5 Pitfalls to Avoid When Building Advanced Voice AI Systems

Understanding the voice AI landscape in 2025

Voice AI has now evolved from an experimental prototype to a crucial tool for enterprise automation. With massive advances in large language models (LLMs), text-to-speech (TTS), and speech-to-text (STT), there is an assumption that building voice AI is simply a matter of connecting these components with telephony infrastructure.

However, real-world implementation reveals a more complicated reality. It might sound like any other development project, but when you try to stitch these components together, things start falling apart. Each individual component introduces complex state dependencies, latency constraints, and scaling inefficiencies when deployed in production.

That’s why, when it comes to real-time, scalable, enterprise-grade AI voice, the industry is stuck. Companies see the promise but get trapped in a cycle of prototypes, integrations, and rework, never quite achieving the seamless, intelligent, natural AI communication they envisioned.

The promise of AI agents crashes against the limitations of fragmented systems. To build truly effective AI-powered contact centers requires overcoming the limitations of fragmented architectures and designing a tightly integrated solution.

First, understand the hidden obstacles that prevent successful deployment. AI voice systems should not be a Frankenstein project of LLMs, TTS, STT, and legacy telecom platforms. If your goal is to implement AI-powered voice, chat, and video that truly works and scales, here are the five biggest roadblocks standing in your way (and how to avoid them before they stall your progress).

Pitfall #1: The "it seems simple" trap

Creating a voice AI system appears straightforward: combine language models for conversation, speech recognition for input, voice synthesis for output, and telephony for delivery.

Many development teams start with this modular approach, assuming they can simply connect these components through APIs: just plug in LLMs, STT, and TTS. But every additional integration introduces delays, state management complexities, and points of failure, especially at scale.

When you attempt to stitch together separate systems, each component introduces additional network latency, new points of potential failure, complex state management challenges, and synchronization problems at scale. A conversation that requires data to travel between multiple separate systems inevitably suffers from delays and disruptions that break the natural flow of human-like conversation.

Solution: The integrated approach

Rather than treating voice AI as a collection of separate APIs, successful implementations require an integrated system architecture. AI capabilities should be integrated directly into the media stack, creating a pipeline for voice processing.

This approach will minimize unnecessary network hops to reduce overall system latency, create more resilient connections, and overall, maintain conversation continuity. By processing voice, language understanding, and response generation within a unified framework, the AI voice system can achieve the sub-500ms latency necessary for conversations that feel natural.

Pitfall #2: The proof of concept wall

You may have been able to successfully create impressive voice AI demos that work perfectly in controlled environments, with demo videos that always look smooth… when you edit out the lag. However, when these systems move into production with real users and unpredictable conditions, they quickly deteriorate.

In real-world deployments, however, latency can spike during high call volumes, WebSockets could drop and hang up on callers, AI can lose context when network issues occur, and AI might face an inability to handle unexpected user interruptions.

The controlled environment of a demo rarely reflects the chaotic conditions of real-world deployment, leading to overall performance degradation at scale and disappointing results when voice AI meets actual users.

Solution: Build for real-world resilience

To overcome the proof-of-concept wall, voice AI systems must be designed with production realities in mind from the beginning. That means AI, speech processing, and telephony should already be integrated into a single real-time execution pipeline, enabling bidirectional streaming and proactive state persistence across network fluctuations. Real-time error recovery mechanisms should gracefully handle packet loss and allow the call state at the telephony level to preserve context even during network disruptions.

Systems should also gracefully handle interruptions, conversation transitions and unexpected user behavior by continuing the conversation with the user while, for example, pulling CRM data in the background.

Finally, test under variable network conditions and high load scenarios, focusing on resilience and real-world performance from the start.

Pitfall #3: The multi-channel integration labyrinth

The modern customer expects consistent experiences across all communication channels. Many organizations start with a single channel (often text-based chat) and later attempt to expand to voice, only to discover that their existing architecture doesn't translate well to real-time audio interactions.

In 2025, your AI system needs to work across voice, video, and messaging. But many AI solutions struggle to expand beyond their original channel, leading to clunky, disjointed experiences. Platforms without built-in telephony, video conferencing, and two-way text messaging may struggle to deliver the desired outcome.

You might find that:

- Voice requires lower latency than text-based interactions
- Different AI models optimized for different channels create inconsistent experiences
- State management becomes exponentially more complex across channels
- Development teams struggle with different technical requirements for each channel
- Customer journeys break when moving between channels

When voice AI is treated as an afterthought to existing systems, the result is typically a disjointed experience that fails to meet customer expectations. That struggle will extend to video and messaging, too, if they're not prioritized.

Solution: Channel-agnostic AI architecture

Rather than building separate AI systems for each channel, implement a unified AI engine that processes all inputs through the same cognitive framework, regardless of source. This avoids fragmentation and curates an omni-channel experience that creates consistent AI behavior across voice, chat, and video, all while maintaining context when users switch between channels.

Without an integrated approach, adding new communication channels leads to exponentially growing complexity. By leveraging AI that treats all communication channels as variations of the same fundamental interaction model, a seamless experience will follow customers across their preferred touchpoints.

Pitfall #4: The tool use conundrum

Basic voice AI that simply answers questions from a predefined knowledge base offers limited business value. Real impact comes when AI can interact with systems like CRMs, support ticketing platforms, payment gateways, and inventory management tools.

However, each additional integration introduces new complexity:

- Multiple API calls increase overall latency (again)
- Additional services create new points of failure
- Webhook servers must scale independently
- Security boundaries between systems complicate data access
- Each tool requires specific error handling and fallback strategies

As voice AI connects to more backend systems, the architecture can become a fragile web of dependencies that's difficult to maintain and troubleshoot.

Solution: Native tool integration framework

A simple voicebot that answers preset FAQs is one thing. But when you need AI to interact with other systems, the complexity skyrockets quickly.

Instead of treating external tools as separate systems requiring complex integration, implement advanced voice AI that supports tool use as a native capability within the AI conversation framework, which allows AI to access business data without excessive API calls, maintains conversation flow during tool interactions, simplifies security and access control, enables real-time, context-aware tool utilization.

Integrated tool use directly in the voice AI platform means that AI agents can incorporate business processes and customer data into natural conversations. For true scalability, the AI, speech processing, telephony, and tool use must function within the same pipeline.

Pitfall #5: The compliance riddle

Across industries, voice AI systems need to be able to handle sensitive information including personally identifiable information (PII), payment details, and confidential data. Standard LLMs weren't designed with these security requirements in mind, creating significant compliance concerns that prohibit the adoption of more advanced AI.

Considerations include:

- LLMs may inadvertently store sensitive information in their context window
- Payment processing often requires PCI DSS compliance
- Healthcare applications must maintain HIPAA compliance
- Different regions have varying data protection requirements
- Data leakage through insecure API transmissions.

Compliance must be prioritized, or organizations will have to choose between security and usability.

Solution: Security-first architecture

To launch enterprise-grade voicebots, you will likely need to collect sensitive information on live calls. How do you accomplish this without exposing sensitive data to public cloud LLMs?

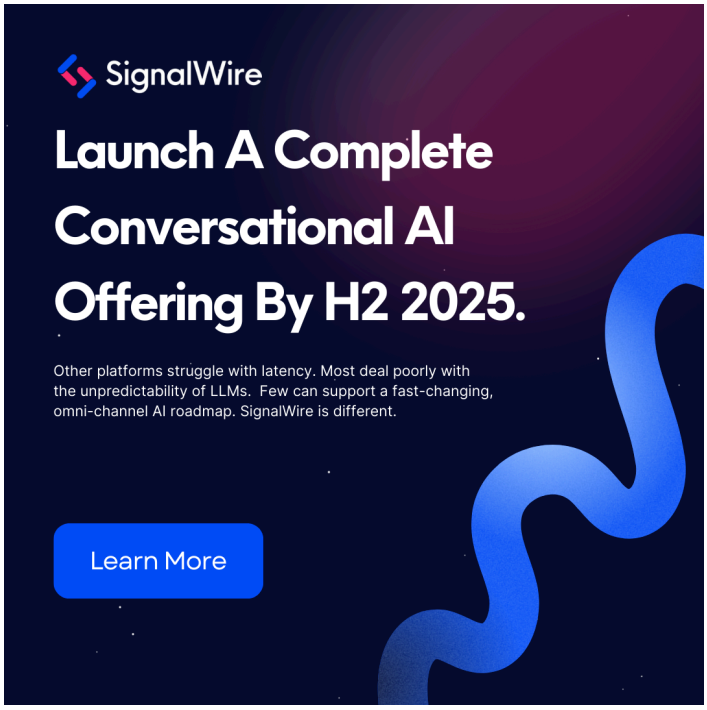
Advanced voice AI implementations can address compliance requirements through architectural design rather than bolt-on security measures, and should separate sensitive data processing from AI interaction. Effective approaches include:

- Dual-channel audio routing that processes sensitive data separately from AI conversation
- Selective context management that excludes sensitive information from LLM inputs
- Purpose-built compliance modes for specific regulatory frameworks
- Secure processing zones for sensitive operations
- Real-time redaction of sensitive information from transcripts and logs

By choosing tools to build voice AI that are already integrated tightly with compliance from the ground up, systems supported by AI can maintain both security and conversational fluidity, no matter the industry.

Building future-proof voice AI

The path to effective voice AI isn't found in choosing the right LLM or voice technology in isolation. Success comes from a holistic approach that addresses these five pitfalls through integrated system design.



The future of voice AI belongs to unified systems that prioritize integration, performance, and resilience over modular flexibility. By avoiding these common pitfalls and embracing a more cohesive approach, you can be at the forefront of unlocking the full potential of voice AI to *actually* transform customer interactions.

The future of voice AI is all about real-time, scalable, and secure conversational experiences. As you search for your holy grail AI system, remember that:

- Voice AI requires tight integration between speech processing, language understanding, and telephony.
- Real-world testing matters. Systems that work in demos often fail under production conditions.
- Customers expect consistent experiences across voice, chat, and video.
- Effective voice AI must interact seamlessly with backend systems and existing tools.
- Compliance should be built into the system design, not added hastily as an afterthought.

By addressing these issues upfront, you can have your AI communication system functioning in the next few months. Don't just impress in demos – deliver the transformative customer experience in real-world deployments that we've all been talking about.

AI Inside the Stack: Why SignalWire's Embedded Approach Is the Future of Real-Time Communication

How SignalWire Redefined Voice AI by Making It Native, Composable, and Instantly Deployable

---

1. Executive Summary

The transformation brought by generative AI has swept across industries, but nowhere is the need for rearchitecture more urgent than in telecommunications. While advances in LLMs have made intelligent interfaces common in consumer apps, real-time infrastructure powering voice and video remains constrained by designs built for an earlier era., but nowhere is the gap between promise and reality more visible than in telecommunications. While large language models (LLMs) and voice assistants have made significant advances in consumer-facing interfaces, the backend infrastructure powering real-time voice interactions has lagged behind. Most platforms are still relying on brittle integrations, clunky middleware, and legacy signaling paths that were never designed to support the dynamic nature of AI-driven workflows.

This leads to a host of systemic issues: inconsistent latency, dropped context, poor speech recognition, and AI agents that can't meaningfully complete tasks or escalate intelligently. Meanwhile, developers are burdened with stitching together cloud APIs, provisioning compute for inference, managing state manually, and compensating for jitter and lag at every layer.

SignalWire offers a radically different approach. By embedding AI directly into the media and signaling fabric itself, rather than layering it on top, SignalWire has created a system where intelligence is native to the communication process, not an afterthought.

This architectural shift means developers can now create and deploy real-time, programmable AI agents in minutes, not weeks. The result: faster, smarter, lower-latency voice experiences that scale natively across SIP, WebRTC, PSTN, and more. In this whitepaper, we break down the core elements of SignalWire's embedded AI model, why it works, and why it's the inevitable future of communications.

---

2. The Problem: Legacy Models Can't Keep Up

AI has been retrofitted into legacy telecom stacks using brittle methods: streaming audio through WebSockets to cloud-hosted LLMs, manually bridging STT, NLU, and TTS components, and building context switching on the fly. This approach not only introduces delays of up to several seconds per turn, but also fractures the user experience, as handoffs between systems become unreliable and context is frequently lost.

Developers tasked with building these solutions often find themselves writing glue code just to make basic functionality work. They must manage session state across services, handle retries and failure recovery, and monitor latency across several different APIs, all before even beginning to apply RISEN-style prompting or define structured task prompting using POM (Prompt Oriented Model).

Meanwhile, enterprise teams struggle with scalability. Each AI interaction requires costly compute, and the architecture can't reliably adapt to spikes in traffic or changes in behavior. Latency becomes inconsistent. Speech recognition breaks under jitter. Callers experience long pauses and awkward transitions. And because most systems rely on a cloud AI provider, privacy, cost, and compliance become even harder to manage.

CPaaS providers advertise "programmable AI" experiences, but what they deliver is fundamentally reactive: calls must leave the platform to perform reasoning, and developers are forced to operate like systems integrators instead of solution builders. What customers want, conversational, responsive, task-completing agents, remains out of reach for all but the most heavily engineered systems.

---

3. SignalWire's AI Kernel

SignalWire took the hard path first: embedding AI into the same runtime that processes calls. Its **AI Kernel** runs inside the same infrastructure layer as the media and signaling engine, eliminating the need to offload audio to remote services and reducing the cognitive load for developers. Unlike traditional cloud integrations, where AI lives at the edge of the call and must be fed audio via proxy, SignalWire's model allows agents to live in-line with the conversation.

SignalWire's embedded execution model works with wide, general-purpose models that are dynamically focused using RISEN-style prompting techniques and innovations like POM (Prompt Oriented Model). This design gives SignalWire the ability to:

- Transcribe and analyze speech with sub-500ms round trip latency, ensuring lifelike conversations
- Execute language model logic directly within the session, without leaving the fabric
- Enforce guardrails, memory policies, and role constraints locally, without exposing raw user data to third-party services
- Scale agents like infrastructure, not like brittle apps that must be provisioned, patched, and monitored manually

This embedded approach solves what most platforms treat as technical debt: variable latency, fragmented control flows, and inconsistent AI state. With embedded execution, every action the AI takes is synchronized with the telephony state, meaning it can influence call routing, media playback, DTMF collection, or even recordkeeping in real time.

With **Call Fabric**, SignalWire enables AI Agents, Rooms, Subscribers, and APIs to behave like addressable endpoints in a real-time network. Logic and orchestration are described in **SWML**, a declarative language that empowers developers to define interactions without procedural code, eliminating the need for redundant backend infrastructure.

---

4. From Prompt to Production in One File

Building a full-featured AI agent in SignalWire requires one SWML file. This file encapsulates everything needed to define the agent's personality, capabilities, and behavior within a live call. A single `ai:` block can set the role (e.g., receptionist, technical support, concierge), define tone and conversational style, configure memory retention policies, and assign tools to take real-world actions.

Developers can add a `post_prompt_url` to trigger a webhook after the call ends, allowing for automatic CRM updates, ticket creation, or even sentiment analysis. Tools can be embedded using the `swaig` syntax to access internal systems and external APIs, perform lookups, or securely collect payment information. All of this logic lives in the same file, tightly coupled with the media layer.

Once defined, the agent can be bound to any channel: PSTN via phone number, SIP via subscriber, or a Room for multiparty video. SignalWire's Call Fabric handles routing and execution instantly, with no need to deploy cloud infrastructure, spin up VMs, or write microservices to connect the pieces.

The result is an entirely serverless deployment flow. Just publish the file and the agent is live, ready to take calls, escalate to humans, access APIs, and summarize conversations in real time.

SignalWire's Voice AI interface is powered by innovations like POM, Prompt Object Model, a strategy for driving agent behavior through a simpler document object schema, rather than model tuning or training. With POM, developers define how the AI should behave using structured domain-specific prompts that structure tasks in a predictable, reusable way. This removes the need for fine-tuning, ensures model generality, and increases transparency and safety. POM enables RISEN-style prompting that adapts wide models to narrow use cases by orchestrating prompts around role, intent, state, execution, and next steps.

5. Developer Velocity: From Months to Weeks

SignalWire's approach collapses development time dramatically by transforming what traditionally takes months of API integrations, infrastructure provisioning, and multi-vendor coordination into streamlined, declarative development that can be completed in days or weeks. By embedding AI directly into the real-time communication layer, SignalWire eliminates the need for developers to manage speech pipelines, orchestrators, or voice SDKs. The stack is unified from day one.

Teams that once needed months to plan and build voice applications with AI, factoring in separate components for STT, TTS, LLMs, logic, telephony, routing, escalation, and post-call workflows, can now ship prototypes in days and production-grade systems in under two weeks.

Certain milestones that used to be complex and take months and years are now reduced to days and weeks

- Create and test an AI intake bot with call routing
- Add real-time API logic (e.g., check open tickets)
- Run a full conversation and test the application.

Because everything is declarative, developers focus on *what should happen* rather than *how to plumb it together.* SignalWire abstracts the real-time stack through Call Fabric and SWML, enabling a development experience that feels more like working with modern web frameworks than legacy telecom systems.

This shift reduces operational overhead, accelerates product delivery, and gives both startups and enterprises a repeatable model for deploying advanced voice AI without the friction of traditional CPaaS or cloud-native telephony stacks.

6. Why Everyone Else Is Behind

Each of these solutions suffers from the same core limitations: they treat AI as something to integrate rather than something to embed. Twilio and Vonage offer AI features via cloud connectors or middleware, which means every conversational turn introduces latency and potential failure points. PolyAI and Dialogflow-style systems rely on static intent models and are best suited for narrow, pre-scripted use cases, not fluid, cross-domain dialogue or action-taking.

LiveKit and Daily provide excellent media pipelines, but leave it up to the developer to build everything else, including agent state, orchestration logic, and system integrations. These infrastructure-only stacks may be performant, but they offer no built-in path to intelligent automation.

Meanwhile, proposals like MCP (Modular Communications Protocol) envision a future of standardized interfaces between AI and telecom, but they lack execution. SignalWire has already shipped what these designs hope to become: a composable, programmable, real-time environment where AI lives inside the fabric, scales on demand, and acts with context.

This is not just a matter of feature parity. It's a philosophical difference. SignalWire enables builders to create meaningful, production-grade AI voice experiences without needing to cobble together a dozen services. That's why developers are switching, and why the next wave of AI in telecom won't be glued together. It will be **composed.**

SignalWire is already running real-time agents in production globally, across SIP, PSTN, mobile, and video, without middleware, plugins, or heavy DevOps. Its composability and embedded performance are unmatched.

---

7. The Strategic Future of AI in Telecom

In a world where AI agents replace forms, menus, and scripted logic, execution matters more than model selection. While large language models continue to improve in accuracy and breadth, they will become increasingly commoditized. The true differentiator will be where and how those models are executed, and whether they can operate with continuity across channels, use cases, and sessions.

SignalWire recognizes that intelligence must live not on the edge of the network, but in its core. By embedding AI directly into the media stack, it makes the agent a first-class system component, one capable of reasoning, taking action, and controlling flow in real time. These agents don't just respond to queries; they drive the conversation, manage escalations, and perform backend logic with secure, verifiable outcomes.

SignalWire's model introduces a new era where call flows are no longer hard-coded or stateless, they're intelligent, dynamic, and conversational. With shared memory and secure context preservation, AI agents can pick up where they left off across calls, or operate in parallel across voice and chat. The result is a fluid, intelligent user experience that feels far more like a human assistant than a phone tree.

This shift redefines what telecom is for: no longer just a conduit for sound, but a programmable, intelligent medium for service, support, and automation. The platforms that embrace this shift will be the ones that define the next generation of customer experience. SignalWire is already there.

This isn't about plugging AI into telecom. It's about transforming telecom into an AI-native execution environment, one where intelligence and media operate as a unified system.

---

8. Conclusion: The Stack is the Strategy

SignalWire is not just a communications platform, it's the foundation for a new generation of AI-native systems. If you're ready to deploy intelligent agents that act in real time, scale instantly, and integrate seamlessly into your communications fabric, SignalWire offers everything you need.

Start building today at signalwire.com, or explore the developer docs to launch your first AI agent with SWML in just minutes.

SignalWire didn't just add AI to voice infrastructure. It **rebuilt the voice stack to support AI** natively, with developer experience, security, and performance at its core. That's why developers can launch useful, intelligent voice agents in minutes, and why enterprises are choosing SignalWire to scale their communications infrastructure with confidence. The next generation of communication isn't stitched together. It's composed. And it's already running, on SignalWire.

Appendices

A. SWML Agent Example with SWAIG Tools
B. Benchmark: Round-Trip Latency vs Competitors
C. Developer Onboarding Flow
D. Glossary of AI Integration Terms (SWML, SWAIG, Post-Prompt, etc.)

Why is it a bad idea to create your own custom LLM or other GenAI model?

Building a custom LLM or GenAI model from scratch is a costly and complex endeavor. It requires vast amounts of data, computational power, and expertise in machine learning and NLP. Additionally, maintaining and fine-tuning such a model is an ongoing effort that demands specialized talent. Even large tech companies struggle with the high costs and time investment involved. Instead of reinventing the wheel, businesses can leverage existing models integrated with platforms like SignalWire, which offer lower latency, high efficiency, and seamless telecom integration.

How does SignalWire AI help developers get around these problems?

SignalWire AI is not just a pre-built solution but a complete IVR framework with an embedded AI kernel. This allows developers to define any AI interaction as prompts and webhooks, making it far more flexible and powerful than other market offerings. Instead of requiring developers to train and maintain AI models, SignalWire provides a framework for fully programmable AI-driven communication. With SWML (SignalWire Markup Language), developers can create dynamic, interactive call flows that leverage AI for real-time speech processing, decision-making, and seamless API integration. This eliminates the need to build AI-driven telecom solutions from scratch while maintaining full control and customization.

Why is it hard to add AI to phone calls and telecom systems?

AI-powered voice interactions require ultra-low latency to maintain natural conversation flow. Traditional telecom systems introduce delays due to signal routing, audio processing, and external API integrations. Most other AI telecom solutions rely on multiple third-party providers, each adding processing overhead. Media is often transported across internet sockets in heavy formats, further exacerbating latency issues. On top of this, PSTN and mobile phone networks introduce inherent latency beyond anyone's control. Each additional processing hop, whether for transcription, AI inference, or call routing, adds delay, forcing competitors to pull media away from the media stack only to return it with significant latency. SignalWire eliminates these inefficiencies by embedding AI directly within its telecom infrastructure, keeping media processing local and reducing round-trip delays to ~500ms.

Why is AI being integrated into the telecom stack a big deal? What problems do telecom networks create that make AI struggle?

Integrating AI into the telecom stack eliminates common issues like high latency, signal degradation, and difficulty in synchronizing real-time speech analysis. Traditional telecom networks were not designed for AI-driven interactions, leading to inefficiencies when integrating AI post-facto. SignalWire's native AI integration solves these problems by handling voice, video, and messaging directly within the same infrastructure, optimizing for real-time performance.

Unlike other solutions that require media to be offloaded to external processing centers, introducing significant round-trip delays, SignalWire's AI kernel is embedded directly in the

media stack. This allows input handling, AI processing, and LLM access to work in unison, ensuring that media never has to leave the system. By eliminating these extra hops, SignalWire significantly reduces latency, preventing the unnatural pauses and lag that typically plague AI-powered voice interactions. The result is a seamless, real-time conversation experience that feels natural and responsive.

Why is it such a big deal that SignalWire AI works across PSTN, WebRTC, SIP? How does SignalWire's AI adjust to different types of calls without breaking?

Telecom networks use different protocols, each with unique constraints. PSTN calls have inherent latency, SIP calls involve complex routing, and WebRTC is optimized for real-time interactions. SignalWire AI seamlessly adapts to each of these environments by operating directly within the SignalWire media stack, ensuring consistent performance across all communication types.

Beyond just AI, the SignalWire platform allows businesses to build rich media applications once and deploy them across PSTN, WebRTC, and SIP without needing to implement different versions for each protocol. This is essential for UCaaS, CCaaS, and AI applications, as it enables customers and agents to access the same features across all networks seamlessly. By eliminating the need for multiple implementations and expensive adapters in the middle of the call flow, SignalWire reduces both infrastructure costs and latency, ensuring a smoother, faster, and more efficient communication experience.

Why would someone use SignalWire AI over other competitors like Bland, Sierra, or even Twilio? How is SignalWire's AI built differently from others?

SignalWire AI is designed from the ground up for ultra-low latency, real-time telecom applications. Unlike competitors that rely on third-party CPaaS for media handling, SignalWire integrates AI directly into the media stack, eliminating unnecessary processing layers. This results in faster response times, better scalability, and a more seamless AI-driven conversation experience.

SignalWire is built on the foundation of FreeSWITCH, which has been a leader in scalable telecom infrastructure for years. On top of this foundation, SignalWire has added enhanced media capabilities, allowing for immediate scalability instead of the challenges that new companies face when trying to scale on third-party CPaaS providers. This reliance on CPaaS by other competitors not only increases their latency, as previously discussed, but also limits their ability to provide the deep media integrations that SignalWire offers.

Additionally, SignalWire AI delivers the fastest response times in its class for voice AI, ensuring a near-instant interaction experience. Its framework is also the most robust for building AI tools that can control middleware, execute native media stack operations like call transfers and digit dialing, and securely perform PCI-compliant number collections. With these capabilities, SignalWire is not just another AI telecom provider, it is a fully programmable and scalable AI-driven communications platform.

How would you describe SignalWire's AI security measures? What encryption standards or compliance measures are in place?

SignalWire AI adheres to strict security protocols, including end-to-end encryption and metadata tokenization. Compliance measures include GDPR, HIPAA, PCI, and SOC 2 Type 2 certifications, ensuring that sensitive customer data remains secure. Access controls and logging features help businesses maintain compliance with enterprise security requirements.

A unique security feature of SignalWire AI is its ability to invoke media stack functions to transition into a secure IVR within an AI conversation. This means that during sensitive interactions, such as collecting payment information or personal identification, the AI can seamlessly hand off to a secure IVR flow where critical data is collected without ever being exposed to the AI or the broader system. This ensures that sensitive information remains protected and prevents unnecessary AI exposure to private user data, maintaining strict compliance with security regulations while providing a seamless customer experience.

CEO

Anthony Minessale

I've been working in the telecommunications industry for 20+ years, and my life's work has revolved around transforming telecom to software and making it programmable.

I have been working on making voice bots that don't suck since they sucked. But I'd like to think we made them suck a bit less and made them more accessible to the world. Before FreeSWITCH, IVRs and voicemail used to cost 100k for a physical server that was converted to pure software.

Like the Internet, when it was new to the general public, as soon as LLMs that could be consumed at scale over a reliable API appeared, all of its use cases were almost immediately obvious. I think anyone can conclude that with an LLM generating human language, you can wire it up to text-to-speech and speech-to-text or to an existing chat interface.

My thesis on approaching this technology started with the hardest thing, voice, as making a chat-based interface is almost trivial in comparison. I have a few key tenets that I worked with initially.

1) Focus on the leading technology (OpenAI)

Like with computers, the Internet, and other inflection point changes, you can presume that the core technology will eventually reach equilibrium and that business cases for using the technology will be the main differentiator. So, rather than an early focus on the model, I wanted to focus on the use case (allowing developers to build digital employees). OpenAI has several things about how its APIS work that give a low-level developer more power to exploit and simplify for end-user developers.

2) Leverage the AI in a way that limits its ability to be exposed to sensitive data by combining it with existing voice menu technology

Our stack is designed as an extension of a more massive stack for general scalable telecom services. The AI stack can actually call into a traditional voice menu to collect sensitive info from the user. Then, it can save a meta-data token and just tell the AI that the data was collected without it ever actually interacting with it.

3) Create an environment for developers that is as easy to use as the Web

Our stack is inspired by UNIX and the Web and is, in essence, an extension of those concepts. The AI Agent you create could be considered a new way to look at web forms in a way that allows you to talk to them. Its prompts tell it how to talk to the user and collect info, then use functions to post that data to your servers, where you can feed information back and modify its behavior in several ways. Because our AI stack is one method in an even larger telephony stack, it has all the powers of its parent platform to work with to transfer calls, send SMS, and scale to any number of concurrent calls that can be over phone lines, SIP (VOIP), WebRTC and UCaaS applications on our platform.

4) Be the lowest latency period

Since the first experiment, I have been working on designs to eliminate latency. I have developed several techniques that are solved mostly by ingenuity and deep knowledge of C, and I have one of the most powerful communications platforms on the planet under my belt.

I know the new GPT-4o is trying to short-circuit the whole thing by doing some of the work in a mobile phone and has models that can ingest audio, but they have a billion dollars to burn, so I would expect nothing less. Meanwhile, I think we have a system that actually does useful things and responds at a speed where you might even consider slowing it down in some cases (which is configurable). We are able to use almost any modern TTS natively and minimize latency at all points of orchestration.

Built to scale

Our AI orchestrator is just another cog in a proven powerful engine so applications can scale forever with the horizontal implementation of its parent platform. Conversational experience will always be a challenge (it still is with real humans, after all) but I think we offer such a wide array of params and ways to approach it that we will evolve further and further.

Vertically focused

We have an elaborate system called SignalWire AI Gateway (SWAIG) that is built on top of the Agent framework and allows deep integration points that are the building blocks to implement drop-in skills. We have several demos for things like booking a restaurant, ordering flowers, cable technician, MFA bot, and more. Rather than focusing on training models, we strive to leverage the least training possible to build working applications

quickly. We have a system in the works that will allow live vectorization of arbitrary documents to provide on-demand

I think most people fake their results and we have a strict rule against that.

Realistic in Scope

This is key. The expectations that are often set by how cool AI is. Sometimes, it causes people to make the wrong assumptions. This is nothing new as we, as engineers, often refer to the National Park vs Bird xkcd comic.

We believe that systems using digital employees created with our stack can do a lot of things immediately and even more as the technology progresses:

- Sorting calls for humans.
- Gathering information ahead vs. keeping customers on hold.
- Being voicemail without being voicemail.
- Acting as personal assistants.

What's different from others?

I've seen a few of the other AI stacks, and while we have some things in common, I think our approach has some key differentiated elements.

Part of a fully Programmable Unified Communications stack.
The AI agent stack is tethered to a horizontally scaling UCaaS, CCaaS, CPaaS platform. You can access the agents from SIP, PSTN, WebRTC UCaaS applications not limiting the experience to over the phone.

Scaling Complexity

As your needs evolve, you can deliver dynamic content to generate on-demand agents or build out SWAIG services to allow your agent to get more and more technically capable.

Context Switching

The agent can be dynamically altered mid-conversation to change its focus or core prompting so you can have a general Agent who suddenly becomes completely focused on finding you a good movie to watch and then suddenly able to book a restaurant reservation. There is also a "steps" mode that will force the agent down a series of pre-determined steps where its core mission is updated as it completes each one. This could be combined with context switching to do a more thorough job.

Sliding Window

Rather than continually posting the entire conversation to the AI, a sliding window can be defined to limit the conversation to a certain number of turns. I have an example where it's wired into an infocom text adventure from the 80's where it gets you to say what you want to do and feeds it to the game (no need to save the whole conversation).

A simple digital employee can be created as a single JSON file that can be hosted on our systems or delivered over webhook.

Multiple Voices, Languages and Fillers

An agent can have a single muti-lingual voice or different voices for each language and can switch between them just by asking it to. Each language can have optional fillers which are a list of phrases to indicate the agent is looking something up where the remote service may contribute enough to latency that it has to say, "ok, one second" or "hold on." For extreme cases like one example where you order flowers and it uses an AI image generator to text you a picture of the flowers you describe, a sound file can be played like pencil scribbling or keyboard typing.

Media Files

The agent can play a looping background sound like people talking in a coffee shop or kids playing in a park or a busy office to add to the experience. Via SWAIG the agent can be instructed to play a file as part of the conversation. For example, I had a "duck debugger" who offers to help you find your problem and pretends to ask a rubber duck who quacks in response to your questions.

Video

If the agent is called over a video enabled line, it can use a series of mp4 files to simulate a state of idle, paying attention, and talking. If it's instructed to play a video file, in place of its avatar, it can stream the video to the remote user. I have plans to eventually have it snapshot your inbound video and "see" what is in the picture.

Advanced Barge Cutoff / Adjustable Latency

To pursue the best conversational experience, the agent defaults to a behavior where if you start talking over each other, not only does it stop and let you win, but it also combines everything you are saying into a single turn and removes any early responses. So if you are talking to the agent with many pauses in your speech pattern and the agent starts answering too soon, just by continuing to talk, it will adjust to the new combined utterances and form a completely new response.

Built-in Post-Prompting and full logging and debugging

A post-prompt can be defined to do a final action such as summarizing the conversation, formulating a pizza order into a json blob, etc. Special SWAIG functions are available at the beginning and end of the conversation to take a leading or final action in particular cases.
Debugging webhooks can be defined to obtain mid-conversation details, so you know what's going on as it works.

Unified support for ASR/TTS

All of our supported ASR/TTS interfaces are available to the AI and the normal voice menu system, so you can mix and match advanced AI portions of the conversation with classic voice menu techniques. Currently, we support Elevenlabs, Google, PlayHT, and Amazon Polly, and adding more is trivial. Any new ones appear in all aspects of the platform.

Persistent Conversation Tracking

Several features enable the agent to be seeded with information from a previous call so you can provide the sense that it knows what happened in previous conversations.

Ability to receive chat or SMS during the conversation

Messages can be sent to the agent while you are talking to it on the phone, allowing you to chat or SMS with it while you are talking to it, for cases like sending your email without reading it out loud or getting helpful links from the agent. The Santa Claus example asks you what you want for Christmas and then sends you links at Amazon. The flowers example asks you what flowers you want and who to send them to and texts them a custom AI-generated picture of flowers based on your input.

SWAIG

The SignalWire AI Gateway is a native interface that is immeasurably flexible as part of our AI stack. It extends the basic 'functions' feature from the language model into a system that allows the developer to create solutions with evolving complexity, starting with static file and template-driven data expansion and moving on to full remote custom webhooks.

A Definitive Guide to Selecting Your AI Voice Agent

Part 1: Introduction to AI agents

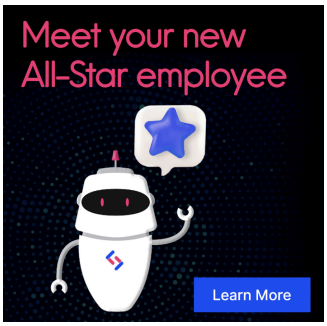
Imagine a voice assistant that's designed to work alongside your team, handling repetitive tasks and freeing up more time for live agents to focus on what they truly excel at: providing personalized, human touchpoints for your customers.

An AI agent is designed to streamline operations such as call transfers and appointment scheduling, empowering employees to create better customer experiences. The AI acts as a digital tool in your team's toolkit, elevating everyone's efficiency and effectiveness.

Although you might typically build an interactive voice response system (IVR) to sort incoming calls, an IVR does not actually understand what a caller wants. It is simply there to transfer the call or repeat a pre-recorded message based on fixed logic.

IVRs that misunderstand our queries can lead to a loop of repeating instructions, leaving the caller stranded and frustrated. Whether it’s calling a pharmacy, a doctor’s office, or a phone service company, getting stuck in a loop with a barely-responsive phone tree that refuses to transfer you to a live agent is a poor customer experience.

An AI voice agent provides natural and engaging conversation with the caller, allowing customer service representatives to offload repetitive calls to efficiently accomplish complex tasks. The AI can perform the initial step of greeting the caller, answering questions, and troubleshooting common issues before transferring the call to the appropriate agent. And it will understand immediately if the caller cuts in and demands to be transferred to a live agent!

[visual inspo:] 

In addition to frustrating callers, integrating a traditional IVR or automated agent involves complex code, logic, and engineering expertise. But what if you could quickly and easily deploy a virtual agent that goes above and beyond the capabilities of traditional IVR?

Part 2: What does an AI agent do for you?

Delivering exceptional customer service is crucial for success and plays a pivotal role in shaping a company's reputation. Customers expect personalized and efficient interactions, and businesses need to find innovative ways to meet these expectations while reducing costs and improving operational efficiency.

Relying too heavily on IVRs can lead to unhappy customers when callers find themselves stuck in long queues, or waiting for assistance from live agents when the IVR can't understand what they want. IVR systems and automated agents often lack the ability to adapt to human variance and fail to deliver the natural experience that customers desire.

An AI agent can handle high call volumes, automate routine tasks, and improve response times, all while maintaining a natural and engaging conversation with customers. It can process vast amounts of data and offers users increasingly accurate and personalized responses.

With adaptability and intelligence, AI paves the way for a new era of customer service, where robotic interactions are replaced by meaningful conversations. By automating responses to common queries, order tracking, FAQs, and basic troubleshooting, AI agents free up human agents to focus on more value-added interactions.

For businesses aiming to provide an unparalleled customer experience, AI agents offer a glimpse into the future of customer service. The ability to understand, engage, and assist customers in their preferred language while seamlessly integrating with existing systems is game-changing.

The versatility of AI agents makes them suitable for a wide range of businesses. Small businesses struggling to scale up can benefit from 24/7 availability, allowing customers to access routine information and assistance at any time. Global enterprises can use an AI agent to efficiently route calls in any language to the appropriate departments while maintaining meaningful interactions.

With an AI agent, you can ensure that your customers receive prompt and personalized support around the clock.

Part 3: How an AI agent can grow your business

The transformative technology of artificial intelligence has diverse applications across industries. Implementing AI into customer service operations is an effective way to remain competitive anywhere in today's market. According to a 2022 study by IBM, over one-third of organizations

implementing AI to improve customer care were doing so in order to improve live agent productivity and create more personalized interactions. 28% were using AI to decrease wait times.

[visual: one-third of organizations implementing AI to improve customer care were doing so in order to improve live agent productivity]

47%

According to Elsevier Health, nearly half (47%) of U.S. healthcare workers plan to leave their positions by 2025.

There are many ways to use AI to cut costs, improve efficiency, and grow your business. With an AI agent, any call center functions can be easily streamlined, like sorting incoming support calls, summarizing the issue before transferring, recording customer issues and troubleshooting. There are also some specific use cases where AI agents excel:

Use Cases

Virtual receptionist

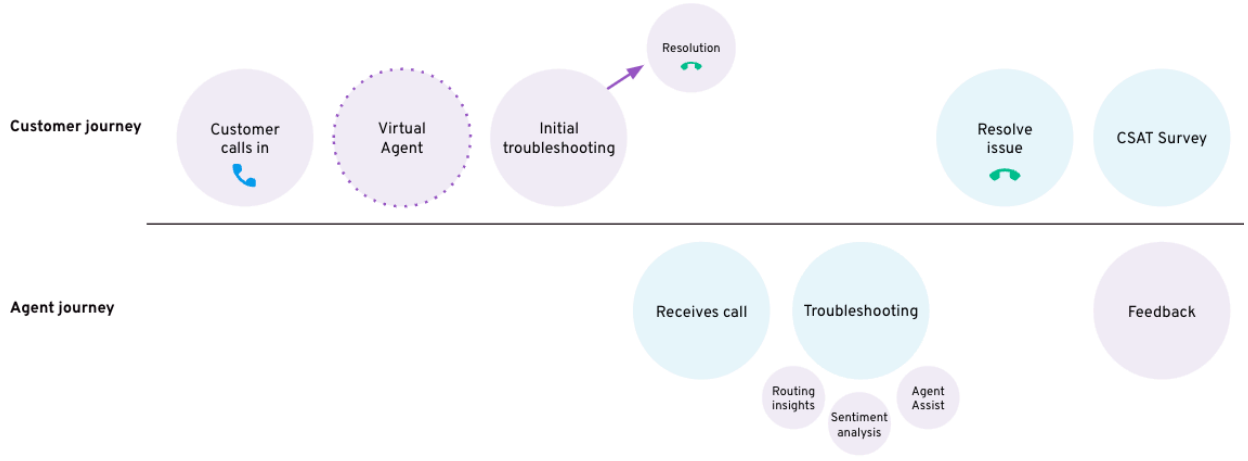
AI agents are proficient virtual receptionists. They optimize calls with intelligent automated conversations, providing personalized greetings and instant information retrieval.

Virtual receptionists do more than just replay pre-recorded messages; they can perform actions such as transferring calls, scheduling meetings, or taking messages, allowing the human receptionist to focus on more involved tasks. This not only improves efficiency but also delivers exceptional first impressions to callers.

If a customer were to reach out at 2AM, a virtual assistant could provide this customer with immediate, AI-powered support. Calls are answered during off hours and important information is recorded for those who need it the next day.

Customer support

[visual inspo: customer support routing through ai agent]



AI agents empower customers by providing intelligent self-service and accurate issue resolution. By retrieving customer information and taking actions to open or close support tickets, the AI agent streamlines incoming support calls.

The AI can summarize the issue before transferring the call to live support, reducing response times and increasing agent efficiency. Customers can get the help they need faster, and even successfully troubleshoot and resolve basic issues.

Managing reservations and appointments

AI agents efficiently manage reservations and appointments. Whether it's booking a table at a restaurant or scheduling a medical check-up, the AI system streamlines the process, ensuring accuracy and convenience for both businesses and customers. This automation improves customer experiences while reducing administrative overhead.

If a medical practice were to implement an AI receptionist, for example, patients could book appointments, inquire about clinic hours, and receive automated reminders for upcoming visits. This streamlines administrative tasks for the healthcare staff, allowing them to dedicate more time to patient care.

Sales and marketing insights

AI agents' ability to observe customer behavior at scale allows for the collection of more data than humanly possible. An AI agent can record and even transcribe invaluable data, including customer interactions, preferences, and pain points. This can be used to inform sales, marketing, and other analytics efforts, allowing you to optimize strategies and tailor marketing campaigns for maximum impact.

Enhanced spam filtering

Security is paramount, and AI agents can play a pivotal role in blocking out spam and scam calls. It can filter spam by analyzing incoming caller IDs and detecting potential bot calls, safeguarding your business from fraudulent activities and ensuring a secure communication environment.

Polling data and surveys

AI agents can efficiently conduct polling by making automated phone calls, gathering data on a large scale. This automation not only saves time but also ensures accuracy in data collection.

Election season in the United States involves extensive phone banking and outreach. Instead of relying solely on traditional phone polling, interviewers can deploy AI agents to reach a broader audience. These agents can conduct surveys via phone calls, collecting valuable data on voter preferences, concerns, and demographics.

Part 4: What to look for in an AI Agent

[visual: what to look for in an AI agent - voice capabilities, Efficiency, Personalization, Multilingual capabilities, data-driven insights, adaptability, low-code, no code options, programmable actions]

Customer service no longer has to be a mundane task handled exclusively by human agents. AI agents can step in to revolutionize the way businesses interact with their customers. Depending on your company's needs, there are a few features that are key to think about when it comes to finding the right AI voice agent for you:

Voice, not just chat

AI-powered chatbots are quite common, and have been around for years. AI voice agents introduce a novel approach to customer service with AI-powered voice. These agents are designed to replicate the experience of talking to a real person and respond to human variance, parsing nonsense from the real logic behind inquiries.

Efficiency

AI agents should speed up response times and connect callers to the right agents more efficiently. They handle routine questions, allowing human agents to focus on more complex issues. They deliver consistent service quality, ensuring a positive customer experience every time. Their 24/7 availability guarantees customers receive support regardless of the time zone or business hours.

Reduced costs

By automating repetitive tasks, AI agents in turn save money on labor and operational costs. AI agents also only incur costs while they are on the phone; they do not need to be paid by the hour or by the day. Additionally, eliminating the need for complex code reduces the reliance on specialized expertise, which significantly lowers development costs and accelerates the time-to-market for conversational AI.

Personalization

A good AI agent offers nuanced, context-aware, natural interactions. It's not just about automated responses; it's about understanding your customers' needs, preferences, and providing tailored assistance. Integrations with backend databases and third-party applications enable personalization and data analytics, delivering a high-quality user experience.

Multilingual capabilities

AI agents should enable you to connect with a diverse global customer base with support for various languages. This is particularly valuable if you have an international clientele. With multilingual support and the ability to understand different dialects, AI agents allow businesses to engage with a global audience effortlessly, breaking language barriers and expanding your reach.

Data-driven insights

Unlike traditional systems, AI agents can offer in-depth analytics and collect data beyond what is humanly possible. An AI agent can not only record calls, it can transcribe them, making it simple to sort through customer conversations later on. This data can be used to understand customer behavior, preferences, and pain points, helping you optimize your strategies.

Adaptability

An AI agent's intelligence extends beyond pre-defined instructions. It possesses the ability to comprehend and respond to queries beyond a scripted format. This adaptability should enable it to engage in meaningful conversations with customers, enhancing the overall customer experience.

While traditional IVR systems may struggle with the natural variance in human speech, AI agents excel. They should be able to understand and respond to queries beyond scripted formats, engaging in meaningful conversations with customers and getting smarter with every call, enhancing the overall customer experience.

Low code, no-code options

Anyone should be able to build their own AI-powered voice applications. Easy-to-use interfaces allow for quick deployment of simple AI agents with absolutely no code.

Implementing AI solutions can be challenging, especially for small and midsize businesses that lack the resources, skills or infrastructure to develop and deploy AI applications. But a no-code builder allows anyone, even those with no engineering background, to quickly build and deploy an AI agent.

A no-code platform simplifies the process of building AI applications that are easily trained and customized. This simplicity streamlines the development process, which delivers AI applications to your existing infrastructure faster.

A user-friendly interface is a necessity. With basic technology skills and plain text language, users should be able to make quick and effortless changes to their AI agents, eliminating the need for time-consuming coding and deployment cycles.

Programmable actions

With programmable voice capabilities, you can create more dynamic and versatile voice applications. For creators and developers, a robust AI agent enhances customization and flexibility with programmable actions. You should be able to integrate with backend CRM systems for relevant responses and specific actions if desired.

This enables you to, for example, send SMS messages or access customer information from backend CRM databases. These integrations empower AI agents with real-time access to relevant data, enabling them to handle customer queries more efficiently and deliver high-quality personalized interactions.

Part 5: What to look for in a vendor

If an AI agent sounds like something that could help your business reduce costs, increase revenue, and decrease repetitive tasks for employees, the next step is to evaluate vendors. The number of products leveraging AI is expanding rapidly. Consider what functions are necessary for your AI agent, any data it will need to access, and the technical requirements involved to implement AI.

[visual:
What does your business need from AI?
- Cost benefit analysis
- What role will AI take on?
- What problem will AI solve?
- What functions does your AI agent need to have?
- What are your technical requirements?]

Once you understand your own needs from AI, and you have evaluated individual products, look to vendors who have:

Strong documentation and developer community

Resources for getting started and maintaining new technology are key to the success of any project. Look for extensive, user-friendly documentation to assist developers in integrating AI agents seamlessly into their systems.

Engaged, active developer communities are also helpful when trying new technology. Look for a vendor who fosters a thriving developer community where you can learn, share, and collaborate to maximize the potential of AI voice agents.

Responsive customer support

Responsive customer support is the biggest necessity for any technical product. You need a team that is committed to providing world-class, responsive assistance to ensure a seamless experience.

Seek out a vendor who takes pride in providing world-class customer support. With a responsive and helpful team, your questions and concerns will always be addressed promptly.

Depth of Knowledge

Look for feature-rich offerings in an AI agent. When assessing vendors, it's essential to compare the features they offer, and how they respond to specific requests about your use case. When it comes to AI voice agents, you don’t just need a partner who excels at AI technology - you need a partner who excels at voice capabilities and is familiar with other communications technologies.

About SignalWire AI Agent

SignalWire's team is the driving force behind the creation of the AI Agent. Built by the minds who created the FreeSWITCH open-source project, SignalWire has unmatched, extensive expertise and a profound understanding of voice technologies.

With a responsive customer support team unrivaled in the industry and a thriving community of developers behind it, SignalWire offers an AI agent that is backed by all the resources you need to get started. Options for programming your own custom AI are available, but with a user-friendly design that requires minimal coding for rapid development. This makes our AI Agent accessible to individuals with varying technical backgrounds.

Users can easily instruct the SignalWire AI Agent on how to respond and behave in different scenarios with plain text instructions. And when it’s not easy, the support team and the community can be reached quickly.

[visual:

Checklist for SignalWire

- ☑ ~~Strong documentation~~
- ☑ ~~Thriving developer community~~
- ☑ ~~responsive customer support~~
- ☑ ~~Depth of knowledge]~~

][visual: Find your best new employee
Friendly, helpful, always in a good mood +
Doesn't need breaks, Available 24/7 +
Depth of knowledge

SignalWire AI Agent is designed for business leaders who are focused on delivering best-in-class customer experiences. No minimums required; pay only for what you need. Whether you're a small business looking to scale or a larger business aiming to cut costs, SignalWire AI Agent can streamline your call center operations and improve your customer experiences.

Buttons:

View demo

Learn more about SignalWire AI Agent

CEO
Anthony Minessale
Real-time communication technologies, such as PSTN, SIP, and WebRTC have revolutionized the way we connect with each other. However, managing and customizing these communication channels can be a complex task.

That's where SignalWire Markup Language (SWML) comes in. SWML is a JSON representation of a call flow that instructs servers how to behave during real-time communications. It is similar to traditional CPaaS solutions, but offers more flexibility and power thanks to the underlying technology stack.

In this post, we will explore the features and benefits of SignalWire SWML, discuss its advantages over older CPaaS solutions, and provide some use case examples to demonstrate its capabilities.

Understanding SignalWire SWML

SWML is a powerful tool that allows businesses to define their call flows and customize their Interactive Voice Response (IVR) systems. It provides a set of instructions for various IVR functions, enabling businesses to create AI-powered virtual assistants, gather information from callers, and execute remote procedure calls for instant processing.

One of the key advantages of SWML is its integration with FreeSWITCH, which offers a wide range of features and flexibility. With SWML, businesses can build AI-powered IVRs that can understand natural language, interact with callers, and perform complex tasks. This opens up a world of possibilities for businesses to create personalized and efficient communication systems.

Key Features of SignalWire SWML

Visual Editor: Call Flow Builder

SignalWire provides a visual editor called the Call Flow Builder, which allows users to create and customize their call flows using SWML. The Call Flow Builder

simplifies the process of designing complex call flows by providing a drag-and-drop interface. Users can easily add and configure various IVR functions, define AI behaviors, and create seamless communication experiences.

AI-Powered IVRs

SWML enables businesses to build AI-powered IVRs that can understand and respond to natural language. By configuring a language model, businesses can create virtual assistants that can gather information from callers, execute remote procedure calls, and provide instant responses. This opens up new possibilities for businesses to automate processes, improve customer service, and enhance overall communication experiences.

Spaces and Resources

SignalWire is divided into spaces, where each organization can create their own space and define API users and subscribers. Subscribers and public callers can connect to a SignalWire space through mapped addresses, similar to extensions in a PBX. These addresses lead to resources created by the space owner and represent various entities that can be called.

Resources can include scripts powered by SWML, multiparty conferences (rooms), authenticated users using mobile apps or SIP registration (subscribers), AI agents defined in the space, call queues, and more. SWML can route calls to different resources, allowing businesses to create complex call flows and provide a seamless communication experience.

REST APIs and Provisioning

SignalWire implements everything as REST APIs, allowing businesses to provision and manage their communication resources programmatically. Users can create, update, and delete resources using REST calls, either directly or through a remote service. This provides businesses with the flexibility to integrate SignalWire into their existing systems and automate various communication processes.

Advantages of SignalWire SWML over older CPaaS solutions

While older CPaaS solutions have been a popular choice for building IVRs and managing call flows, SignalWire SWML offers several advantages that make it a more modern and powerful solution.

JSON-Based Format

SWML is based on JSON (JavaScript Object Notation), a lightweight data-interchange format. Unlike older CPaaS solutions, which use XML, SWML's JSON-based format is more modern and aligns with the industry's shift towards using JSON for data exchange. Programming languages widely support JSON, and JSON provides a more flexible and readable format for defining call flows.

Complete Call Flow in a Single Document

SWML allows users to express a complete call flow as a single document. This means that all the instructions and configurations for a call flow can be contained within a single SWML document, eliminating the need to fetch additional information from external sources. This makes it easier to manage and transfer call flows between different sections within the document. This also allows more low-code no-code possibilities for simple call flows.

Powered by FreeSWITCH

Like much of SignalWire, SWML is powered by FreeSWITCH, an open-source communication platform built by the founders of SignalWire and known for its flexibility and extensive feature set. This power is harnessed and made available to everyone in a secure, easy-to-understand cloud interface.

Unified Audio and Video Support

SignalWire provides support for both audio and video calls and APIs over a single media service. This means that businesses can seamlessly transition between audio and video communication channels without needing separate services or configurations. This unified support for audio and video sets SignalWire SWML apart from traditional CPaaS solutions and makes it a more versatile solution.

Use Case Examples

To better understand the capabilities of SignalWire SWML, let's explore some use case examples:

AI-Powered Virtual Assistant

Imagine a business that receives a high volume of customer calls and wants to automate the process of gathering information from callers. With SignalWire SWML, the business can create an AI-powered virtual assistant that can understand natural language and interact with callers. The virtual assistant can ask questions, collect information, and execute remote procedure calls to retrieve relevant data. This not only improves efficiency, but also enhances the overall customer experience.

Call Queues and Routing

A customer support center often receives a large number of incoming calls, and it's crucial to handle them efficiently. SignalWire SWML allows businesses to create call queues where callers can get in line to talk to a support representative.

SWML can be used to route calls to the appropriate queue based on various criteria, such as caller preferences, agent availability, or the nature of the inquiry. This ensures that callers are connected to the right person and reduces wait times.

Multiparty Conferences

SignalWire SWML enables businesses to create multiparty conferences, also known as rooms. These conferences can be used for various purposes, such as team meetings, webinars, or virtual events.

SWML allows businesses to define the behavior of the conference, such as who can join, how participants are muted or unmuted, and what actions can be performed during the conference. This provides businesses with the flexibility to create customized conference experiences tailored to their specific needs.

Secure Data Collection

In certain scenarios, businesses may need to collect sensitive information from callers, such as personal identification numbers or credit card details. SignalWire SWML provides a secure way to handle such data by allowing businesses to create secure subroutines within the call flow. These subroutines can be used to collect and verify sensitive information without involving the AI-powered virtual assistant. This ensures that sensitive data remains private and secure.

Security and Privacy

SignalWire takes security and privacy seriously. All APIs use encryption, ensuring that data is transmitted securely between the client and the server. In telephony signaling, callers can only interact via voice or by dialing digits, limiting the potential for unauthorized access. In web applications, data is sent over a secure connection to the server, protecting it from interception or tampering.

When it comes to AI-powered interactions, SignalWire SWML provides a mechanism called SWAIG (SignalWire AI Gateway) to handle sensitive data collection. SWAIG allows businesses to section off any sensitive data collection, ensuring that it happens in a secure and isolated environment.

For example, businesses can define a SWAIG function to authenticate callers, which can be executed using SWML. This approach keeps sensitive data separate from the conversation and minimizes the risk of data breaches.

Conclusion

SignalWire SWML is a powerful tool that empowers businesses to create customized and efficient communication systems. With its JSON-based format, integration with FreeSWITCH, and support for AI-powered virtual assistants, SWML offers a modern and flexible solution for managing call flows.

Whether it's building AI-powered IVRs, creating multiparty conferences, or routing calls to different resources, SWML provides businesses with the tools they need to deliver exceptional communication experiences.

SignalWire SWML is poised to become the go-to choice for building robust and scalable communication systems. Its ease of use, extensive feature set, and focus on security make it an ideal solution for businesses of all sizes.

So, if you're looking to take your communication systems to the next level, consider SignalWire SWML. With its powerful capabilities and seamless integration with FreeSWITCH, SWML is the future of real-time communications.

To learn more about SignalWire SWML and how it can benefit your business, sign up for a free test account, and bring any questions you have to our community Slack or forum.

Call Fabric is the effort to make a horizontal scaling self-serve communication platform where users can create and arrange communication pathways on-demand. This encompasses all the features of an IP PBX, UCaaS, Soft-Switch and makes the features available as high-level primitives. In programming, a primitive are the building blocks you have to work with when creating the code. Since we are trying to reduce complexity, the primitives in Call Fabric, known as Resources, are features deconstructed to the ideal puzzle pieces that can be arranged in endless combinations. Rather than spending time programming these very, very complex concepts, APIs are just used to create and position resources.

A resource is simply something that "can be called," meaning an audio/video/text stream can be established to it.

Initial resources include:

Scripts and markup: (SWML/Laml/Relay) Create a program that can interact and control the call using logic. This is the basic building block of SignalWire V1 (before call fabric)

Rooms: A room is a multi-user conference where many callers can talk over audio or video.

Subscribers: Subscribers represent a registered user. The other end of a subscriber is a mobile app or private phone that is registered to the platform. A private line.

AI Agents: A digital employee powered by AI that can interact with the caller, transfer calls, do actions in the real world.

Callflow Builder: A visual representation of a call flow that processes calls and routes them based on logic such as time of day and other criteria.

These resources can be created and managed from the SignalWire Dash and made accessible by assigning it a phone number, web address, or SIP address, making it publicly available.

EXAMPLES

Add 2 subscribers give them both phone numbers; they can call each other via the SignalWire Network or call any valid phone number on the public telephone network. A normal mobile phone or landline can call the subscribers by dialing their number.

Create a room and give it a web address and a phone number. The subscribers can find the room and join it from the mobile app. Guests could go to the web address in the browser to reach the room, and mobile and landlines could dial the number to get to the room.

Make a call flow builder that sends calls during business hours to both of the subscribers at the same time, and when it is outside business hours, forward the calls to an AI Agent who takes a message and sends it to you. Give that call flow builder a phone number and a web address, and allow customers to dial the number or visit the web address embedded in the corporate website to place the call.

SignalWire AI Telecom Agent Market Context

High-Level Summary

Core Market Challenges:

- Building enterprise-grade omnichannel conversational AI agents that can handle complex tasks across multiple channels (voice: PSTN, SIP, WebRTC; text messaging: SMS/MMS/RCS/OTT messengers/web chat; video) while maintaining ultra-low latency (<500–600 ms per turn).

Key Requirements:

- **Low Latency at Scale**: Ensures lifelike conversations, especially when interfacing with third-party APIs and databases.
- **Task-Oriented AI Agents**: Keeping AI agents on task while they complete complex goals.
- **LLM Hallucination Avoidance**: Ensuring brand alignment and factual grounding.
- **Omnichannel Integration**: Handling multi-channel conversations across different communication protocols without losing context.
- **Complex State and Context Management**: Maintaining real-time transcription, summarization, and memory for long and multi-session conversations.
- **Real-Time API & Database Integration**: Ensuring reliable interactions with CRMs, ordering systems, and knowledge bases.

Competitive Landscape

- Major telecom platforms struggle with AI + telecom pipeline integration.
- Most competitors suffer from **1–3 seconds of roundtrip latency**.
- Twilio's "ConversationRelay" attempts to address this, but requires developers to manage WebSockets and LLM workflows manually.
- Other voice AI vendors rely on third-party telecom connectivity (Twilio, Vonage), which introduces latency and scaling issues.

SignalWire's Advantage

- **"Call-center grade" telecom orchestration** with **native LLM, TTS, and STT integration**.
- **Multi-threaded, bare-metal telecom<>LLM pipeline** delivers **ultra-low latency** AI agents across voice, text messaging, and video.
- **Unified platform (SWML, ai.params, SWAIG)** to orchestrate telecom, structure conversations, define AI agent roles, goals, and toolsets.

- **Abstracts away complexity** related to concurrency, scaling, and multi-channel integration, allowing developers to focus on differentiation.
- **Real-time transcription, summarization, and translation** for memory and complex conversational flows.
- **Integrated with multiple TTS/STT providers** and **direct OpenAI API interface**.
- **Enterprise-grade features**: Global edge network, compliance, logging, analytics, and security.

Biggest Challenges in AI + Telecom

1. Maintaining Ultra-Low Latency

- **Why It Matters**: Conversational AI needs near **500 ms round-trip latency** for natural speech.
- **Scaling Complexity**: Handling thousands to millions of concurrent AI calls requires optimized infrastructure.
- **API Delays**: Third-party integrations (CRMs, databases) add response time.

2. Avoiding LLM Hallucinations

- **Fact Grounding**: Modern LLMs can generate plausible but incorrect responses.
- **Brand Consistency**: AI answers must align with company guidelines and prevent drift.

3. Managing Context & State

- **Memory Retention**: AI must “remember” prior conversation details over extended interactions.
- **Multi-Session Handling**: Ensuring AI retains context across multiple calls.
- **Context Loss Prevention**: Requires detailed tracking of user data and preferences.

4. Third-Party System Integration

- **Essential for Task Completion**: AI must interact with CRMs, ticketing, ordering, and knowledge bases.
- **Unpredictable API Latency**: External API response times vary and can cause delays.
- **RAG Database Performance**: Ensuring accurate, real-time knowledge retrieval.

5. Omnichannel AI Across PSTN, SIP, WebRTC, and Messaging

- **Different Protocols, Different Constraints**: AI must dynamically adjust to voice, text, and video environments.
- **Unified Conversational Memory**: Conversations need seamless handoff across different channels.

SignalWire’s Solution

- **Integrated Telecom + AI Stack**: No need for third-party voice platforms, reducing latency.
- **SWAIG (SignalWire AI Gateway)**: A complete AI framework with built-in orchestration.
- **Multi-Channel AI Adaptation**: Works seamlessly across PSTN, WebRTC, SIP, and messaging without breaking session continuity.
- **Enterprise-Ready Deployment**: Built-in compliance, logging, and analytics for businesses deploying AI at scale.

Conclusion

SignalWire's AI-powered telecom platform overcomes the limitations of traditional CPaaS and UCaaS by embedding AI directly into the media stack. With a focus on ultra-low latency, omnichannel adaptability, and real-time intelligence, SignalWire enables businesses to build scalable, enterprise-grade AI communication solutions.

Marketing Copy for Developers – SignalWire's Low-Latency, Low-Bandwidth AI Communication Platform

Efficient, Scalable, and Seamless: How SignalWire Optimizes AI Communication for Developers

---

1. Introduction: The Challenges of Applied AI Communication

The Problem:

In AI-powered communication, **most systems sacrifice efficiency**, leading to **higher costs and frustrating delays**. SignalWire redefines the standard with solutions **optimized for real-world performance**.

Context:

Many platforms rely on streaming **raw audio via WebSockets**, which causes:

- **1–3 second latencies**
- **High bandwidth costs**
- **Heavy system loads**

These inefficiencies **break the flow of natural conversation** and **frustrate developers**.

Problem Statement:

Traditional architectures are **fragmented and inefficient**, resulting in:

- **Suboptimal performance**
- **Rising costs** as usage scales

---

2. SignalWire's Solutions: Cutting Costs and Latency While Improving Performance

Sub-500ms Latency

- **The Problem:** Most platforms have **1–3 seconds of latency** due to fragmented systems and reliance on external CPaaS services.
- **SignalWire's Solution:**

    - AI is **natively integrated** into SignalWire's **media stack** (eliminating third-party processing delays).
    - The result: an **average latency of ~500ms**, ensuring **smooth, real-time interactions**.

Internalized Raw Audio Handling

- **The Problem:** Streaming **raw audio over WebSockets** increases bandwidth and delays.
- **SignalWire's Solution:**
    - **Processes raw audio within its media stack**, minimizing network overhead.
    - Uses **low-bitrate encoded audio** for communication.

Reduced Bandwidth Costs

- **The Problem:** Streaming raw audio **exceeds 64kbps per stream**, driving up costs.
- **SignalWire's Solution:**
    - **Uses low-bitrate encoding** externally while processing **high-quality raw audio internally**.
    - **Significantly reduces bandwidth needs** without sacrificing quality.

Unified Media and AI Workflows

- **The Problem:** Many platforms treat AI and media handling as separate processes, leading to **latency and redundancy**.
- **SignalWire's Solution:**
    - AI is **embedded directly into the communication platform**.
    - Provides **seamless speech-to-text (STT), AI processing, and text-to-speech (TTS) workflows**.

Real-Time Orchestration and Scalability

- **The Problem:** Scaling traditional AI systems often **introduces delays** and increases **system load**.
- **SignalWire's Solution:**
    - **Built on horizontally scalable Call Fabric**, enabling **high call volumes** without performance loss.

Data Security and Compliance

- **The Problem:** AI workflows handling **sensitive data** risk breaches and compliance failures.
- **SignalWire's Solution:**
    - **End-to-end encryption** and **metadata tokenization** for **secure, compliant operations**.
    - Adheres to **GDPR and HIPAA standards**.

---

3. Why Developers Choose SignalWire

Low Latency, High Performance

- **Achieves an average latency of ~500ms**, significantly faster than industry norms (**1–3 seconds**).

Cost-Effective Operation

- **Reduces bandwidth costs** through **internal raw audio processing** and **low-bitrate encoding**.

Native AI and Media Integration

- **Eliminates inefficiencies** by **embedding AI directly into the media stack**, reducing dependence on third-party CPaaS.

Global Communication Platform

- **Combines AI, voice, video, and messaging** into a single **low-latency system**.

Developer-Friendly APIs

- Tools like **Call Flow Builder**, **low-code options**, and **modular APIs** enable **rapid deployment and customization**.

---

4. SignalWire's AI Integration with Communication Systems

AI in Voice Communication Use Cases

- **The Problem:** Voice AI is **closely tied to CCaaS and UCaaS systems**, where **seamless integration is critical**.
- **SignalWire's Solution:**
    - AI capabilities **bundled directly** into its **CPaaS platform**.
    - Built-in support for **STT, TTS, and real-time AI orchestration**.

Conclusion

SignalWire is **pioneering ultra-low-latency AI communication**. By eliminating inefficiencies in traditional CPaaS platforms, **developers can build smarter, faster, and more cost-effective AI-driven communication solutions**.

Features Available in SignalWire but Not in FreeSWITCH

1. **RELAY Call Control System**
    - A WebSocket-based system for real-time call control, handling events and enabling dynamic programming in any language.
2. **SignalWire Markup Language (SWML)**
    - Declarative scripting in JSON/YAML for defining call flows, AI agents, and dynamic routing without external controllers.
3. **Programmable Unified Communications (PUC)**
    - A unified model combining CPaaS, UCaaS, and CCaaS into a modular, programmable infrastructure.
4. **AI-Driven Workflows**
    - Native AI integrations for:
        - Real-time transcription
        - Live translation
        - AI Agent orchestration
        - Retrieval-Augmented Generation (RAG) for data-grounded interactions
5. **Call Fabric**
    - A resource-centric approach that allows composing telecom workflows dynamically using:
        - Rooms (audio/video conferencing)
        - AI agents
        - Subscribers (user accounts for SIP/WebRTC registration)
        - Scripts (SWML and cXML)
        - Gateways (SIP trunking and interconnection)
6. **Horizontal Scalability**
    - Seamless global scaling with edge networks and geographic redundancy.
7. **Integrated AI Agents**
    - Framework for building and deploying natural-language AI agents for conversational interfaces and task automation.
8. **Omnichannel Communication**
    - Unified support for voice (PSTN, SIP, WebRTC), messaging (SMS, MMS, RCS), and video in a single platform.
9. **Dynamic Context Switching**
    - Real-time reconfiguration of workflows during active calls, e.g., transferring calls or initiating AI actions without interruptions.
10. **Live Transcription and Translation**
    - Built-in tools for transcribing and translating calls in real time, enabling multilingual support.
11. **Low-Latency Media Handling**
    - AI embedded directly in the media stack for sub-500ms latency in real-time communications.
12. **Data Security and Compliance**

    - Built-in support for HIPAA, GDPR, PCI and other compliance standards, with features like tokenized metadata and end-to-end encryption and secure payment collection.
13. **Resource Routing Engine**
    - Intelligent routing of calls and messages to resources based on predefined rules or dynamic input.
14. **Low-Code and No-Code Tools**
    - Call Flow Builder and other visual tools for creating workflows without deep programming knowledge.
15. **REST/RPC APIs**
    - Dynamic control of communication resources via APIs for tasks like call spawning, recording, and toggling workflows.
16. **Twilio-Compatible cXML**
    - Backward-compatible syntax for Twilio's TwiML, allowing easier migration of existing applications.
17. **Storage-Backed Recordings**
    - Integrated support for recording and securely storing media streams.
18. **Unified Addressing System**
    - Simplifies resource access with universal addresses across SIP, WebRTC, and PSTN.
19. **Composable Telecom Infrastructure**
    - Breaks down communication systems into reusable building blocks for scalable, flexible deployment.

Fundamental Differences Between SignalWire and FreeSWITCH

1. Abstraction vs. Granularity

- **SignalWire** abstracts the complexities of telephony into higher-level programmable interfaces, enabling rapid development without managing underlying protocols.
- **FreeSWITCH** offers granular control over telephony, requiring deep domain knowledge for configuration and operation.

2. Modularity and Composability

- SignalWire's Call Fabric enables modular design, treating communication elements as reusable resources.
- FreeSWITCH operates as a monolithic telephony engine, focused on single-server configurations.

3. Scalability

- SignalWire supports horizontal scaling with edge networks and global redundancy, ensuring reliability at hyper-scale.
- FreeSWITCH is inherently single-instance and requires external efforts to scale.

4. AI Integration

- SignalWire embeds AI directly into its media stack, enabling real-time transcription, translation, and natural-language interaction with sub-500ms latency.
- FreeSWITCH lacks built-in AI capabilities, requiring third-party tools for similar functionality.

5. Developer Enablement

- SignalWire provides developer-friendly APIs, low-code tools, and a declarative scripting language (SWML).
- FreeSWITCH relies on developers' expertise to configure and extend its functionalities through external scripts or modules.

6. Unified Platform

- SignalWire combines CPaaS, UCaaS, and CCaaS capabilities into a single, programmable platform.
- FreeSWITCH is focused on VOIP and media processing, leaving application-level functionality to be built externally.

7. Time to Market

- SignalWire's abstractions and ready-to-use components drastically reduce time-to-market for communication solutions.
- FreeSWITCH requires significant setup, configuration, and integration efforts.

8. Integration

- SignalWire natively supports omnichannel communication and integrates seamlessly with third-party tools like Salesforce.
- FreeSWITCH offers protocol-level flexibility but lacks out-of-the-box omnichannel support.

SignalWire reimagines telephony as a modular, scalable, and programmable system, enabling businesses to innovate faster and adapt to modern communication needs. FreeSWITCH, while powerful, operates more like a foundational engine that requires extensive customization to achieve similar results.

Model Context Protocol: Evolving to Work Together with Real-Time Media Plumbing

Introduction

At SignalWire, we've spent years designing robust, real-time communications infrastructures, notably FreeSWITCH and the SignalWire Cloud. Before the introduction of the Model Context Protocol (MCP), we developed the SignalWire Markup Language (SWML)-a structured, YAML- and JSON-based static markup language enabling dynamic interactions across voice calls, video conferences, messaging, and AI-driven conversational agents. Additionally, we created RELAY, a WebSocket-based RPC call control protocol specifically designed to manage live calls remotely, ensuring low latency, state synchronization, and efficient error recovery.

For Conversational AI IVRs, SWML combined with our SignalWire AI Gateway (SWAIG) allows applications to intuitively interact with REST APIs and execute structured RPC-like calls. Our infrastructure explicitly supports robust, stateful real-time WebSocket connections, dynamic AI-driven interactions, clear error handling mechanisms, explicit session and state management, and well-defined message framing and flow control.

Evaluating the MCP Specification

When reviewing the MCP specification, we recognized and appreciated its ambition to standardize interactions between Large Language Models (LLMs) and external services. The schema demonstrates comprehensive thinking around defining prompts, resources, and interactions. However, our extensive experience with protocols like SIP, WebRTC, RTP, and high-volume real-time call control highlighted potential challenges in MCP's transport layer that could limit practical scalability and reliability in demanding production environments.

Limitations of MCP’s Current Transport Approach

MCP currently uses "streamable HTTP," implemented via long-lived HTTP POST requests with chunked JSON responses. This approach simplifies initial integration and is well-suited for prototypes or demonstrations. Nonetheless, we identified several critical scalability limitations:

- **Connection fragility:** Persistent HTTP streams can become unstable, causing reliability issues at scale.
- **Fan-out difficulties:** Managing multiple simultaneous streaming consumers becomes increasingly complex.
- **Lack of multiplexing:** Limits the efficiency and scalability of concurrent interactions.

- **Insufficient error handling and recovery:** Current methods lack comprehensive strategies for session resumption and connection recovery.
- **Framing ambiguity:** Absence of standardized framing complicates parsing and can degrade reliability under heavy usage.

Real-World Lessons from SIP, RTP, WebRTC, and High-Volume Call Control

Deployments involving SIP, RTP, WebRTC, and our high-volume call control protocols demonstrate that large-scale, real-time systems significantly benefit from robust transport technologies. Lessons learned include:

- **Multiplexing:** Essential for efficient handling of thousands of simultaneous streams.
- **Explicit session management:** Robust session and connection resumption strategies are critical for maintaining high availability.
- **Built-in flow control:** Prevents overload and ensures stable system performance.
- **Standardized message framing:** Simplifies parsing, reduces errors, and ensures reliability at scale.

These proven transport mechanisms inherently provide the necessary reliability and scalability required by demanding real-world scenarios.

Recommendations for Transport Enhancements

To address these identified limitations, we recommend future MCP iterations explicitly support or clearly define alternative transports, keeping practical considerations in mind:

- **WebSockets:** While a logical option due to their real-time capabilities, scalability considerations should be taken into account.
- **Standard REST:** A stateless REST-based approach could offer simplified client access and reduce server-side complexity, greatly enhancing scalability.
- **HTTP/2 or QUIC:** Forward-thinking options providing efficient multiplexing, built-in flow control, and significantly reduced latency, essential for quick response times in LLM-based applications.
- **Explicit session management:** Strategies for robust connection establishment, error recovery, and resumption.
- **Clear framing standards:** Binary or line-delimited JSON framing to enhance parsing efficiency and reliability.

Quick response time is paramount when providing tools to LLM-based applications. Adopting these enhancements would elevate MCP from its current prototype-friendly form to a fully production-ready protocol capable of supporting enterprise-level scalability, performance, and reliability.

Future Compatibility with SignalWire’s Infrastructure

The structured format of SWML and the extensibility of our SWAIG functions could naturally integrate as an MCP client if the transport and scalability concerns were adequately addressed. SignalWire's voice AI kernel operates as a central hub, seamlessly connecting telecom features with AI capabilities, coordinating voice interactions, listening, and cognitive processing via an integrated LLM. Developers can define conversational behavior using prompts and callbacks that map to SWAIG functions, enabling dynamic modifications of bot behavior, context management, and feature expansions without implementing each functionality individually.

Given this existing architecture, integrating MCP would be straightforward and highly beneficial. MCP could effectively extend SignalWire's robust conversational engine by providing additional remote services that could be effortlessly plugged into the platform. Such integration would amplify interoperability, further enhancing the adaptability and scalability of SignalWire's AI-driven conversational experiences.

SignalWire's existing infrastructure is well-equipped to support MCP's JSON-RPC schema, offering broader interoperability while preserving our native protocols' robustness and performance.

Conclusion

MCP's vision aligns closely with ours-enabling powerful, standardized interactions between AI-driven agents and external resources. By incorporating transport-layer enhancements informed by our extensive experience with SIP, RTP, WebRTC, and high-volume call control, MCP can mature into a reliable, scalable protocol suitable for demanding real-time applications at enterprise scale. We look forward to MCP's evolution and strongly encourage proactive incorporation of these recommendations to ensure broad industry adoption and effective deployment.

SignalWire, the pioneers of Programmable Unified Communications (PUC) and founders of the FreeSWITCH open-source project, today announced the launch of their new conversation AI and voice integration platform, signalwire.ai. This platform moves beyond typical “bolted-on” AI tools by offering developers a full-stack solution, encompassing everything from transport to real-time intelligence, ensuring audio, routing, and AI operate in sync.

Traditional IVRs often frustrate users with their reliance on rigid scripts and slow processing times. In contrast, SignalWire’s AI Agents offer dynamic, natural voice interactions with ultra-fast response times with an average of 500 milliseconds. By integrating AI directly into the telecom media stack, voice data remains securely within the network, enhancing both performance and security, and ensuring seamless user experiences.

"This isn’t just AI added to phones; it’s AI built directly into the core telecom stack," said Anthony Minessale, CEO of SignalWire. "We designed this platform for developers and contact centers to create powerful, intelligent agents that function as fully programmable digital employees ready for real-time performance at scale."

The platform’s composable architecture ensures compatibility with PSTN, SIP, WebRTC, mobile apps, and IP-based chat systems, enabling developers to rapidly create and deploy AI Agents across various channels. Potential applications include intelligent intake agents, multilingual assistants, contact centers, concierge bots, post-call surveys, and advanced RAG-connected knowledge agents supporting sales or customer support teams.

SignalWire’s AI Agents are now available for immediate use through the SignalWire Dashboard, as well as via our comprehensive APIs and SDKs. They require no additional hosted infrastructure and fully integrate with other SignalWire primitives, including Rooms, Gateways, and Subscribers.

About SignalWire

SignalWire is a full-stack communications platform optimized for low latency, real-time control, and developer agility. Built by the team behind FreeSWITCH, it

offers a seamless environment for programming voice, video, messaging, and AI applications—without the overhead of legacy systems or vendor sprawl. From infrastructure to interaction, everything is unified, giving developers the power to build responsive, event-driven communications at any scale.

Salesforce Q&A - SignalWire Overview

Describe the origins of SignalWire and your view on your competitive position

SignalWire emerged from FreeSWITCH, an open-source project that transformed telecom by enabling scalable, flexible, and programmable communications. SignalWire builds on this foundation to give businesses full control over their communication workflows without the complexity of managing hardware.

SignalWire leads the **Programmable Unified Communications (PUC)** category, combining the programmatic flexibility of CPaaS with the unified capabilities of UCaaS and CCaaS. This addresses key challenges in traditional solutions:

- **Rigid on-premise systems** that are expensive and hard to scale.
- **Fragmented cloud services** leading to poor integration.
- **Resource-intensive DIY infrastructure**.

SignalWire's competitive advantage comes from its **composable telecom infrastructure** approach, where communication elements are modular, reusable resources that can be programmatically assembled. Key differentiators include:

- **Programmability** through APIs, webhooks, and RPCs for real-time workflow changes.
- **Low latency** through native media stack integration.
- **Global scalability** with geographic redundancy.
- **Cost efficiency** through consolidated tools and automation.
- **Developer-friendly open standards** (SIP, REST, WebRTC).
- **SWML for advanced call flows and dynamic workflows**.

This enables businesses to build precisely what they need without vendor lock-in or fragmented tools while maintaining enterprise-grade reliability and scale.

Describe how FreeSWITCH and SignalWire interact; do open-source developers convert to paid SignalWire customers?

SignalWire serves as the sponsor for the **FreeSWITCH** open-source project. SignalWire uses an enterprise version of FreeSWITCH loaded with proprietary features as one key component of its platform. FreeSWITCH is mostly used by others in an **on-premise** environment, so as digital transformation occurs, SignalWire is able to offer a **more modern and flexible cloud-native solution**.

Describe how customers are using your products and key customer use cases

SignalWire is used by **developers** to build AI agents, call flows, and other solutions. Calls can originate from **PSTN, SIP, WebRTC, and other sources**, accessing and sharing the same resources. SignalWire is also used by **enterprises** to build their own custom communication solutions.

When a developer builds an AI agent or call flow using SignalWire, do they still need a traditional CCaaS, UCaaS, or CPaaS solution?

Not in most cases. SignalWire focuses on providing the **core real-time communications (RTC) infrastructure** and high-level primitives needed to build UCaaS and CCaaS applications.

Rather than competing with UCaaS/CCaaS solutions, **SignalWire provides the underlying infrastructure** that enables these applications to be built. Through standards-based protocols (**SIP, WebRTC, etc.**) and APIs, SignalWire can integrate with any system while handling complex real-time media, routing, and scaling requirements.

High-level application features (like CRM integration and workforce management) can be integrated through SignalWire's **programmable interfaces**, including webhooks, REST APIs, and real-time events. This allows customers to:

- Build custom **UCaaS/CCaaS** solutions directly on SignalWire.
- Integrate **SignalWire capabilities** into existing applications.
- Use **SignalWire as the communications backbone** while leveraging other tools.
- **Mix and match** components through **open standards and APIs**.

Where does SignalWire fit in a CCaaS/UCaaS solution stack?

SignalWire **sits between the customer and the underlying communications infrastructure**. It provides the **core real-time communications infrastructure and high-level primitives** needed to build communication paths for **UCaaS and CCaaS applications**.

How do your products integrate with systems of record such as Salesforce?

SignalWire's **APIs and webhooks** allow for seamless integration with Salesforce and other systems of record. This enables:

- **Dynamic call and message routing** based on customer data and behavior.
- **Post-call data integration** into any backend system.

Product Portfolio: AI Agents, SWML, APIs, Call Flow Builder, etc.

Describe the voice applications that can be built on SignalWire

SignalWire supports a wide range of voice applications, including:

- **Call forwarding**
- **Answering machine detection**
- **Interactive Voice Response (IVR)**
- **Virtual assistants**
- **AI-driven customer interactions**

Using **SWML (SignalWire Markup Language)** and APIs, developers can create flexible, programmable call flows and integrate AI for advanced automation.

History:

We started FreeSWITCH because we wanted to solve *our* own challenging problems.

We could not find a stable, scalable way to implement our 2004 call center as a service product. DIY solutions were a collection of random tools with constant stability and maintenance issues. Premade solutions were massively expensive and still required physical data centers anyway.

We built FreeSWITCH on our vision of solving those problems, made a community, and used it for insight. This took us well beyond the original goal. FreeSWITCH became so ubiquitous that the world made many vertical products with it that generate billions of dollars a year.

SignalWire POV:

At SignalWire, we empower businesses with easy-to-use and customizable cloud communications tools. **Programmable Unified Communications** supplants one-size-fits-all solutions, sparking innovation and growth.

Telecommunications (technology-assisted human communication) is one of the most important technologies humankind has ever developed. It has evolved from written messages on paper carried by messenger to a vast array of devices and mediums, such as telephones and computers, using audio and video.

Telecommunications have pushed businesses forward with every advancement as long ago as the telegram. The more information can be easily shared or discussed, the better it is for buyers and sellers alike. Modern telephone networks and the Internet have pushed this concept even further. Industries like entertainment, health care, education, and customer service have all benefited greatly.

Before SignalWire began, our founding team spent the better part of 2005-2015 helping to innovate the concept of IP Communications at scale with disruptive open-source technology.

Unfortunately, in recent times, there has been a massive slowdown in the evolution of this technology. Large providers often sacrifice innovation to avoid cannibalizing profits, which leads to rigidity and suffering for end users. Innovation was stifled, and this short-sightedness caused a traffic jam ahead of us.

We started SignalWire because we wanted to take our pioneering communications technology that took decades to develop and present it in a digestible format that allows customers to deploy exactly what they want, freeing them from the obstacles of time, cost, and resources.

We fundamentally believe in our dedication to removing obstacles and paving the way forward for each other as well as our customers. That mentality drives everything we do and empowers us to innovate and grow as a global community.

There is a crisis in the communications industry.

Strategies to execute the transformation to cloud communications infrastructure often present three significant pitfalls. These can lead to a reluctance to commit fully or to avoid the transformation entirely as everyone around them evolves.

The first pitfall comes from trying to deploy on-premise solutions. They are the most rigid of all the options and by far the most costly. Physical hardware becomes obsolete in a short time, causing endless refresh cycles. The hardware will likely fail, causing late-night support issues that stop businesses cold. Most of the hardware requires expensive support contracts per unit, and the businesses are at the mercy of the vendors.

The next pitfall is encountered by businesses who choose to use a series of end-to-end applications. They are forced to use multiple products, some repetitive in nature, getting phone service with one company, customer call centers with another, and video and conferencing service from another. Nothing is connected. These application providers are more focused on the status quo than moving their customers forward. This leads to a loss in productivity and a lousy customer experience for end-users.

The last common pitfall plagues businesses that try the other option, which is to build something proprietary. They face a litany of challenges they did not anticipate. There are huge decisions to make, and most evolve around going well outside their core competencies. This comes with the burdens of hosting and maintaining complex services and staffing developers with rare and elusive skill sets. In the end, many businesses scrap the project after wasting years of money and opportunity costs. The ones who managed to get something working often are too late and are plagued with maintenance and other distractions.

SignalWire is uniquely qualified to provide businesses with a path to cloud communications that reduce and eliminate distractions and obstacles while providing access to all of the features that were previously only available in expensive end-to-end solutions coupled with ever-evolving advanced features not available from anywhere else.

We see a world where all businesses can set up and own the entire chain of communications to and from their company, combining traditional phone-number-based communications with mobile app and web-app-based paradigms. We make that a reality by focusing on solving the most challenging infrastructure and feature deployment problems and presenting them to customers as point-and-click options. Integrations with SaaS tools and simple scripting unlock everything in between.

This vision benefits not only our customers but also our customers' customers. Everyone universally despises the idea of calling somewhere and dealing with ancient auto-attendants that hang up on you or send you in circles or being asked for your account number by no less than five people while still not getting to someone who can help. Instead, powerful AI Agents

can gather information and perform tasks to solve problems and narrow down the number of calls that end up on hold. Clear lines of communication from corporate websites, phone lines, and mobile applications can coalesce in one place, be easy to use, easy to extend, and always on.

SignalWire sets itself apart in the cloud communication landscape by offering highly customizable solutions, a stark contrast to the rigid packages offered by many. Our innovative use of advanced technologies, like AI, and an accessible, open-source foundation enable us to meet diverse business needs more effectively. This approach not only breaks from industry norms but also addresses the evolving communication challenges faced by businesses today.

At SignalWire, we believe that by making powerful cloud communications tools and features easily programmable, configurable, and accessible to all businesses without the overhead of one-size-fits-all solutions, the result will be prosperous innovation and growth for the entire ecosystem. **Programmable Unified Communications** unlock a new era of simplicity and control for everyone.

Solution:

SignalWire Call Fabric is the implementation of **Programmable Unified Communications** where users can create and arrange communication pathways on-demand. This encompasses all the features of an IP PBX, UCaaS, Soft-Switch and makes the features available as high-level primitives. In programming, a primitive are the building blocks you have to work with when creating the code. Since we are trying to reduce complexity, the primitives in Call Fabric, known as Resources, are features deconstructed to the ideal puzzle pieces that can be arranged in endless combinations. Rather than spending time programming these very, very complex concepts from the ground up, APIs are just used to create and position resources.

A resource is simply **something that can be called**. meaning a 1 or 2 way audio/video/text stream can be established to it.

Initial resources include:

Scripts and markup: (SWML/RELAY/cXML) Create a script or program that can interact and control the calls by routing, collecting data, invoking digital employees with AI Agents.

Rooms: A room is a multi-user conference where many callers can talk over audio or video. There are live prieviews of the room and the ability to move around between different rooms.

Subscribers: Registered user accounts for end-users to have a virtual phone line into the system that can find and call other resources and gateway to the traditional phone network. These subscribers can register with desk phones, the SignalWire mobile app or custome clients that can be developed with our SDK.

AI Agents: A digital employee powered by AI that can interact with the caller, transfer calls, and do actions in the real world.

Callflow Builder: A visual tool to create voice menus, call routes, and applications that can be deployed instantly and called at scale.

All of These resources can be created and managed from the SignalWire Dash and made accessible by assigning them a phone number, web address, or SIP address, allowing unification from all mediums to call and interface with the system.

EXAMPLES

A company can create a subscriber account for each employee, giving them a business phone number and a web address. A main toll-free number and a web phone on the main website can be deployed to point to a voice menu designed with the call flow builder that routes calls between AI Agents and the various employees.

A remote work environment can be added to allow subscribers to meet in various video conferences, which they can access from a web or mobile app.

A voice assistant can be created that integrates with the middleware of the company, looks up information, and routes calls to appropriate destinations. This can be traditional or AI-driven, depending on the complexity of the situation. This opens the door to custom call center functionality designed in minutes.

History:

We started FreeSWITCH because we wanted to solve *our* own challenging problems.

We could not find a stable, scalable way to implement our 2004 call center as a service product. DIY solutions were a collection of random tools with constant stability and maintenance issues. Premade solutions were massively expensive and still required physical data centers anyway.

We built FreeSWITCH on our vision of how to solve those problems, made a community, and used it for insight, and it went well beyond the original goal.

FreeSWITCH became so ubiquitous that the world made many vertical products with it that generate billions of dollars a year.

SignalWire POV:

At SignalWire, we empower businesses with easy-to-use and customizable cloud communications tools, moving beyond one-size-fits-all solutions to spark innovation and growth.

Telecommunications (technology-assisted human communication) is one of the most important technologies humankind has ever developed. It has evolved from written messages on paper carried by messenger to a vast array of devices and mediums, such as telephones and computers, using audio and video.

Telecommunications have pushed businesses forward with every advancement as long ago as the telegram. The more information can be easily shared or discussed, the better it is for buyers and sellers alike. Modern telephone networks and the Internet have pushed this concept even further. Industries like entertainment, health care, education, and customer service have all benefited greatly.

Before SignalWire began, our founding team spent the better part of 2005-2015 helping to innovate the concept of IP Communications at scale with disruptive open-source technology.

Unfortunately, in recent times, there has been a massive slowdown in the evolution of this technology. Large providers often sacrifice innovation to avoid cannibalizing profits, which leads to rigidity and suffering for end users. Innovation was stifled, and this short-sightedness caused a traffic jam ahead of us.

We started SignalWire because we wanted to take our pioneering communications technology that took decades to develop and present it in a digestible format that allows customers to deploy exactly what they want, freeing them from the obstacles of time, cost, and resources.

We fundamentally believe in our dedication to removing obstacles and paving the way forward for each other as well as our customers. That mentality drives everything we do and empowers us to innovate and grow as a global community.

There is a crisis in the communications industry.

Strategies to execute the transformation to cloud communications infrastructure often present three significant pitfalls. These can lead to a reluctance to commit fully or to avoid the transformation entirely as everyone around them evolves.

The first pitfall comes from trying to deploy on-premise solutions. They are the most rigid of all the options and by far the most costly. Physical hardware becomes obsolete in a short time, causing endless refresh cycles. The hardware will likely fail, causing late-night support issues that stop businesses cold. Most of the hardware requires expensive support contracts per unit, and the businesses are at the mercy of the vendors.

The next pitfall is encountered by businesses who choose to use a series of end-to-end applications. They are forced to use multiple products, some repetitive in nature, getting phone service with one company, customer call centers with another, and video and conferencing service from another. Nothing is connected. These application providers are more focused on the status quo than moving their customers forward. This leads to a loss in productivity and a lousy customer experience for end-users.

The last common pitfall plagues businesses that try the other option, which is to build something proprietary. They face a litany of challenges they did not anticipate. There are huge decisions to make, and most evolve around going well outside their core competencies. This comes with the burdens of hosting and maintaining complex services and staffing developers with rare and elusive skill sets. In the end, many businesses scrap the project after wasting years of money and opportunity costs. The ones who managed to get something working often are too late and are plagued with maintenance and other distractions.

SignalWire is uniquely qualified to provide businesses with a path to cloud communications that reduce and eliminate distractions and obstacles while providing access to all of the features that were previously only available in expensive end-to-end solutions coupled with ever-evolving advanced features not available from anywhere else.

We see a world where all businesses can set up and own the entire chain of communications to and from their company, combining traditional phone-number-based communications with mobile app and web-app-based paradigms. We make that a reality by focusing on solving the most challenging infrastructure and feature deployment problems and presenting them to customers as point-and-click options. Integrations with SaaS tools and simple scripting unlock everything in between.

This vision benefits not only our customers but also our customers' customers. Everyone universally despises the idea of calling somewhere and dealing with ancient auto-attendants that hang up on you or send you in circles, or being asked for your account number by no less than five people while still not getting to someone who can help. Instead, powerful AI Agents can gather information and perform tasks to solve problems and narrow down the number of calls that end up on hold. Clear lines of communication from corporate websites, phone lines, and mobile applications can coalesce in one place, be easy to use, easy to extend, and always on.

SignalWire sets itself apart in the cloud communication landscape by offering highly customizable solutions, a stark contrast to the standard, rigid packages of many providers. Our innovative use of advanced technologies, like AI, and an accessible, open-source foundation enable us to meet diverse business needs more effectively. This approach not only breaks from industry norms but also addresses the evolving communication challenges faced by businesses today.

At SignalWire, we believe that by making powerful cloud communications tools and features easily programmable, configurable, and accessible to all businesses without the overhead of one-size-fits-all solutions, the result will be prosperous innovation and growth for the entire ecosystem.

The End of CPaaS: Why SignalWire's AI-Integrated Call Fabric Redefines Real-Time Communication

A Technical Blueprint for the Stack Everyone Else is Still Trying to Build

---

1. Executive Summary

The telecom and CPaaS landscape has remained stagnant for over a decade, a landscape plagued by bolt-on APIs, vendor lock-in, and fragmented media handling. As the market rushes to retrofit AI into traditional voice stacks, latency balloons, complexity surges, and developers are left struggling to integrate components that were never designed to work together.

SignalWire redefines this reality with a radical new paradigm: **Call Fabric**, a programmable, addressable network layer that treats every participant, room, and AI as a first-class, routable resource. It's a unified, programmable, and composable media infrastructure where **AI is embedded directly into the real-time communication stack**. With **sub-500ms latency**, seamless support for **voice, video, messaging, PSTN, SIP, and WebRTC**, and the powerful declarative scripting language **SWML**, SignalWire delivers the future of communications today.

Others are still raising capital to build what SignalWire has already deployed globally.

---

2. The Problem with CPaaS, Vertical AI, and "Real-Time" Infrastructure

2.1 CPaaS is Not Programmable Enough

Traditional CPaaS exposes APIs, but developers are still limited to rigid voice and messaging workflows, such as predefined IVRs or hardcoded call trees with limited conditional logic. Customization often requires external logic hosted elsewhere, leading to disjointed experiences. You can't embed intelligence into the stack, you wrap it around the stack, and the seams always show.

2.2 Voice AI Is Still a Bolt-On

Platforms like Twilio's Voice AI and Vonage AI Connect stream raw audio through WebSockets to external LLMs. This adds latency (1.5 to 3 seconds), increases costs, and fractures context management. These systems cannot maintain fluid, lifelike conversations.

2.3 Channels Are Siloed

SIP, PSTN, WebRTC, video, and messaging are typically handled by distinct services or even entirely separate vendors. Developers have to stitch together identity, routing, and AI logic across tools that were never meant to interoperate. This makes omnichannel AI nearly impossible.

2.4 Fragmentation Adds Latency and Cost

Every additional service adds hops. Each hop adds delay. Developers face escalating costs for bandwidth, compute, and coordination, while users endure awkward pauses and brittle handoffs between systems.

---

3. Introducing SignalWire's Call Fabric

Call Fabric is SignalWire's next-generation programmable communications layer. It treats every element, media stream, user, AI, room, or script, as a composable, addressable resource.

3.1 Composable Resources

- **Subscribers**: SIP endpoints, mobile apps, or authenticated user identities
- **Rooms**: Multiparty audio/video bridges
- **SWML Scripts**: JSON/YAML logic documents that describe real-time call flows
- **AI Agents**: Digital employees that interact with users across modalities
- **Queues**: Intelligent routing and dispatch primitives

3.2 Addressed Like the Web

Every resource has an address (e.g., `/support/agent_bot`) that can be referenced, dialed, and interacted with, just like web URLs. This means developers can route calls or messages between resources using declarative logic, with full AI orchestration in between.

3.3 Execute Across All Channels

SignalWire supports PSTN, SIP, WebRTC, SMS, IP messaging, and video conferencing as native, first-class channels. Developers can initiate, bridge, and route interactions across any modality in real time, enabling dynamic workflows that preserve context, memory, and state across the entire conversation lifecycle, even as the channel changes.

3.4 Real-World Composition

For example, consider an inbound PSTN call answered by an AI Agent. The agent greets the caller, identifies their intent using natural language processing, queries a business hours API, and determines whether to route the call immediately or offer voicemail. If escalation is required, it dynamically places the caller into a Room with an available representative. After the session ends, the agent sends a personalized summary and follow-up via SMS, all executed seamlessly within a single, declarative workflow.

---

4. The AI Voice Stack: Embedded, Composable, and Live

4.1 AI Embedded in the Media Plane

Unlike competitors, SignalWire's AI kernel lives **inside** the media and signaling stack. This allows real-time transcription, inference, and synthesis with **an average 500ms round-trip latency**. No WebSockets. No proxy servers. No detours.

4.2 Full Runtime with STT, TTS, and LLMs

Speech-to-text, text-to-speech, and natural language reasoning are all tightly coupled, which ensures faster response cycles, reduced failure points, and more context-aware interactions for the user. The AI understands the media context, maintains conversation state, and can trigger functions, route calls, or escalate to humans, all declaratively.

4.3 AI Agents as Resources

Agents can be invoked by address, reused in scripts, transferred into, or even embedded inside Rooms. You can treat AI like a SIP user, but with intelligence.

4.4 Serverless Tools via SWAIG

SWAIG lets developers define actions the AI can perform using only JSON. These actions can be templated serverless workflows or proxied HTTP requests with runtime variable expansion. This makes it trivial to plug AI into CRMs, knowledge bases, payment APIs, and more, with no glue code.

---

5. SWML: A Language to Program Communication

5.1 Declarative Control

SWML (SignalWire Markup Language) is a document-based, JSON/YAML-friendly language designed for real-time control of communication and AI logic.

5.2 One-Liners for Everything

Want an agent to answer and collect info? One line.

```
Unset
- ai:
    prompt:
      text: "You're a receptionist. Greet the caller and collect
their name and reason for calling."
```

5.3 Full Flow Programming

With `execute`, `goto`, `cond`, and `switch`, you can branch flows, collect input, loop, and embed reusable logic. Unlike low-code IVR builders, SWML scales to real-world, multi-step workflows.

5.4 Post-Call Intelligence

Use `post_prompt` or `post_prompt_url` to summarize, log, or escalate sessions. The AI can convert natural conversation into JSON, ready for back-end systems.

---

6. LiveKit Reality Check: They're Building What We Already Built

In April 2024, LiveKit raised $22 million to become the programmable media infrastructure for AI. While their ambitions mirror SignalWire's, their product does not.

LiveKit provides WebRTC infrastructure. It has no SIP, no PSTN, and no AI runtime. According to their own April 2024 Series B announcement, developers are responsible for stitching together LLMs, memory systems, and transcription services externally, requiring custom logic, orchestration middleware, and integration layers for even basic agent interactions.

SignalWire already delivers all of this today.

Capability	SignalWire	LiveKit
Voice Stack w/ SIP/PSTN	✅ Native	❌ None
AI Kernel in Media	✅ Yes	❌ No
Real-Time STT + TTS	✅ Embedded	❌ External Only
Declarative Call Orchestration	✅ SWML	❌ DIY Glue Code

Composable Telecom Primitives	✅ Rooms, Subs, Queues, AI Agents	❌ Media Rooms Only
Stage of Product	✅ Deployed	⚡ In Development (Series B)

---

7. Displacing the CPaaS + AI Stack

Feature	SignalWire	Twilio	Vonage	PolyAI	Dialogflow	LiveKit
AI Built Into Media Stack	✅	❌	❌	❌	❌	❌
Fully Addressable Resource Model	✅	❌	❌	❌	❌	❌
~ 500ms Round Trip Latency	✅	❌	❌	❌	❌	❌
SIP + WebRTC + Video + PSTN	✅	⚠️	✅	❌	❌	❌
Declarative AI + Call Logic	✅	❌	❌	❌	⚠️ Prompt only	❌
No Glue Code or Middleware	✅	❌	❌	❌	❌	❌

---

8. Developer Experience: One Line to Start, Infinite Depth

SignalWire was designed with the developer in mind, from the first line of SWML to full-scale applications that can power global call centers. The developer experience is built on three principles: simplicity, composability, and control.

Getting started with AI-driven communications takes a single line of SWML. Developers can spin up an AI Agent with a purpose-specific prompt, route a call, and deploy it to a phone number or SIP endpoint in moments. There's no infrastructure to manage, no API keys to juggle across providers, and no need to manually connect speech recognition to a language model.

As needs grow, so does the depth of customization. Developers can integrate retrieval-augmented generation (RAG) systems to feed the AI live data, add memory to support stateful interactions, and define tools using SWAIG to allow agents to take meaningful actions like checking order status, submitting forms, or scheduling appointments. All of this is expressed declaratively in a single SWML document, with optional external APIs or webhooks for deeper integration.

Whether routing based on user intent, escalating to human agents, sending SMS follow-ups, or transcribing calls in real time, developers remain in full control of the experience. SignalWire removes the burden of orchestration and instead delivers a model where the platform handles real-time execution, while developers focus on outcomes.

SignalWire doesn't just simplify development, it elevates it. Developers get a fully programmable, AI-native comms stack that responds in milliseconds, scales automatically, and speaks the language of modern software.

---

9. Performance Engineering at Global Scale

SignalWire's architecture is performance-first, engineered from the ground up to handle real-time audio, video, and AI processing with near-zero latency. Unlike platforms that pass media through cloud microservices or stream audio to third-party processors, SignalWire keeps everything local to its own global edge network. This minimizes round-trip delay and ensures a seamless, responsive user experience.

The platform is built on FreeSWITCH, the open-source engine trusted by Tier 1 carriers and VoIP providers worldwide, and has been battle-tested in some of the most demanding telecom environments on the planet. SignalWire extends FreeSWITCH's low-level power into a modern cloud-native fabric, allowing any developer to tap into its capabilities without needing to manage servers or SIP infrastructure.

AI processing is embedded within the media path. This design eliminates the typical latency spikes that occur when raw audio is shipped to distant LLMs or transcription services. Instead, speech-to-text, natural language processing, and synthesis all happen inside the same runtime. The result: average latencies below 500 milliseconds, even for complex interactions involving memory and external API calls.

SignalWire's infrastructure is geographically distributed, automatically routing traffic through the closest media edge location. This ensures that users across North America, Europe, and Asia experience consistent performance and quality. The system supports dynamic scaling, horizontal load distribution, and automatic failover, all essential features for enterprise-grade deployment.

Security and compliance are also core to the performance model. Audio data is encrypted end-to-end, sensitive information can be handled outside the AI's memory space, and metadata can be tokenized to protect user identity while preserving context. Whether it's for healthcare, finance, or government, SignalWire is built to meet modern compliance standards without sacrificing speed.

---

10. The Strategic Future

SignalWire's architecture isn't just solving today's telecom problems, it's laying the foundation for a new era of programmable communication. In this new model, every conversation is orchestrated dynamically in real time, with media, memory, logic, and modality all managed within a single, coherent system. As more organizations realize the limitations of retrofitting legacy systems with AI, the shift toward native, embedded intelligence will accelerate. SignalWire is already leading that transition with infrastructure built for scale, security, and real-time action.

The concept of **Call Fabric** mirrors the early internet, where every resource is addressable, composable, and built to interoperate. Just as URLs transformed how we access content, SignalWire's resource model allows communication endpoints, AI Agents, Rooms, Subscribers, to act like building blocks for new applications. This architecture positions SignalWire not just as a telecom company, but as the communications layer of the modern internet.

In the coming years, we will see AI Agents evolve from assistants to collaborators. They won't just answer questions, they'll complete tasks, escalate conversations intelligently, and learn over time. SignalWire enables this by making AI a first-class citizen of the media stack, not an external service tacked on with middleware and latency.

As digital identity becomes more portable and interaction channels multiply, SignalWire's composable infrastructure ensures continuity across modalities. A conversation that starts on the phone can shift to video, then to messaging, with context and state preserved. This fluid, multimodal approach unlocks entirely new forms of customer experience, support, and automation.

The future of communications isn't a product, it's a platform. And SignalWire's Call Fabric is already powering it.

---

11. Conclusion

The communications industry is standing at a pivotal inflection point. Legacy CPaaS vendors and next-gen media startups alike are racing to stitch AI into telecom, but they're burdened by outdated paradigms, fragmented tooling, and an architectural gap between conversation and action.

SignalWire changes that. It doesn't offer another API, chatbot, or voice add-on. Instead, it delivers a platform where AI, media, logic, and infrastructure are one and the same. Call Fabric unifies programmable communication across voice, video, and messaging, while embedding real-time AI agents into the stack itself. The result: faster development, natural conversations, and an execution model that scales.

While others define roadmaps and raise rounds to catch up, SignalWire is already there. Live in production. Deployed globally. Built by the same minds who revolutionized telecom with FreeSWITCH, now doing it again with programmable AI.

The shift is no longer coming. It's here, and it's composable, real-time, and built on SignalWire.

Appendices

A. Full AI Agent SWML Example
B. LiveKit PR & Competitive Notes
C. Benchmarks: Latency, Call Setup, Bandwidth
D. Call Fabric Resource Diagram
E. Glossary: PUC, SWAIG, SWML, etc.

SignalWire AI Gateway (SWAIG) and How it Compares to MCP

Introduction

The emergence of the Model Context Protocol (MCP) has sparked excitement across the AI and developer ecosystems. Positioned as a forward-looking standard for enabling large language models (LLMs) to securely access external tools, data, and context, MCP proposes a structured, extensible framework that emphasizes modularity, consent, and safety. It introduces concepts like resources, tools, and prompts, bound together by role-based interactions that could allow AI assistants to perform meaningful actions.

But while MCP defines an ambitious blueprint, SignalWire has already built and operationalized this vision. SWAIG, the SignalWire AI Gateway, is a mature, production-grade protocol and runtime system powering real-world AI agents across telecom channels at scale. From programmable voice workflows to real-time messaging orchestration, SWAIG is not just a concept. It is the infrastructure behind live applications.

This blog explores why SWAIG stands as the realized model of an AI Gateway, showcasing practical features, declarative simplicity, and battle-tested execution environments that MCP has yet to reach. MCP paints a compelling vision of how AI integrations might work; SWAIG demonstrates how they already do: cleanly, scalably, and securely.

The Core Philosophy

Both SWAIG and MCP share a foundational goal: to move AI beyond passive response generation and toward safe, structured interaction with the world. An AI Gateway Protocol must allow LLMs to execute actions with contextual awareness and operational control.

MCP introduces this through a layered, declarative model composed of resources (data), tools (functions), and prompts (task templates). It separates system roles-Host, Client, Server-and suggests negotiation mechanisms for capability discovery and permission granting. The framework promotes modularity and abstraction, and in theory, it supports general-purpose AI integrations.

However, in practice, MCP remains early-stage. Public specifications exist, and reference SDKs are emerging, but its deployment is largely confined to IDE assistants and localized environments. Despite claims of vendor neutrality, most implementations remain experimental or non-secure (e.g., Cursor, Sourcegraph, Windsurf). There is no clear path to secure, real-time operational deployment in production-grade applications.

SWAIG, by contrast, is fully implemented and running live across SignalWire's programmable voice and messaging infrastructure. It is not a toolkit for building an integration stack. It is the integration stack. SWAIG defines executable functions, validates arguments, manages consent and scope, and routes real-time media-all through a declarative interface embedded in SignalWire's agent markup language, SWML.

Rather than conceptual roles, SWAIG delivers operational behavior. Its functions can be serverless (templated calls to APIs) or routed to HTTP services, and they are invoked directly in conversation flows without glue code or orchestration layers. The agent understands how and when to call them, what to do with the results, and how to proceed in the interaction.

Design and Simplicity

MCP emphasizes a flexible, modular architecture. However, its design requires considerable scaffolding: multiple abstract layers (resources, tools, prompts, metadata, context), permission negotiation, and external orchestration for runtime behavior. This creates overhead and friction for developers trying to build production agents.

SWAIG simplifies this by doing less. Its primitives are compact and expressive: `function`, `parameters`, `data_map`, `output`, and `action`. These map directly to behavior and require no separate discovery mechanism or orchestration layer. One YAML or JSON block defines the interface, validation, conversational response, and downstream actions.

Example: Serverless Function with Data Map

Unset

```
- function: get_weather
  purpose: Get the current weather based on the user's location
  argument:
    type: object
    properties:
      location:
        type: string
        description: The name of the city or zip code
  data_map:
    expressions:
      - pattern: ".*"
        string: "${args.location}"
  output:
    response: "Weather data retrieved."
```

```
    action:
      - SWML:
          version: "1.0.0"
          sections:
            main:
              - send_sms:
                  to_number: "+15554441234"
                  region: "us"
                  body: "Here's the weather for ${args.location}"
                  from_number: "+15554441234"
```

Example: Hosted Function with Webhook

```
Unset
- function: get_weather
  description: To determine the current weather for a given
location.
  parameters:
    type: object
    properties:
      location:
        type: string
        description: The location to check
  fillers:
    en-US:
      - "Checking the weather for you."
  web_hook_url:
https://api_key:secret_token@your-swaig-server.com/get_weather
```

Expected server response:

```
Unset
{
  "response": "It’s currently 72°F with clear skies in Austin,
Texas.",
```

```
  "action": {
    "set_meta_data": {
      "last_checked_location": "Austin"
    }
  }
}
```

In this model:

- `response` shapes the next AI turn
- `action` defines operational behavior (e.g., say, transfer, metadata)

No external discovery. No runtime glue. All validated and run in context.

Integration in the Real World

MCP's primary integrations today are with IDE copilots and developer tools. These environments are constrained, localized, and generally asynchronous. They are valuable but limited in operational scope.

SWAIG is deployed in live telecom flows: PSTN, SIP, WebRTC, and SMS. A SWAIG agent can receive a phone call, parse speech via STT, call a serverless function, respond via TTS, and take action-all within a single call session. It runs real-time, across channels, with no intermediary orchestration.

Real-World Example: Appointment Scheduler

- Define `get_availability` as a hosted or serverless SWAIG function
- Ask the user for a time
- Use `data_map` to query the booking API
- Let user pick a time
- Use `send_sms` to confirm

No backend logic is required beyond the API call. No IVR scripting. No third-party orchestration.

SWAIG is already in use for:

- Voice bots answering questions and routing leads
- SMS agents scheduling appointments or collecting preferences

- Cross-channel assistants combining messaging, voice, and data
- Real-time agents integrated with CRMs and operational backends

---

Security and Operational Trust

MCP focuses on user-centric permission and abstraction but does not define how runtime environments should enforce execution constraints. This leaves security as an implementation detail-flexible, but inconsistent and risky in operational settings.

SWAIG functions operate within a tightly controlled execution model:

- JSON Schema defines argument validation
- Only a fixed set of actions are permitted
- All functions execute within a call/session context
- Timeout, barge-in, and state transitions are managed by the platform
- Transitions and actions are logged and auditable

This yields a deterministic, traceable behavior model where AI actions are predictable and safe.

Examples of allowed actions:

- `say` or `response`: conversational output
- `SWML`: telephony instructions (transfer, hold, record)
- `set_meta_data`, `toggle_functions`, `stop`, etc.

The result is a structured, declarative, and secure environment for AI operations-critical in regulated industries or sensitive applications.

Scalability and Deployment

MCP aspires to be language-agnostic and modular. However, it lacks a shared runtime, unified execution model, or deployment guidance. Today, it runs mostly in browser tools or developer IDEs without scale benchmarks or secure endpoint integration.

SWAIG is already deployed across SignalWire's telecom-grade infrastructure. It inherits FreeSWITCH's performance and reliability, and has been used in thousands of concurrent voice sessions.

Teams can:

- Start serverless (API-only functions)
- Deploy hosted SWAIG servers (Heroku, Fly.io, AWS)

- Mix real-time calls, SMS, and backend logic
- Reuse functions across multiple agents
- Scale incrementally and predictably

This composability allows fast iteration without compromising structure or safety.

Built-in Tool Hosting and Discovery

A unique feature of SWAIG is its ability to dynamically fetch tools from external sources via HTTP POST. Developers can:

- Host SWAIG-compatible tools on their own server
- Expose functions via authenticated POST endpoints
- Register tool URLs in SWML agents declaratively
- Enable auto-discovery with explicit permission control

This is conceptually similar to MCP's vision of tool discovery, but significantly simpler. No negotiation protocol, no capability graph. Just set the endpoint, define the schema, and let the agent invoke it.

It is declarative, scalable, and secure-without overengineering.

How MCP Could Learn from SWAIG

MCP's structure is thoughtful, but early implementations lack the constraints and integration patterns required for production readiness. It could evolve faster by adopting elements from SWAIG:

1. **Data-first execution models**
    - Instead of resource abstractions, SWAIG uses `data_map` blocks to bind inputs to live data.
2. **Declarative action flows**
    - SWAIG returns typed, structured `action` blocks that eliminate ambiguity.
3. **Lifecycle integration**
    - SWAIG functions execute within voice/messaging sessions and manage state transitions.
4. **Unified format**
    - SWAIG functions behave the same whether serverless or hosted-minimizing mental overhead.

MCP's principles are solid, but operational grounding is what transforms protocols into platforms. SWAIG illustrates that real-world design requires execution, not just abstraction.

Operational Milestones

SWAIG has been running in production since early 2023, integrated directly into SignalWire's global telecom infrastructure. It has handled Thousands of concurrent real-time voice sessions.

This operational maturity is not speculative. It reflects learnings and refinements from live deployments in regulated, customer-facing environments.

Conclusion

MCP is a promising attempt to define the future of LLM-driven tools. It is modular, extensible, and aligned with privacy-first principles. But while it proposes how AI Gateways might work, SWAIG proves how they already do work, today.

SWAIG is declarative, enforceable, composable, and live in production. It eliminates unnecessary complexity, runs directly in the SignalWire platform, and powers real-time voice and messaging AI agents.

It is not a concept. It is a framework. It closes the loop between intent and action, between specification and behavior, between architecture and execution.

For organizations looking to operationalize AI agents today, not in theory, SWAIG provides the shortest path from design to deployment. It is the blueprint for operational AI agents. And it is already live.

Examples and More Information

To explore SWAIG and SWML in practice, see the following live resources:

- SignalWire AI Platform
- AI Workshop Examples by @briankwest
- SignalWire Digital Employees Reference
- Official SWML AI Guides and Documentation

Bitter Lessons in AI and Communication

Learning from Practical Applications

Key Lesson: Systems must adapt to real-world use cases and performance requirements rather than theoretical idealizations.

SignalWire's Alignment:

- SignalWire's Call Fabric (a horizontal implementation of Programmable Unified Communications, or PUC) emphasizes flexibility and real-world scalability by breaking down complex communication systems into modular resources.
- The use of SWML (SignalWire Markup Language) allows rapid deployment of dynamic communication solutions tailored to real-world needs, such as AI-powered IVRs and seamless call routing.

Optimization of Latency and Efficiency

Key Lesson: High-performance systems require a deep understanding of the full technology stack.

SignalWire's Strength:

- Unlike many platforms that rely on third-party CPaaS for media handling, SignalWire integrates the media stack and CPaaS capabilities, reducing latency by eliminating intermediate layers.
- Innovations like real-time transcription, vector-based AI integration, and highly optimized orchestration ensure low latency, making SignalWire a leader in efficient communication systems.

Avoiding Narrow Vertical Solutions

Key Lesson: General-purpose solutions often outperform narrowly focused vertical products.

SignalWire's Differentiation:

- By offering developer-friendly APIs and support for diverse applications like video conferencing, telehealth, and real-time customer support, SignalWire provides a general-purpose, horizontally scalable platform.
- Competing solutions often lock users into specific use cases (e.g., Twilio’s less-flexible voice AI), whereas SignalWire enables full programmability and customization across industries.

Focus on Developer Enablement

Key Lesson: Empower developers with tools that abstract complexity while offering depth.

SignalWire's Implementation:

- The SWAIG (SignalWire AI Gateway) integrates powerful AI tools like OpenAI's GPT models into a framework that enables developers to build advanced digital employees with minimal effort.
- SignalWire offers low-code/no-code options, making communication technology accessible without compromising on power.

Integration Over Isolation

Key Lesson: Successful systems interact seamlessly with existing tools and platforms.

SignalWire's Value:

- SignalWire's platform is built around open standards like SIP, WebRTC, and REST APIs, ensuring easy integration with legacy systems and modern cloud solutions.
- Unlike competitors, SignalWire's PUC approach avoids vendor lock-in, supporting hybrid and multi-cloud architectures.

Adaptability and Future-Proofing

Key Lesson: Systems must be able to evolve alongside technological and market shifts.

SignalWire's Approach:

- The modular design of SignalWire's Call Fabric enables continuous updates and expansions without major overhauls.
- Features like dynamic context switching and real-time resource allocation allow for ongoing adaptation to user demands and evolving AI capabilities.

Cost and Accessibility

Key Lesson: Democratizing access to technology is as important as the technology itself.

SignalWire's Achievement:

- By significantly lowering entry barriers through straightforward pricing and out-of-the-box interoperability, SignalWire reduces both time-to-value and operational costs.
- Unlike traditional CPaaS solutions that incur high integration overheads, SignalWire empowers businesses of all sizes to leverage cutting-edge communication tools.

Conclusion

SignalWire exemplifies many principles from the "bitter lessons" of AI by focusing on modularity, developer empowerment, scalability, and adaptability. Its comprehensive PUC platform transcends the limitations of traditional CPaaS and UCaaS, providing an unparalleled blend of real-time performance, customization, and seamless integration.

For more information, visit SignalWire to explore how it aligns with these transformative principles.

SignalWire is a telco-grade, AI Voice Platform for building the next decade of communication systems. SignalWire helps developers and businesses succeed by simplifying Communication technology into a future-proof solution that enables them to quickly build scalable, innovative solutions that empower their customers to thrive.

Three Critical Inflection Points in Technology

Inflection Point 1

1989 World Wide Web

Browsers emerge, HTML becomes the universal interface

Impact: Read-only internet - the birth of online content + websites become the first layer of digital presence

Inflection Point 2

2006 The Cloud

Launch of EC2 and S3, API driven compute

Impact: The cloud-native revolution begins

Inflection Point 3

2022 AI

Foundation models and real-time inference become accessible

API-first platforms shift from commands to conversations

Impact: The internet becomes intelligent

Telecom locked inside on-prem infrastructure

2006: FreeSWITCH is born – giving voice to the programmable internet.

Impact: Becomes the number one open-source telephony platform that revolutionized the VoIP industry. Powers 95% cloud communications industries:

CPaaS
UCaaS
CCaaS
} Powered by FreeSWITCH

2018: SignalWire is formed to take FreeSWITCH into the cloud.

Modular, protocol-agnostic telecom core built for developers

Bridges SIP, PSTN, WebRTC, and more under one engine

The foundation for Programmable Unified Communications (PUC)

Impact: Real-time communications become programmable, composable, and embeddable

SignalWire's PUC Platform is to Communication what EC2 was to Compute and what LLMs are to Understanding

Impact: PUC is the missing layer that lets Humans and AI interact via comms channels in real time.

The Biggest Inflection Point is Here – NOW

The convergence of several events are forcing the first re-engineering of global voice platforms in 15+ years

- Genesys, Metaswitch, and legacy on-prem are being phased out
- AI is transforming contact centers and workflows
- Developer adoption is the new distribution
- The developer platform that enables this shift will own the next decade

SignalWire was architected for this shift from its inception.
We have the infrastructure, AI orchestration, and protocol stack **live today – at scale**

The Legacy Stack is Breaking

AI is now central to customer interaction, but the legacy communications stacks were not built for it

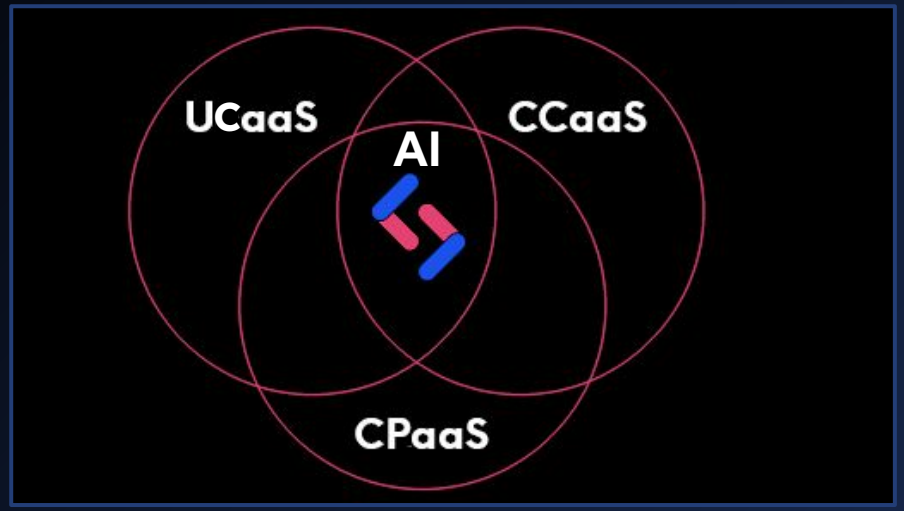
Today's systems are:

- **Latency-prone** and stitched together non-defensible AI use-cases with multiple API
- Designed for **rigid** use cases of **low quality audio** and rudimentary call routing to humans only
- Inflexible across all protocols, both legacy and modern (SIP, PSTN, WebRTC, and Mobile)
- Impossible for developers to extend to modern ideas and product requirements

Current Solutions are Limited

Company	Limitation
Twilio (CPaaS)	**Gen 1 CPaaST** – Not built for real-time AI workflows. Requires multiple third-party integrations for advanced routing, NLP, etc. Cannot innovate on the already ancient voice stack.
LiveKit (AI)	**New in Town** – No programmable telecom stack. Needs years to accelerate
Replicant (AI IVR)	**Closed IVR System** – Not a developer platform or infrastructure layer, making the solution rigid
Genesys/MetaSwitch (CCaaS)	**Legacy CCaaS** – Complex and expensive solution, not programmable. EOL on-prem causing a mass exodus to the cloud.

SignalWire : What Makes Us Unique?



Creating something entirely new:
Programmable Unified Communications

Join a Video Call with AI

- ➔ **Composable Infrastructure Approach**
  - ◆ Translates between mobile/browser world and traditional telecom. Global scale with intelligent routing network
- ➔ **Deep Programmability**
  - ◆ Built by the creators of FreeSWITCH; developers gain direct, low-latency control over media and call flows
- ➔ **Native AI Integration**
  - ◆ Real-time transcription, translation, and programmable Digital Employees all baked into the platform (no third-party “bolt-ons”)
- ➔ **Flexible, Usage-Based Model**
  - ◆ Scales affordably without enterprise-level license fees. Ideal for everything from startups to large-scale call centers
- ➔ **Developer-Centric Ecosystem**
  - ◆ Integrates easily with CRMs, ERPs, LLMs and other AI services, and more

SignalWire : Use Case One

SignalWire's Programmable Unified Communications (PUC) approach abstracts telecom primitives into composable **Resources** (like AI Agents, Subscribers, Scripts, Rooms, Gateways), which can be mixed and matched to create dynamic, scalable communication systems. Applications can be up and running in weeks rather than years.

Secure, Compliant Healthcare Appointments (HIPAA)

Goal: Let patients schedule, confirm, or cancel appointments via voice

Resources Used:

- AI Agent: For appointment interaction
- Room: For escalated video calls with doctors
- Subscriber: For back-office agents or nurse line
- SWML Script: For call routing and compliance logic
- SWAIG: For integration with EMRs and scheduling systems

Flow:

1. Patient calls a number tied to a SWML script
2. An AI Agent confirms their identity and retrieves availability via SWAIG
3. They confirm or change appointments
4. If escalation is needed, patient is transferred to a Room or Subscriber

✅ **Sensitive information is kept secret for PCI and HIPAA compliance**

SignalWire : Use Case Two

Virtual Receptionist for Distributed Teams

Goal: Build a smart receptionist that can handle distributed, hybrid staff across the globe.

Resources Used:

- AI Agent: Natural conversation + name/intent capture
- Subscribers: Remote employees with SIP/WebRTC endpoints
- Queue: If staff is unavailable
- Script: Core logic
- SendSMS: To notify team members of missed calls

Flow:

1. Caller is greeted by the AI agent
2. AI asks “Who are you trying to reach?” or “How can I help you?”
3. Based on user intent, routes to the correct Subscriber (e.g., marketing, sales).
4. If person is away, the call goes to Voicemail, then sends an SMS/email.

SignalWire: Platform Overview – SignalWire is the programmable fabric that connects real-time AI, SIP, PSTN, WebRTC, Mobile, SMS, and Video.

Real-Time Infrastructure

A highly available, distributed, and reliable cloud-based communication infrastructure that breaks down key application features into development building blocks.

AI Kernel Embedded in the Media Stack

Voice, transcription, memory, and LLM reasoning run inside the media layer. No detours

Model-Aware Agent Infrastructure

SignalWire's production-ready equivalent of MCP (Model Context Protocol) is called SWAIG. It securely defines agent goals, memory, tools, and behavior in a stable format that has been in use for over a year.

Omnichannel by Design

Unified platform across PSTN, SIP, WebRTC, SMS, Video, and AI natively built into the stack.

Subscribers

Managed user identities that enable programming of PBX and Call Center features, including roles and call routing, to build integrated customer experiences.

Declarative Call Logic

Developers build conversational applications using structured prompts and JSON logic, rather than wiring APIs and sockets by hand.

SignalWire

GLOBAL NETWORK

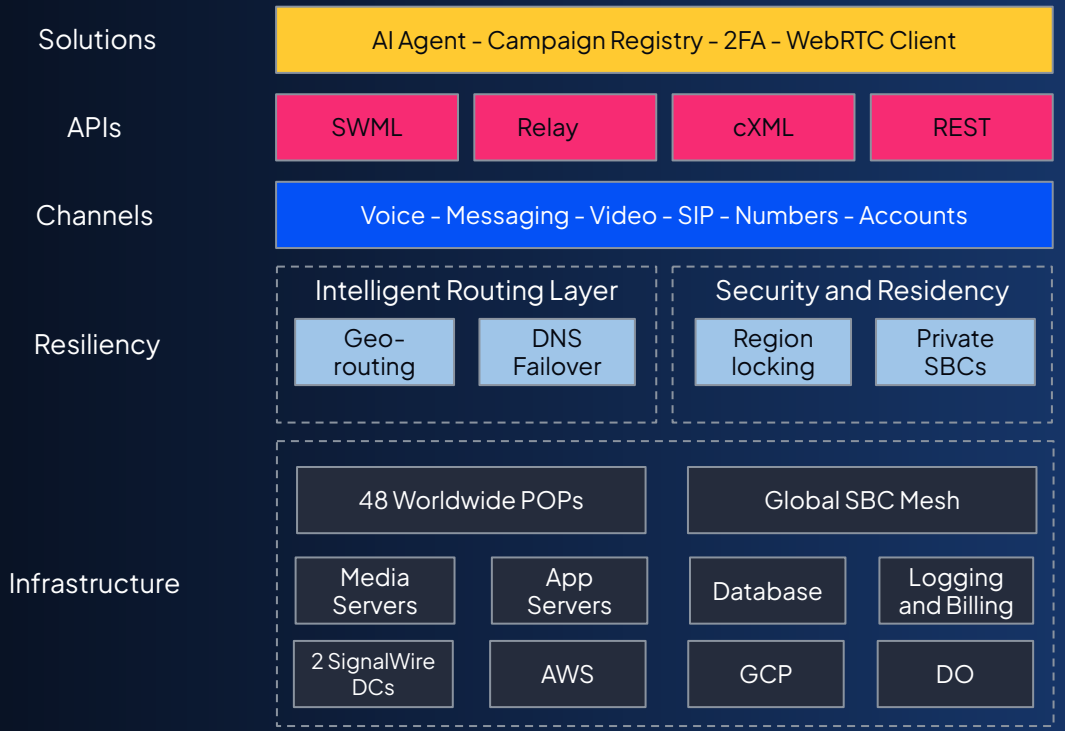
Covering over 5 billion people within 50ms & globally within 100ms

- Nodes placed worldwide to ensure high availability, low latency, and scalable communication infrastructure.
- Tailored solutions for specific business needs such as data sovereignty requirements, serving clients like Sprinklr, Deutsche Telekom, and large call centers.
- Cloud agnostic architecture allows deployments to any data center or network.

Continuous analysis detects disruptions and uncovers path optimizations on a per-endpoint basis.

Secured with SOC II, HIPAA,PCI compliance, and built-in access control.

SignalWire: A Complete set of Services



SignalWire: Go-to-Market

Product Led Growth partnering with Sales Led Growth

- Customer focused platform designed for Product Led Growth (PLG), including self-service and POC acceleration
- Strategic Partnerships with Enterprise and telcos to integrate with SignalWire's powerful Programmable Unified Communications (PUC) platform, future-proofing their offering
- Customer Support and Customer Success increases consumption and retention. 2024 net retention 130%
- Revenue Streams: SignalWire generates revenue from various sources including consumption-based fees, monthly subscriptions, dedicated infrastructure, and professional services for platform integration. 96% revenue is recurring

Focus on industries with heavy customer engagement needs, companies with technical ability, and development resources.

SignalWire: What Makes Us Unique?

- **Everything is a composable resource**: Use APIs to create. APIs to control. SDKs and phone numbers to place and receive calls.
- **Real-time programmable logic**: Using SWML, you can orchestrate everything at runtime.
- **Cross-channel support**: Works with PSTN, SIP, WebRTC, and messaging - with consistent logic.
- **Global scale + edge routing**: Deploy anywhere, route intelligently.
- **AI-native**: AI Agents aren't bolted-on - they're part of the core. They can see, be seen, hear, speak and control calls based on simple instructions and integration points using SWAIG (our product ready MCP equivalent)

Join a Video Call with AI