website URL	Live action contract omits Generic self-reschedule
External calendar integrations	Google verified end-to-end; Microsoft connection/review available when configured	Not present in current Live baseline
Migration head	035 recurring-schedule-v2	033 recurring-schedule-v2 (different historical numbering)
Terminology/defaults	Generic configurable labels/defaults	Customer-specific labels/defaults

Critical branch rule: A filename match is not proof that a whole file is interchangeable. Port shared behavior against each branch latest baseline and preserve branch policy/configuration explicitly.

4. Runtime startup and composition

- app.js / server.js: process hosting and startup.
- ledger-router.js: composition root. It wires middleware, focused route modules, services and the LedgerDatabase facade; do not accumulate feature logic here.
- lib/application-lifecycle.js: owns orderly startup/shutdown of background services and database lifecycle.
- lib/config.js: canonical production environment-variable reader/normalizer. Feature modules should not read process.env directly.
- lib/logger.js: structured technical logging with secret redaction.

5. Repository ownership map

Path	Owns / change here when...
views/admin/	Admin HTML fragments: tabs, settings panels, dialogs.
public/assets/app/features/	Feature-specific browser behavior.
public/assets/app/core/	Shared browser runtime, feature registry, generic helpers only.
public/assets/app/styles/	Source styles. Rebuild public/assets/app.css through lib/admin-styles.js.
lib/server/	HTTP routes, auth/CSRF, request parsing, status/serialization.
lib/database/	Data-oriented domain operations, persistence and business invariants.
lib/database/appointments-groups/	Appointment + Group persistence/domain decomposition.
lib/database/lesson-packages/	Package rules, assignment, progress, finance, lifecycle.
lib/database/migrations/	Immutable numbered schema migrations.
lib/*notifications*.js	Delivery orchestration / event routing.
lib/whatsapp-utility-registry.js	Supported WhatsApp Utility contracts and defaults.
product-site/	Generic-only homepage, Help and documentation assets.
test/	Static/architecture/domain regressions and public method manifests.

6. Admin presentation architecture

Admin HTML is server-composed. Browser modules bind to stable IDs/data attributes after composition. The Appointment workspace now has three first-class tabs: Appointments, Recurring schedules and Saved appointments.

- Appointments remains the operational planner. Day/date headers group rows and the selected week is Sunday-Saturday.
- Recurring schedules is independent management of repeating rules. It is not hidden inside Add appointment.
- Saved appointments is secondary retrieval/history; summary-first/lazy detail must not block initial planner rendering.
- Composer dialogs should preserve mobile readability without page-level horizontal overflow.
- Global dropdown/menu behavior should close on outside click and Escape; avoid local one-off menu implementations.

7. HTTP and domain/data flow

lib/server/*-routes.js owns transport concerns. Server routes call database/domain seams and then trigger side effects such as notifications only after persistence succeeds.

```
Example: explicit appointment batch
POST /api/appointments/bulk
 -> saveAppointmentsBestEffort(inputs)
   -> validateAppointmentBatch(inputs)
     -> normaliseAppointment + availability/capacity/conflict/package validation
   -> saveAppointments(accepted inputs) under BEGIN IMMEDIATE
 -> locate persisted appointments
 -> notification + Staff notification side effects
 -> 201 or 207 with per-row issues
```

8. Database facade and migration model

LedgerDatabase is a facade composed from responsibility modules with duplicate-method/manifest testing. Published migrations are immutable. Add the next branch-specific number; never renumber the other branch to “make them match.”

Generic	Live	Purpose
027	-	Calendar OAuth foundation (Generic only)
028	-	Calendar review/mapping memory (Generic only)
029	027	Location reschedule policy
030	028	Role-scoped secure access

031	029	Secure client booking confirmation
032	030	Recurring schedule plans
033	031	Recurring schedule patterns
034	032	Recurring Group schedules
035	033	Recurring schedule V2

Migration invariant: A deployment database that records a migration unknown to the running code is rejected. Schema and application build must travel together.

9. Scheduling context: Service + Offering + canonical time

```
Service
 + Location / Online Meeting
 = Service Location Offering
   -> availability / rate / currency context
   -> appointment Offering identity
   -> package entitlement
   -> notification / report / export context
```

- Physical scheduling uses the selected Location timezone.
- Online scheduling follows the Organization/online scheduling rule.
- A saved appointment carries one canonical UTC instant plus scheduling-local context; recipient-local rendering must never mutate the appointment instant.
- Client/Staff timezone is a display/communication context, not a replacement for the appointment Location/Meeting.
- Conflict, duplicate and capacity checks use canonical time and exact resource identity.

10. Explicit appointment creation and batch semantics

Add appointment now supports one shared Participant/Service/Location/Staff context with one or many arbitrary date/time selections. These are explicit bookings, not recurrence.

- The composer collects selections; Create N appointments posts them immediately to /api/appointments/bulk.
- saveAppointmentsBestEffort validates rows sequentially/cumulatively and returns accepted rows plus per-row issues.
- Capacity projection includes already saved appointments and accepted rows in the same batch.
- Package validation is cumulative per Package, so a six-lesson Package cannot accept eight future reservations just because zero lessons are completed.
- The database write rechecks resource conflicts and package capacity while holding BEGIN IMMEDIATE.
- Successful new appointments trigger normal client confirmation and Staff assignment side effects. Failed rows do not manufacture notifications.

Reservation vs consumption: Scheduling reserves Package capacity; Package progress/consumption is outcome-driven (Completed always counts; No-show/Cancelled follow organization policy). Do not conflate these states.

11. Recurring schedule architecture (V2)

A recurring schedule is a first-class instruction. It can exist without a seed appointment. Individual and Group plans use separate plan/pattern/occurrence tables but share recurrence-rule semantics.

```
Recurring plan
 + start date
 + repeat unit / interval
 + weekly pattern(s) OR monthly rule
 + ending rule
 -> pattern rows
 -> occurrence ledger (created / skipped)
 -> operational preparation only when the relevant week is processed
```

Aspect	Individual recurring schedule	Group recurring schedule
Identity	Participant + Service + Location/Offering + Staff + linked Package	Service + Location/Offering + Staff; no permanent participants
Default end	When linked Package becomes ineligible	No automatic end
Other end modes	Count or date	Count or date
Weekly	Multiple weekday/time pattern groups; every N weeks	Same rule shape
Monthly	Day-of-month or nth/last weekday	Same rule shape
Operational preparation	listRecurringScheduleDraftsForWeek returns planner drafts; normal Save appointments persists them	prepareRecurringGroupSchedulesForWeek creates empty Group Session shells
Management	Pause/Resume, Edit future, Skip occurrence, End	Pause/Resume, Edit future, Skip occurrence, End

- New schedules cannot start in the past.
- Individual schedule creation resolves an eligible Package against the first matching occurrence.
- Recurring schedules require normal Staff-Service mapping; the one-appointment administrator override is not a permanent recurrence entitlement.
- Occurrence tracking prevents duplicate preparation for the same pattern/date and links existing exact matches when appropriate.
- Edit future anchors the new rule from the effective date and does not rewrite historical appointments/sessions.

12. Group Session architecture

A Group Session is a real administrator-owned session shell. Participants are child appointment records tied to the shell; the empty shell remains meaningful before participants join.

- Add group session can create one or many arbitrary session shells immediately.
- Optional selected participants are attempted for every explicit shell; participant Package eligibility is revalidated per session/date.
- Recurring Group schedules never imply a permanent roster.
- Capacity/minimum rules belong to the Group Session/Service lifecycle, not to the recurrence definition.
- Client self-service may join/leave existing eligible sessions but does not create or reschedule the Group Session.

13. Session Package entitlement, reservation and consumption

- New Packages are one currency and contain exact Service + Location/Meeting Offering entitlements.
- Package holder/payer, weekly-progress recipient and members are separate roles.
- Fixed Packages can carry expiry and complimentary rules; open-ended basis is supported.
- Future scheduled appointments reserve available Package allocation even though progress count changes only when an outcome is counted.
- Completed always counts. No-show and Cancelled consumption are organization policy.
- Historical corrections recalculate effect without moving appointments between cycles or reopening closed cycles.
- Payment Statements use one reviewed/frozen PDF snapshot across selected delivery channels.

14. Notifications and side effects

```
deployment capability
 x organization route
 x recipient preference / eligibility
 x Utility enabled + valid (WhatsApp)
 = effective operational delivery
```

- Explicit appointment creation sends normal creation side effects after persistence.
- Saving a recurrence definition sends no appointment notification because no appointment exists yet.
- Individual recurring drafts notify only when they are actually saved as appointments.
- Recurring Group shell creation follows Group Session lifecycle; participant notification is occurrence-specific.
- Staff reassignment is Unassigned(old) + Assigned(new), not an extra duplicate Updated event.
- Magic-link authentication Email is separate from operational notification capability.

15. External calendar intake

Generic uses a provider-based read-only calendar intake layer. Google is currently verified end-to-end; Microsoft connection/read-only review is available when deployment OAuth configuration is supplied.

- Settings → Integrations owns provider connection state.
- Appointments → Calendar Sync reads current external events, stores review/mapping memory and validates AppointFrame context before import.
- Import creates ordinary AppointFrame appointments only when mapped/validated Ready.
- External event identity links prevent duplicate import.
- Source changes/deletion are surfaced on later review; they do not silently mutate the AppointFrame appointment.

16. Secure client and Staff access

- Administrator credentials remain the only full-privilege application account.
- Secure Participant access is role-scoped and uses verified existing-client identity; eligibility still reuses normal Service/Offering/Package rules.
- Unknown WhatsApp senders do not get identity/availability leakage; they receive assistance guidance.
- Secure Staff access is read-only teaching schedule + Past sessions in Staff-local time; it does not expose admin configuration/reports/packages.
- Session Notes signed links expose the focused notes workflow and require supported token/authentication checks.

17. Reporting, history and auditability

- Saved Appointments, Communication History, Audit Trail and Closed Package Series use summary-first/lazy-detail presentation.
- Participant reports render recipient-relevant times; Staff reports use Staff-local context; Organization reports preserve appointment scheduling semantics.
- Location/Meeting is a filter/context attribute, not a replacement report clock.
- CSV export carries canonical/timezone/Location/Offering identity required for portability.
- Audit Trail is business/operational evidence; diagnostic.log is technical evidence.

18. Migration and historical maintenance

Migration remains staged: Upload → Resolve → Packages → Review/Preflight → protected Commit. Review must not write operational records.

- Commit uses a fresh review fingerprint, backup, transaction and idempotency protection.
- Migration does not send ordinary appointment/Email/WhatsApp notifications.
- Irregular historical package cycles are authoritative and retain actual reconstructed size.
- Historical date/time stays immutable in correction paths; allowed Service/Staff/status repairs must keep package/reporting invariants coherent.
- Unknown historical Location stays unknown when it cannot be established safely.

19. Security, diagnostics, backups and background services

- State-changing browser requests use session/CSRF protection; provider webhooks/Flows use provider verification.
- Never log SMTP passwords, Cloud API tokens, OAuth refresh tokens, private keys, OTPs, cookies or administrator credentials.
- diagnostic.log is rotating technical evidence with secret redaction.
- Backups precede destructive Data/Migration commit paths and are recommended before deployments touching schema/data semantics.
- ApplicationLifecycle owns closable/background services; do not leave timers/scanners unmanaged at module scope.

20. Performance/scannability patterns

- Primary planner loads before large histories.
- Dashboard summary endpoints should aggregate server-side rather than download broad row sets for browser counting.
- History surfaces fetch summary first, detail on expansion.
- Recurring schedules should prepare only operationally relevant occurrences rather than manufacturing months of rows upfront.

- Mobile uses stacked/contained dialogs and avoids page-level horizontal overflow.
- Long menus/dropdowns must close on outside click/Escape and remain within viewport.

21. Extension decision tree

Change requested	Start here	Also check
New admin field	views/admin fragment + owning feature JS	route payload, DB method, migration if persisted, tests
New scheduling rule	lib/database/appointments-groups or availability domain	all entry paths: admin, secure client, copy, recurrence, calendar import
New recurring behavior	recurrence-rules.js + recurring-schedules/group modules	patterns/occurrence ledger, UI preview, tests
New Package rule	lib/database/lesson-packages/*	assignment/progress/finance/reporting + migration/history
New WhatsApp Utility	whatsapp-utility-registry.js + settings	parameter sources, provider send seam, template tests
New Email event	notification event seam + template registry	org/recipient gates, audit, timezone rendering
New report/export field	reporting DB module + route/UI	timezone context + CSV/PDF + regression
New calendar provider	lib/calendar-integrations/providers	OAuth config, read-only review semantics, mapping/import tests
New schema field	next branch migration + focused DB module	manifest/checksum/tests, branch numbering
Historical data repair	staged/temporary fail-closed tool	backup, transaction, audit, remove after use

22. Architecture review checklist

- Correct branch and latest baseline?
- Owning layer chosen; no duplicated business rule in UI/routes?
- Service + exact Location/Meeting Offering identity preserved?
- Canonical instant + recipient timezone semantics preserved?
- Package reservation/consumption/cycle invariants preserved?
- Recurrence change handles plan, patterns, occurrence ledger and already-prepared history correctly?
- Notification side effects happen only after successful persistence and only when capability/routing/recipient allows?
- External calendar remains read-only intake unless an explicit reviewed import occurs?
- Secure role surface cannot escalate into administrator capability?
- Migration/historical action backs up, is transactional and fails closed?
- Responsive behavior and lazy/scannable history patterns preserved?
- Migrations, manifests, generated CSS/assets, cache versions and targeted tests updated?

23. Appendix A - migration heads

Generic	Live	Name / capability
025	025	whatsapp-inbound-action-queue
026	026	package-policy-transfer-and-location-cancellation
027	-	calendar-oauth-foundation (Generic)
028	-	calendar-review-and-mapping-memory (Generic)
029	027	location-reschedule-policy
030	028	role-scoped-secure-access
031	029	secure-client-booking-confirmation
032	030	recurring-schedule-plans
033	031	recurring-schedule-patterns
034	032	recurring-group-schedules

035	033	recurring-schedule-v2

24. Appendix B - Generic WhatsApp Utility defaults

Function	Default Meta template
Appointment reminder	af_appointment_reminder_v3
Group Session reminder	af_group_session_reminder_v3
Participant cancellation	af_appointment_cancelled_client
Staff cancellation	af_appointment_cancelled_staff
Staff assignment	af_appointment_assigned_staff
Staff appointment update	af_appointment_updated_staff
Staff unassignment	af_appointment_unassigned_staff
Daily Staff Session Notes reminder	af_session_notes_reminder_staff
Weekly package progress - fixed	af_package_weekly_progress_fixed
Weekly package progress - open	af_package_weekly_progress_open
Client progress report	af_client_progress_report
Policy review	af_policy_review
Package commencement	af_package_commencement
Package completion	af_package_completion
Package expiry reminder	af_package_expiry_reminder
Package payment reminder (legacy fallback)	af_package_payment_reminder
Package Payment Statement	af_package_payment_statement
Complimentary-session notice	af_package_complimentary_notice
Refund proposal	af_package_refund_proposal
First-session thank-you	af_first_session_thank_you
General announcement	af_general_announcement