Unified Comms + CRM + Jobs Platform GHL - Migration & Build Roadmap

Document Owner: Paddy Yoosin
Created: March 4, 2026
Purpose: Technical + product roadmap to build a GHL-like system that unifies messaging (chat/SMS), calls (Twilio), pipelines, and job dispatch/assignment with strict RBAC and full audit trails. The existing Integration Hub + vendor modules remain part of the system; this roadmap adds the missing “GHL layer” plus frontends.

Table of Contents

1. Executive Summary
   - Current State
   - Target State
   - Scope Summary
2. Migration Scope
   - In Scope
   - Out of Scope
3. Core Domain Model
   - Tenant/Account
   - Users/Roles/Permissions
   - Contacts (Clients / Vendors)
   - Pipelines & Stages
   - Conversations (Chat/SMS)
   - Calls (Twilio)
   - Jobs & Dispatch
   - Associations & Link Tables
   - Activity & Audit
4. Frontend Modules (Portals + Consoles)
   - Internal Console
   - Vendor Portal

    - Client Portal
    - Shared UI Foundations
5. API Endpoints (By Module)
6. Workflows & Business Logic
7. Role-Based Access Control
8. Notification System
9. Data Migration & Backfill Strategy
10. Jira Epics & Story Breakdown
11. Migration Checklist
12. Appendix

- Standard API Response Format
- Event Naming & Webhook Contracts
- Twilio Mapping Cheatsheet

---

1. Executive Summary

Current State

Today, communications and ops data are fragmented:

- **GoHighLevel** handles calls + texts, but threads/jobs are not first-class in Proof360.
- **Gmail/Tawk.to** sit outside the operational record.
- **Airtable/Make** are acting as the workflow glue in Phase 0/1, but aren’t scalable as the long-term system.

Target State

A single platform that provides:

- **Pipelines:** leads → qualified → scheduled → in-progress → completed (custom per team)
- **Unified conversations:** client chat + SMS + Twilio call logs in one “job/record thread”
- **Call ingestion:** Twilio calls recorded/transcribed → attached to conversation thread → can create jobs automatically
- **Jobs + assignment:** job created from inbound communication → vendor assigned → vendor/client/internal all see same job context
- **Strict RBAC:** internal teams, vendors, clients, admins have scoped views and actions

- **Audit + lineage:** every message/call/job change is logged and traceable to user/system actor

Scope Summary (High-Level)

Module	Includes	Priority
Identity & RBAC	tenants, users, roles, permissions, service tokens	High
Pipelines	pipelines, stages, lead/job progression, automation rules	High
Conversations	threads, chat, SMS, attachments, job linkage	High
Calls (Twilio)	inbound/outbound logs, recordings, transcription, linking	High
Jobs & Dispatch	job creation, assignment, statuses, SLA, notes/photos/docs	High
Associations	client↔job, vendor↔job, vendor↔client, pipeline↔job	High
Portals	internal console + vendor portal + client portal	High
Analytics/Reporting	KPIs, funnel conversion, response time, vendor performance	Medium
AI Copilot	summaries, missing-step detection, next action, PDIL alignment	Medium

---

2. Migration Scope

2.1 In Scope

CRM + Pipelines

- Pipeline definitions per tenant/team
- Stages, stage transitions, reasons, timestamps
- Pipeline entities: Leads, Opportunities, Jobs (depending on your model)

- Basic automations: stage change triggers job creation / vendor assignment suggestion

Unified Messaging

- Threaded conversations tied to **Contact + Job + Pipeline entity**
- Channel types: in-app chat, SMS (Twilio), email later (optional)
- File uploads: photos/docs
- Search: by contact, job ID, phone, keywords
- Internal notes vs client-visible messages

Calls + Tracking (Twilio)

- Twilio number management per tenant/location
- Webhooks ingestion (call start/end, recording, transcription if enabled)
- Call log objects linked to contact + conversation + job
- Automatic job creation rules (configurable)
- Missed call flows: create task + notify CS + “call back” workflow

Jobs + Vendor Assignment

- Job creation from:
  - inbound call
  - inbound chat/SMS
  - pipeline stage change
  - manual internal creation
- Vendor selection + assignment + reassignment
- Job lifecycle statuses aligned with dispatch logic
- Job artifacts: notes, photos, documents, checklists, SLA timestamps

Associations & Data Lineage

- Vendor↔Client relationship model
- Vendor↔Job, Client↔Job, Job↔Conversation, Conversation↔Call
- Audit events for every mutation

Frontend (Explicitly In Scope)

- Internal ops console (CS/dispatch/admin)
- Vendor portal (mobile-first job execution + comms)

- Client portal (chat + job history + docs + billing later)

2.2 Out of Scope (for this phase)

- Payments + invoicing engine (can integrate later)
- Full email replacement (Gmail continues initially)
- Marketing funnels/landing page builder (GHL-style page builder) unless you explicitly want it
- Loyalty/referral tooling (belongs to separate growth/monetization track)

---

3. Core Domain Model

Note: This is the canonical model you'll implement first; everything else is UI or workflow on top.

3.1 Tenant / Account

Represents an org, region, brand, or franchise unit.

Collection/Table: `tenants`

- `tenant_id` (unique)
- `name`
- `default_timezone`
- `twilio_account_sid` (optional, if per tenant)
- `settings` (pipeline defaults, job rules, etc.)

Indexes:

- `tenant_id` unique
- `name` text

3.2 Users / Roles / Permissions

Collections: `users`, `roles`, `permissions`, `user_roles`

- Users belong to a tenant

- Roles grant permissions (resource + action)
- Support service-to-service tokens for legacy components (Integration Hub/global-node)

Key permission domains:

- `contacts:*`
- `pipelines:*`
- `conversations:*`
- `calls:*`
- `jobs:*`
- `vendors:*`
- `admin:*`

3.3 Contacts (Client / Vendor)

Collection: `contacts`

- `contact_id` (unique)
- `type` enum: `client | vendor | internal | mixed`
- `name`, `phone_e164`, `email`
- `addresses[]`
- `tags[]`
- `preferred_channel`
- `created_at`, `updated_at`

Indexes:

- `tenant_id + phone_e164` unique (or near-unique with merge handling)
- text index on `name`, `email`

3.4 Vendor Profile

Collection: `vendors`

- `vendor_id` (unique)
- `contact_id` (ref)
- compliance fields (or link to compliance module you already defined)
- service categories, service area, availability, rates summary

(You can reuse/bridge your existing vendor entry/compliance models and just add tenant-scoping + linkage.)

3.5 Pipelines & Stages

Collections: `pipelines`, `pipeline_stages`, `pipeline_items`

- `pipeline_items` can reference `lead_id` or `job_id` depending on strategy
- store stage history as events (don’t just overwrite)

3.6 Conversations (Chat/SMS Threads)

Collection: `conversations`

- `conversation_id`
- `tenant_id`
- `contact_id` (client primary)
- `job_id` (nullable until created)
- `channel_mix`: `in_app`, `sms`, later `email`
- `last_message_at`
- `status`: `open | pending | closed`
- `participants`: internal user IDs, vendor IDs, etc. (scoped by RBAC)

Collection: `messages`

- `message_id`
- `conversation_id`
- `direction`: `inbound | outbound | internal_note`
- `channel`: `in_app | sms`
- `body`
- `attachments[]`
- `author`: user/vendor/system
- `created_at`

Indexes:

- `conversation_id + created_at`
- search index on `body`

3.7 Calls (Twilio)

Collection: `calls`

- `call_id`
- `tenant_id`
- `from_e164`, `to_e164`
- `twilio_call_sid` (unique)
- `status`: ringing/completed/missed/voicemail
- `started_at`, `ended_at`, `duration_seconds`
- `recording_url` (if enabled)
- `transcript` (if enabled)
- `linked_contact_id`
- `linked_conversation_id`
- `linked_job_id`
- `disposition` (optional): e.g., booked, follow-up, spam

Indexes:

- `twilio_call_sid` unique
- `from_e164`, `to_e164`, `started_at`

3.8 Jobs & Dispatch

Collection: `jobs`

- `job_id` (human readable + internal id)
- `tenant_id`
- `client_contact_id`
- `vendor_id` (nullable pre-assignment)
- `status`: `new | triaged | assigned | in_progress | pending_review | completed | canceled`
- `priority`, `sla_due_at`
- `source`: `call | sms | chat | manual | pipeline`
- `summary`, `notes`
- `location`
- `created_at`, `updated_at`

Collection: `job_events`

- append-only event log (status changes, assignment changes, notes added, etc.)

3.9 Associations

You'll want explicit link tables/collections to avoid ambiguity:

- `vendor_clients` (vendor_id, client_contact_id, relationship_type)
- `job_vendors` (job_id, vendor_id, role, assigned_at)
- `conversation_jobs` (conversation_id, job_id) if you allow many-to-many

3.10 Activity & Audit

Collection: `audit_events`

- `event_id`
- `tenant_id`
- `actor` (user/vendor/system)
- `resource_type`, `resource_id`
- `action`
- `before`, `after` (diff-friendly)
- `timestamp`

---

4. Frontend Modules

This is the piece your current migration doc doesn't include — here it's first-class, aligned with your "UI/UX & Front-End Lead" expectations.

4.1 Internal Console (CS / Dispatch / Admin)

Key screens

- Inbox (unified): conversations + missed calls + open jobs
- Conversation detail: timeline (SMS/chat), call logs, job panel, quick actions
- Pipeline board: kanban stages, drag/drop (with permission + logging)
- Jobs list + job detail: status, assignment, checklist, docs/photos
- Vendor directory: search, compliance snapshot, assign action
- Client directory: history, active jobs, conversation history
- Admin: roles/permissions, Twilio numbers, routing rules, templates

Frontend tasks

- Component library / design system (buttons, tags, badges, status pills)
- Realtime updates (SSE/WebSocket) for inbox + thread messages
- Powerful filtering + saved views
- Audit view (read-only) for admins

4.2 Vendor Portal (Mobile-first)

Key screens

- Assigned jobs list
- Job detail: steps/checklist, upload photos/docs, add notes
- Vendor ↔ internal messaging thread (and optionally vendor ↔ client if allowed)
- Availability + service area management
- Compliance upload/update (if you want vendor self-serve)

Frontend tasks

- Mobile-first UX, offline-tolerant uploads, retry queue
- Push-style notifications (web push or in-app)

4.3 Client Portal (Chat-first)

Key screens

- “Message us” thread (primary)
- Job status tracker + appointment times
- Job history
- Documents/photos shared
- (Later) billing/funding views

Frontend tasks

- Clean chat UI with attachment support
- Explicit separation: “messages” vs “internal notes” (client never sees notes)

4.4 Shared UI Foundations

- Auth integration (Memberstack or native auth later)
- RBAC-aware routing + component gating
- Telemetry: page performance, action tracking, error tracking

- Accessibility baseline: contrast, keyboard nav, touch targets

---

5. API Endpoints (By Module)

(Names are examples; adapt to your Integration Hub conventions.)

5.1 Auth & RBAC

- `POST /api/v1/auth/login`
- `GET /api/v1/me`
- `GET /api/v1/roles`
- `POST /api/v1/roles`
- `PUT /api/v1/roles/:id`
- `GET /api/v1/permissions`
- `POST /api/v1/service-tokens` (for global-node / legacy)

5.2 Contacts

- `GET /api/v1/contacts`
- `POST /api/v1/contacts`
- `GET /api/v1/contacts/:id`
- `PUT /api/v1/contacts/:id`
- `POST /api/v1/contacts/merge`
- `GET /api/v1/contacts/:id/history` (jobs + conversations + calls)

5.3 Pipelines

- `GET /api/v1/pipelines`
- `POST /api/v1/pipelines`
- `PUT /api/v1/pipelines/:id`
- `GET /api/v1/pipelines/:id/stages`
- `PUT /api/v1/pipeline-items/:id/move` (stage transition)
- `GET /api/v1/pipeline-items` (filterable)

5.4 Conversations & Messages

- `GET /api/v1/conversations`
- `POST /api/v1/conversations` (create thread)
- `GET /api/v1/conversations/:id`
- `GET /api/v1/conversations/:id/messages`
- `POST /api/v1/conversations/:id/messages` (send in-app)
- `POST /api/v1/conversations/:id/sms` (send SMS via Twilio)
- `POST /api/v1/uploads` (attachments)
- `GET /api/v1/search` (cross-entity search)

5.5 Calls (Twilio)

- `POST /api/v1/twilio/webhooks/call-status`
- `POST /api/v1/twilio/webhooks/recording`
- `POST /api/v1/twilio/webhooks/transcription` (if used)
- `GET /api/v1/calls`
- `GET /api/v1/calls/:id`
- `POST /api/v1/calls/:id/link` (link to job/conversation)
- `POST /api/v1/calls/:id/disposition`

5.6 Jobs

- `GET /api/v1/jobs`
- `POST /api/v1/jobs`
- `GET /api/v1/jobs/:id`
- `PUT /api/v1/jobs/:id`
- `POST /api/v1/jobs/:id/assign-vendor`
- `POST /api/v1/jobs/:id/unassign-vendor`
- `POST /api/v1/jobs/:id/events` (append event)
- `POST /api/v1/jobs/:id/checklist`
- `POST /api/v1/jobs/:id/complete`

5.7 Associations

- `POST /api/v1/vendor-clients` (create relationship)
- `GET /api/v1/vendor-clients` (query)

- `DELETE /api/v1/vendor-clients/:id`

---

6. Workflows & Business Logic

6.1 Inbound Call → Thread → Job

1. Twilio webhook receives call status
2. System resolves/creates Contact by `from_e164`
3. System finds/creates “Client conversation thread”
4. Call log attached to thread
5. If call missed OR “booking intent” detected (rule-based first, AI later), create Job
6. Notify internal inbox + optionally auto-assign vendor (rule-based)

6.2 Inbound SMS/Chat → Same Thread

- All inbound messages route into the same conversation record
- Thread can be linked to an existing job, or can spawn a job if “new work” intent is detected

6.3 Pipeline Stage Change → Job Lifecycle

- Example: pipeline item moved to “Scheduled” triggers job creation + schedule fields
- Moving to “In Progress” marks job assigned/in_progress

6.4 Vendor Assignment Rules

- Basic rules first: service_type match + region + availability + compliance status
- Later: performance scoring + response time + workload balancing

6.5 Audit & Compliance

- Every status change, assignment change, message send, and call link is an audit event
- Admin can reconstruct exactly “who did what, when”

---

7. Role-Based Access Control

Start with these role families (expand later):

- **Client roles:** view own jobs + own conversation only
- **Vendor roles:** view assigned jobs + vendor portal threads; never see unrelated clients
- **CS roles:** full comms inbox, create jobs, manage clients
- **Dispatch roles:** vendor assignment, job state control
- **Admin roles:** manage roles/permissions, Twilio routing, templates
- **Developer/service:** integration tokens, diagnostics

RBAC rules must apply at:

- API level (middleware)
- Query level (tenant scoping + relationship scoping)
- UI level (hide actions user can’t do, but never rely on UI-only security)

---

8. Notification System

In-app (Realtime)

- New inbound message

- Missed call
- Job created
- Vendor assigned / reassigned
- SLA breach approaching

Email/SMS (Optional)

- Client: “We received your request”
- Vendor: “New job assigned”
- Internal: daily digest of missed calls, overdue jobs

Implementation strategy:

- Publish internal events to an event bus (or queue)
- Notification service consumes events and fans out to channels

---

9. Data Migration & Backfill Strategy

Reality check: you will not get a clean cutover from GoHighLevel on day one.

Phased approach:

1. **Mirror mode:** ingest Twilio events + messages while GHL continues to send/receive
2. **Thread linking:** backfill past call logs (if accessible) into `calls` table and link to contacts
3. **Gradual cutover:** switch outbound SMS/calls initiation to your system while keeping GHL as fallback
4. **Decommission:** only after parity (inbox, search, templates, routing, reporting)

---

10. Jira Epics & Story Breakdown

EPIC A — Platform Foundations (Tenant, Auth, RBAC)

- A1: Tenant model + service tokens
- A2: Users/roles/permissions framework
- A3: Audit events framework (append-only)

EPIC B — Contacts + Identity Resolution

- B1: Contacts CRUD + phone normalization (E.164)
- B2: Contact merge + dedupe
- B3: Contact history API

EPIC C — Conversations (Chat/SMS)

- C1: Conversation + message models
- C2: In-app messaging APIs
- C3: SMS send/receive via Twilio
- C4: Attachments + storage
- C5: Search

EPIC D — Calls (Twilio)

- D1: Twilio webhooks ingestion
- D2: Call record linking to conversation/contact/job
- D3: Recording/transcription ingestion
- D4: Missed call workflow + tasks

EPIC E — Pipelines

- E1: Pipelines/stages setup
- E2: Pipeline items + stage transitions
- E3: Automation hooks (stage → job)

EPIC F — Jobs & Dispatch

- F1: Job model + status lifecycle
- F2: Vendor assignment engine (rules v1)
- F3: Job checklist + artifacts
- F4: Vendor/client association model

EPIC G — Frontend: Internal Console

- G1: Inbox UI (threads + missed calls + jobs)
- G2: Conversation detail UI (timeline + quick actions)
- G3: Pipeline board UI
- G4: Job detail UI
- G5: Admin settings UI (roles + twilio numbers + rules)

EPIC H — Frontend: Vendor Portal (Mobile-first)

- H1: Assigned jobs list + job detail
- H2: Upload photos/docs + checklist completion
- H3: Messaging panel
- H4: Notifications UX

EPIC I — Frontend: Client Portal (Chat-first)

- I1: Client auth + “Message us” thread
- I2: Job status tracker
- I3: Job history + shared docs

EPIC J — AI Copilot (Later but planned)

- J1: Thread summarization
- J2: “Create job from call” suggestion (human-in-the-loop)
- J3: Missing info detection + next best action

This matches the expectation that UI/UX + frontend is a first-class workstream, not an afterthought.

---

11. Migration Checklist

Pre-Migration

- Decide tenant boundaries (city/brand/franchise)
- Choose initial RBAC role set + permissions matrix
- Lock event naming conventions + audit requirements

Phase 1: Foundations + Ingestion

- Twilio webhook ingestion live
- Contact resolution working
- Conversations + calls stored and visible internally

Phase 2: Operational Workflows

- Job creation from call/chat
- Vendor assignment flow working
- Vendor portal supports job completion + uploads

Phase 3: Full Replacement Targets

- Outbound SMS in your system
- Inbound routing rules managed by admin UI
- Reporting dashboards for response times + conversion

Post-Migration

- GHL reduced to “legacy archive” then removed
- Formal SLOs + monitoring + on-call runbooks

If you want, I can also generate a **second version** that’s **time-phased (Week 1 → Week 16)** and explicitly shows “Phase 0 (keep GHL) → Phase 1 (mirror) → Phase 2 (partial cutover) → Phase 3 (replace)” so it’s easier to execute with your team cadence and the current UEP plan.

Revision #1
Created 5 March 2026 02:06:56 by Paddy Yoosin
Updated 5 March 2026 02:08:44 by Paddy Yoosin