WHITEPAPER

Broad Spectrum Memory: Why Agents Need to Hear Every Channel, Not Just One

Wappari

By Arno Hart, Founder, Wappari

The Problem Starts With the Channel

Every piece of customer communication happens inside a channel, and every channel is narrow spectrum. A voice call is a channel with its own shape: a waveform, a start time, an end time, maybe a transcript if you're lucky. A chat message is a different shape entirely: short, text-native, timestamped turn by turn. A voice note, a meeting recording, a document, an email. Each one is its own self-contained slice, built for a single purpose, with its own format, its own storage system, and its own way of being read.

It's tempting to assume this is only a problem for agents, and that a human doing the same job wouldn't struggle. In practice, it's difficult for human beings too. A lot of these systems purport to give a person a single view across everything, call, chat, email, notes, all lined up together on one screen. But having the view isn't the same as having the time. Nobody actually sits down and listens to the full call recording, plays back every voice note in full, or reads through the twelve documents a customer submitted. Even a human ends up doing a cursory pass and leaning on footnotes, comments, and metadata to get the gist: the same low-fidelity shortcut an agent is forced into. So this isn't purely an AI problem; it affects anyone who has to make sense of a fragmented, multichannel record.

It's especially acute for agents, though, because an agent doesn't have the judgment a human brings to skimming. The moment you put an agent, AI or otherwise, behind that same conversation, the narrow-spectrum nature of each channel becomes a hard architectural problem rather than a soft human one. The channels sit side by side. They're never actually joined. And an agent that can only read one shape at a time is an agent that only ever sees a fragment of what's really going on.

Two Ways Enterprises Try to Solve This, and Why Both Fall Short

There are, broadly, two ways enterprises currently try to give agents access to this fragmented world.

The first is a single back-end agent (or small set of them) that queries across channels as needed. It reaches into the call system, the CRM, the email server, the document store, and tries to glean what it can. The problem is that every one of those systems has its own shape, and most of them are expensive and awkward to consume: a voice recording isn't a text file, a scanned document isn't a chat log. So what actually comes back to the agent isn't the conversation at all. It's metadata: a CRM field, a call summary, a subject line, a note someone typed after the fact. Two commercial products show this pattern plainly: Twilio's Conversation Memory[1] (which extracts observations, summaries, and traits from

each channel and reconciles them into a customer profile) and Zoho's Zia Conversation Summary[2] (which reviews an agent's recent interactions per channel and hands back deadlines, sentiment, and friction points, explicitly so nobody has to read the full transcripts). Both genuinely reach across channels. Both hand back a distillation of the conversation, not the conversation itself. The fidelity of what comes back is very low, and the cost of getting even that much is high.

The second approach is to build a dedicated agent for each channel's shape. A voice agent that understands call recordings. A chat agent for digital messages. A speech agent that diarizes and summarizes meetings, but only for meetings, and only inside that one system. A document agent that extracts data from files. A mail agent for email. Each one gets genuinely good at its own narrow slice, and a lot of the industry's real sophistication is spent making each specialist better at exactly that. This is where tooling like LangChain's memory types[3] (buffer, summary, windowed) and Oracle's hybrid layered memory pattern[4] (sliding window plus summarization plus vector retrieval plus structured and episodic memory) actually lives: not as an attempt to unify channels, but as an investment in handling longer conversations well within one of them. A chat agent gets better at remembering a long chat. A voice agent, in principle, could get better at remembering a long call history. But that's depth added to a single shape, not breadth across shapes, and it doesn't change the fact that each of these agents also lives inside its own department, with its own budget, its own roadmap, and its own pace of development. They're fiefdoms, some of them now quite sophisticated fiefdoms. Nothing shares information between them, so even though every channel now has a capable, and possibly quite advanced, reader, no single view of the customer ever comes together. There's still no broad spectrum memory anywhere in the enterprise: not in the extraction agent, and not in the collection of siloed specialists either, no matter how well any one of them manages its own long conversations.

Either way, the fidelity of what an agent can actually work with stays low, and the cost of getting it stays high.

The Broad Spectrum Idea: Bond the Slices, Not the Silos

The way out isn't to build a smarter reader for each shape. It's to stop having different shapes in the first place.

That's the broad spectrum idea: bond the narrow-spectrum slices into one broad spectrum record, and reduce every channel, call, chat, voice note, meeting, document, email, to the same shape. In practice, that means resolving every channel to the same speaker-attributed structure, customer, agent, customer, agent, using whatever signal already carries that attribution (a stereo call, a chat's sender ID, a meeting's per-participant session), and reaching for true diarization only where nothing else provides it, like a single-track recording. A call becomes that shape. A chat becomes that shape. A meeting recording, a document, an email. All reduced to the same speaker-attributed text. Once every channel produces the same output format, the cost and friction of consuming it collapses. An agent doesn't need six specialized readers anymore. It needs one prompt, one structure, and it can analyze the entire brand, every channel, every touchpoint, the same way, at a fraction of the cost.

This is also, not coincidentally, the shape a language model already reads best. Speaker-attributed text is the lowest common denominator between "what actually happened in a conversation" and "what a

system prompt is built to expect." Nothing about the original signal gets summarized away in the process: every channel still shows up in full, just in a form that's finally consumable at scale.

This is the approach Wappari's own architecture already runs on. More on that below.

What Broad Spectrum Memory Actually Changes

Once every channel lands in the same shape, you don't just get consumption efficiency. You get something qualitatively different: one continuous memory that spans the entire customer relationship, with high fidelity, at low cost.

A few places where that shows up directly:

- **Deal management.** A live, accurate read on where a deal actually stands, because the record isn't reconstructed from fragments. It's the whole conversation, across every channel it touched.
- **Support handoff.** The next agent to pick up a case, human or AI, starts with the full context of everything that came before, instead of piecing it together from call notes and a subject line.
- **Omnichannel response.** Whoever answers next, on whatever channel, email, SMS, voice, a human rep or an AI agent, is replying from the same complete record, not a partial one specific to that channel.
- **Cross-vertical application.** The same principle holds in legal, in financial services, in any domain where the accuracy of a response depends on how much of the actual conversation the responder can see.

In every one of these cases, the pattern is the same: efficiency goes up, accuracy goes up, and the cost of getting there goes down, because you're no longer paying to maintain six different specialized readers, or accepting the low-fidelity output of a system that only ever sees metadata.

Not a Concept: This Is What Wappari Is Built On

It's worth being precise about one thing: broad spectrum memory isn't a hypothesis or something on a future roadmap. It's what Wappari's chat-native architecture already produces today. It's the reason the company exists.

That architecture started on the WhatsApp Business API, deliberately, because WhatsApp already natively supports a wide range of shapes inside a single thread: text, voice notes, images, documents, video. Building on that foundation meant multiple channel shapes were already living together from day one, instead of having to bolt disparate systems together after the fact. From there, the architecture has expanded outward: SIP phone calls, meeting recordings, contact-center call recordings, sales calls, video meetings, and more have all since been folded into the same broad spectrum memory.

So the normalized, one-shape-per-conversation record described above isn't a theoretical end state. It's the working architecture underneath Wappari today, and it keeps growing to cover more of the channels a real business actually runs on.

The Core Idea

This is what broad spectrum memory means at Wappari, and it's the thing the company was built around: one conversation, every channel. Not a call record here and a chat log there and an email thread somewhere else, each requiring its own agent to interpret, but a single, continuous, high-fidelity memory of the entire relationship, in a shape that's cheap to read and impossible to fragment.

An agent is only ever as good as the memory it can read. Broad spectrum memory is what makes sure that memory is actually whole.

References

[1] Twilio. “Conversation Memory.” Twilio. https://www.twilio.com/en-us/products/conversational-ai/conversation-memory

[2] Zoho. “Zia Conversation Summary: Context at a Glance for Every Customer Interaction.” Zoho Community / Help. https://help.zoho.com/portal/en/community/topic/zia-conversation-summary-context-at-a-glance-for-every-customer-interaction

[3] Pinecone. “Conversational Memory for LLMs with LangChain.” Pinecone Learn. https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/

[4] Pavlenco, Daniela. “Which Agent Memory Approach Is Best for Long Conversations?” Oracle Developers Blog, June 5, 2026. https://blogs.oracle.com/developers/which-agent-memory-approach-is-best-for-long-conversations