HyperLocal 2: CRM-Native Conversational Assistant

Technical Whitepaper

Gateway Corporate LLC

April 2026

Contents

HyperLocal 2 Whitepaper

Executive Summary

HyperLocal 2 is NashTwin's second-generation architecture for tenant-governed conversational CRM operations over SMS. In the current implementation, it is not a generic chatbot bolted onto a phone number. It is a routed, policy-aware messaging runtime that binds AI behavior to CRM context, operator-owned agent profiles, explicit permissions, curated knowledge sources, closeout rules, and subscription-aware operating limits.

The practical outcome is a system that can do more than answer inbound texts. HyperLocal 2 can:

- route inbound SMS to tenant-specific agent profiles
- enrich replies with CRM context, prompt examples, and approved knowledge sources
- create or update CRM contacts when explicitly allowed
- create, move, or delete CRM calendar events when explicitly allowed
- maintain conversation records, transcript artifacts, and closeout scheduling
- enforce rate limits, route-level transport settings, and operator-visible workspace controls

At its core, HyperLocal 2 is an opinionated message operations platform. It treats messaging as a governed business workflow rather than a pure language-model interaction.

1. Why HyperLocal 2 Exists

Traditional business messaging systems split into two weak patterns.

The first pattern is the human inbox. It is flexible, but expensive, inconsistent, and difficult to scale. The second pattern is the automation tree. It is cheap, but brittle, context-poor, and usually disconnected from the CRM system that the rest of the business actually operates.

HyperLocal 2 is designed as a third model: a conversational operations layer. It keeps the fluidity of freeform SMS, but anchors every decision in tenant-owned configuration, CRM state, explicit permissions, and transport-level controls.

The code reviewed for this whitepaper shows that this is already the implementation direction, not just a product aspiration. The runtime couples AI completion, route configuration, Twilio delivery, CRM record linkage, knowledge retrieval, and closeout handling inside a single tenant-scoped service boundary.

2. Design Principles

HyperLocal 2 is defined by six design principles.

2.1 Tenant First

Every meaningful record in the messaging system is tenant-scoped. Agent profiles, knowledge sources, routes, conversations, prompt examples, and closeout jobs are

all modeled as tenant-owned records. This keeps the system aligned with NashTwin's operating model and prevents conversational behavior from drifting into global, shared-state logic.

2.2 CRM-Native, Not CRM-Adjacent

The runtime does not treat CRM as a passive export destination. Conversations can link to contacts and deals. Agent actions can create or update CRM contacts. Calendar operations can create or change CRM events. Outbound and inbound messaging are stored as first-class operational records instead of disposable transport logs.

2.3 Permissioned Agency

HyperLocal 2 does not assume that every agent should be able to mutate business records. Agent profiles carry explicit capabilities such as contact management and calendar management. Those toggles are enforced in the orchestration layer, not just suggested in UI copy.

2.4 Curated Context

The platform supports two kinds of context enrichment.

- operator-authored prompt examples and agent directives
- tenant-curated knowledge sources, currently implemented as synced webpages

This matters because it shifts answer quality away from prompt luck and toward managed operational knowledge.

2.5 Reliable Transport, Bounded Automation

The route layer binds Twilio credentials, inbound numbers, webhook keys, fallback agents, and policy settings to each messaging line. Rate limiting, closeout windows, and completion settings are route-aware, giving operators a way to tune behavior without turning the system into an unmanaged black box.

2.6 Useful Failure Modes

The system includes deterministic fallbacks. If AI completion is unavailable, the runtime still produces a bounded business-safe reply. If an agent lacks permission to mutate CRM records, the system returns escalation-oriented guidance rather than silently failing or overreaching.

3. System Architecture

HyperLocal 2, as implemented today, can be understood as five cooperating layers.

3.1 Agent Layer

An agent profile defines the operational identity of a messaging assistant. Each profile includes:

- name and business identity
- business address and locality

- personality and directives
- fallback human contact
- optional contact list for reference
- capability flags for CRM contact and calendar mutation
- linked prompt examples
- linked knowledge sources

This turns an agent into a governed operating profile rather than a raw prompt.

3.2 Route Layer

A route binds the runtime to a public inbound number. Routes store:

- primary agent assignment
- optional fallback agent
- inbound number and normalized variants
- unique webhook key
- Twilio account SID
- encrypted Twilio auth token
- optional messaging service SID

The Messaging workspace exposes these routes to operators and presents the route webhook as an absolute POST URL. This reinforces that transport configuration is part of the product surface, not hidden infrastructure.

3.3 Conversation Layer

Each thread persists as a conversation record with linked messages, timestamps, closeout deadlines, optional transcript summary text, optional transcript body, and optional CRM linkage to a contact or deal. Inbound and outbound messages are stored as distinct records, which makes the transcript and closeout system auditable and replayable.

3.4 Orchestration Layer

The orchestration layer receives inbound SMS, resolves a route, assembles context, evaluates rate limits, optionally invokes CRM sidecars, generates a reply, sends the outbound SMS, records activity, and schedules closeout work. This is the real heart of HyperLocal 2.

3.5 Governance Layer

The platform already includes subscription gating, prompt budget enforcement, knowledge-source limits, agent limits, transcript closeout scheduling, and a broader manifest surface for retention and policy operations. Even where some policy surfaces extend beyond the newest code additions, the direction is clear: HyperLocal is being built as a governed communication subsystem.

4. The Runtime Flow

The inbound SMS runtime follows a clear sequence.

1. A public webhook receives a form-encoded Twilio request.
2. The system validates request shape and checks for the Twilio signature header.
3. The route is resolved by webhook key.
4. The tenant install and HyperLocal subscription entitlement are enforced.
5. The runtime loads the route's agent, prompt examples, and linked knowledge sources.
6. The system attempts CRM-aware contact handling when appropriate.
7. The system loads or creates a conversation and stores the inbound message.
8. Rate-limit policy is evaluated against recent inbound traffic.
9. Context is assembled for reply generation.
10. The assistant reply is generated, sent through Twilio, and stored.
11. A closeout job is scheduled for inactivity-based transcript handling.

This matters because the AI step is only one segment of the workflow. HyperLocal 2 is fundamentally a transaction pipeline for conversational business operations.

5. AI Runtime and Model Strategy

The current implementation uses OpenAI chat completions with `gpt-4.1-mini` as the default model. The completion adapter supports:

- configurable retries
- configurable timeouts
- temperature control by task type
- a deterministic fallback reply if AI is unavailable

This is important for two reasons.

First, the runtime uses AI in multiple specialized modes rather than one generic completion call. It uses low-temperature classification for tasks like query generation or action planning, and a more standard completion setting for final customer-facing responses.

Second, HyperLocal 2 is designed to degrade safely. If `OPENAI_API_KEY` is unavailable or the completion path fails, the system falls back to a simple business-safe response promising follow-up. That is a stronger operational posture than treating LLM uptime as a hard dependency.

The implementation also uses Google Places as a recommendation-oriented knowledge adapter when a Places key is configured. In practice, that lets the agent support localized recommendation workflows without confusing external search with tenant-authored knowledge.

6. Contact and Calendar Action Plane

One of the most important additions in the reviewed code is the transition from passive messaging to permissioned business action.

6.1 Contact Management

When contact management is enabled on an agent profile, HyperLocal can classify inbound SMS messages as requests to create or update CRM contact records. The system does not do this blindly.

The implementation shows several safeguards:

- contact actions are only attempted when the message matches a contact-oriented intent pattern
- the owning CRM actor context is resolved from the agent's membership
- updates try to resolve a target contact by ID, email, phone, name, or linked conversation context
- new contact creation is blocked until the customer provides an email address

That email-first requirement is especially significant. It shows HyperLocal 2 moving toward structured CRM hygiene, not just opportunistic lead capture.

6.2 Calendar Management

Calendar handling follows the same architectural model. When enabled, an agent can interpret SMS requests into CRM calendar actions. When disabled, the system still understands the request contextually but explicitly refuses mutation and instructs operators to escalate.

This symmetry between contact and calendar handling is one of the strongest signals that HyperLocal 2 is evolving toward a reusable action-orchestration framework.

7. Knowledge Sources as an Operator-Controlled Memory Layer

The most important new context feature in the current implementation is the introduction of messaging knowledge sources.

These knowledge sources are tenant-managed records, currently backed by webpages. Operators can:

- register a source with title and URL
- normalize and store the source URL
- sync fetched page text into the workspace
- view source status, excerpt, hostname, sync time, and errors
- assign sources to specific agents

At runtime, HyperLocal selects relevant sources with a simple and transparent scoring model based on token overlap between the user message and the synced source text. The selected excerpts are then injected into the prompt context with explicit instructions to ground the answer in approved material.

This is strategically important. HyperLocal 2 does not depend on opaque vector infrastructure to begin becoming knowledge-aware. It starts with a tenant-visible, operator-auditable, low-complexity knowledge layer that can evolve over time.

8. Operator Experience and Control Surface

The Messaging workspace is not a cosmetic admin page. It is the operational control plane for the system.

The reviewed UI shows several important product decisions:

- operators create and edit agent profiles directly in the workspace
- knowledge sources are visible, syncable, and assignable in-place
- route configuration is exposed as a first-class workflow
- Twilio auth tokens are masked rather than displayed
- replacement tokens are separated from currently stored secrets
- webhook URLs are rendered as absolute endpoints with explicit POST labeling

This is what makes HyperLocal 2 viable for real operators. The system is not asking users to trust invisible logic. It exposes the operational levers directly where messaging is administered.

9. Packaging, Limits, and Commercial Strategy

The code also reveals an early commercial segmentation strategy.

In HyperLocal Pro:

- a tenant may create up to 1 agent profile
- a tenant may create up to 5 knowledge sources
- published prompt examples are constrained by a 3,200 token budget

In HyperLocal Enterprise:

- additional agents are permitted
- additional knowledge sources are permitted

This is a strong packaging model because it prices operational scale, not just message volume. It treats richer configuration and broader context coverage as enterprise capabilities.

10. Data Model and Operational Records

The schema behind HyperLocal 2 is already mature enough to support a serious product.

Key records include:

- `MessagingAgentProfile`
- `MessagingKnowledgeSource`
- `MessagingAgentKnowledgeSource`
- `MessagingPromptExample`
- `MessagingRoute`
- `MessagingConversation`
- `MessagingMessage`
- `MessagingCloseoutJob`

Together these records support identity, transport, context, retention-adjacent closeout behavior, and conversational evidence. This is not the schema of a toy chatbot. It is the schema of an auditable messaging subsystem.

11. What Is Implemented Today Versus What the Platform Surface Signals

The implementation reviewed for this whitepaper is strongest in SMS, CRM orchestration, knowledge-source grounding, and route governance.

The manifest and action surface also indicate a broader HyperLocal trajectory that includes:

- voice routing
- missed-call follow-up
- policy and retention workspaces
- transcript governance
- graph-aware evidence artifacts and lineage

The important distinction is this: HyperLocal 2 is already credible as an SMS operations runtime today, and its manifest indicates a path to become a broader governed communications platform without changing its architectural center of gravity.

12. Strategic Interpretation

HyperLocal 2 should be understood as the beginning of a local-business operating layer for conversational workflows.

Its core insight is that messaging quality is not primarily a model-selection problem. It is a systems problem. Good business messaging requires:

- the right identity
- the right business context
- the right CRM permissions
- the right transport settings
- the right fallbacks
- the right knowledge
- the right closeout behavior

The current codebase reflects exactly that philosophy.

Conclusion

HyperLocal 2 is not merely “AI texting for businesses.” It is a tenant-governed CRM messaging runtime that turns inbound and outbound SMS into structured operational work.

The code reviewed for this whitepaper shows a coherent architecture already in motion:

- OpenAI-backed completion with deterministic fallback
- Twilio-backed route delivery with masked secret handling

- operator-owned agent profiles and prompt examples
- permissioned contact and calendar mutation
- synced webpage knowledge sources for grounded replies
- tenant-scoped conversations, messages, and closeout jobs
- subscription-aware limits that define the product tiers

If HyperLocal 1 was about adding conversational capability to the platform, HyperLocal 2 is about operationalizing that capability so it can be trusted, administered, and expanded.

That is the right foundation for a messaging system that intends to sit inside real business workflows rather than beside them.