

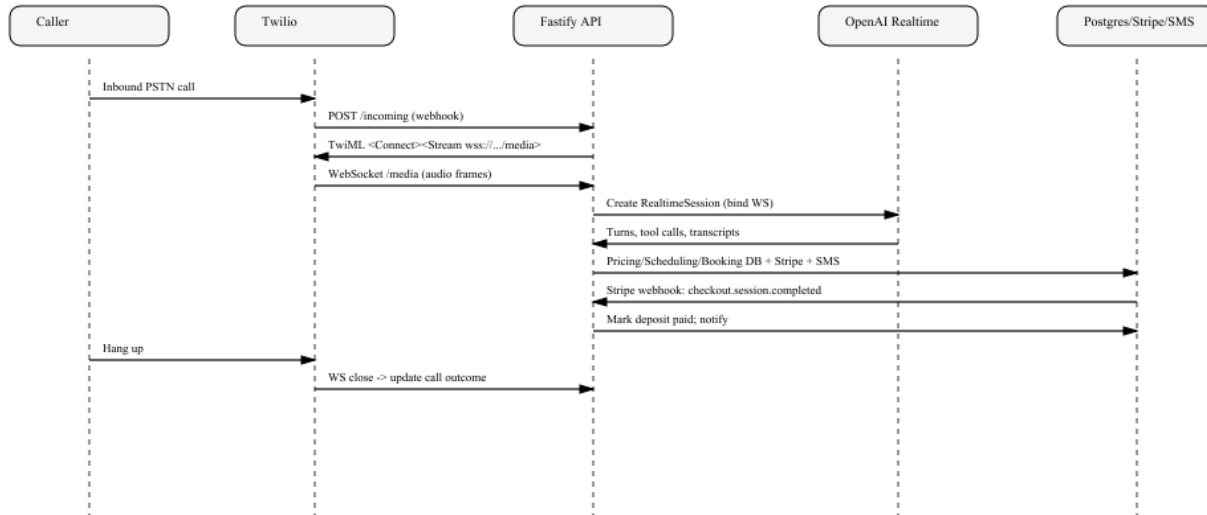
Agentic Trades Voice Agent — End-to-End Operation & Simulated Call

Date: 2025-08-29

1) Bird's-eye overview

- Customer calls your number (Twilio answers, hits /incoming).
- We reply with TwiML to bridge audio via to /media.
- Twilio opens a WebSocket to /media and streams audio.
- We create an OpenAI Realtime session bound to that WS; the agent follows business rules and uses tools.
- Tools: pricing, scheduling, booking (Stripe Checkout deposit + SMS), escalation to human.
- DB writes: calls, agent_turns, customers, jobs, payments (on webhook).
- Metrics: custom CloudWatch metrics for quotes, bookings, latencies, escalations.

Sequence overview



2) Detailed call flow, step by step

- **Inbound call** → **/incoming**: optional Twilio signature validation; respond with TwiML streaming to /media.

- **Twilio connects WebSocket** → **/media**: insert a row into *calls*; bind the OpenAI Realtime session; start logging agent turns.
- **Agent greeting & disclosure**: short, compliant, friendly.
- **Qualification**: single-question cadence; detect emergency keywords.
- **Pricing**: PricingEngine applies base rate, after-hours multiplier, complexity factor.
- **Availability**: SchedulingEngine proposes 2-hour windows excluding conflicts (or immediate emergency window).
- **Booking**: upsert customer; insert job; create Stripe Checkout deposit; send SMS link and confirmation.
- **Payment webhook**: on checkout.session.completed, mark deposit paid and update job/payment tables.
- **Closeout**: update calls.outcome and leave agent_turns for QA and analytics.

3) Realistic simulation (dialogue)

Agent: "Thanks for calling CoolCare HVAC. I'm the AI scheduling assistant. How can I help with your HVAC?"
 Customer: "My AC stopped cooling and the outdoor unit is rattling."
 Agent: "Got it. Is anyone smelling gas, smoke, or carbon monoxide? Or is it just the cooling issue and noise?"
 Customer: "Just the cooling and noise. No smells."
 Agent: "Understood. Can I grab your name and the best callback number?"
 Customer: "Sarah Miller, 555-123-0000."
 Agent: "Thanks, Sarah. What's the service address?"
 Customer: "123 Oak Street, Springfield, Ohio."
 Agent: "Thanks. Since it's after hours, there's an emergency fee. Let me get you a quick estimate, then I'll schedule you."
 Agent: "Typically this runs \$300-\$800 depending on what we find. Tonight, the service call would be about \$206."
 Customer: "Yes, please."
 Agent: "I can do tonight 9-11pm, or tomorrow 8-10am. Which works?"
 Customer: "Tonight 9-11pm."
 Agent: "All set for today 9-11pm. You'll get a text with a deposit link – it's \$206. Anything I should note?"
 Customer: "Gate code 1234. One friendly dog."
 Agent: "Thanks! I've added those notes. I just texted your link; pay when you can. See you this evening!"

4) Tool calls and responses (examples)

Pricing — get_pricing_quote (request)

```
{
  "service_type": "repair",
  "description": "AC not cooling; outdoor unit rattling.",
  "is_after_hours": true
}
```

Pricing — get_pricing_quote (response)

```
{
  "service_call": 165,
  "estimated_repair": 825,
  "range": "$300-$800",
  "total_estimate": 825,
  "emergency_fee": 248,
  "message": "Here's your pricing estimate. Would you like to schedule an appointment?"
}
```

Availability — *check_availability* (request)

```
{
  "service_type": "repair",
  "is_emergency": true,
  "preferred_date": null
}
```

Availability — *check_availability* (response)

```
{
  "available_slots": [
    { "datetime": "2025-08-28T21:00:00-04:00", "display": "Thu, Aug 28, 9:00 PM" },
    { "datetime": "2025-08-29T08:00:00-04:00", "display": "Fri, Aug 29, 8:00 AM" },
    { "datetime": "2025-08-29T10:00:00-04:00", "display": "Fri, Aug 29, 10:00 AM" }
  ],
  "message": "For emergency service, I can get you on today's schedule."
}
```

Booking — *book_appointment* (request)

```
{
  "customer_name": "Sarah Miller",
  "phone": "+15551230000",
  "email": null,
  "address": "123 Oak St, Springfield, OH",
  "service_type": "repair",
  "description": "AC not cooling; outdoor unit rattling. Gate code 1234; dog on site.",
  "scheduled_time": "2025-08-28T21:00:00-04:00",
  "estimated_cost": 825,
  "is_emergency": true
}
```

Stripe Checkout response (summary)

```
{
  "checkout_url": "https://checkout.stripe.com/c/session/...",
  "deposit_amount": 206
}
```

5) What gets stored

- **calls:** one row per call (sid, timestamps, business_hours, outcome).
- **agent_turns:** every assistant/user/tool turn with latency and tool metadata.
- **customers:** upserted from booking flow.
- **jobs:** one per booking, with window, priority/urgency, estimated costs, deposit flags.
- **payments:** deposit/charge rows (on webhook) with processor refs.
- **metrics/views:** aggregates for dashboards and KPIs.

6) Variants & safeguards

- Emergency keywords trigger 911 guidance; still allow scheduling after safety advice.
- After-hours fees/multiplier and messaging are automatic.

- Deposit not paid → leave scheduled with deposit_paid=false; your ops policy can auto-cancel/notify.
- Escalation path is a single tool call; can bridge to /handoff with an on-call number.
- Compliance: no card collection by voice; mandatory recording disclosure; address spell-back prompts.

7) TL;DR sequence

- Twilio → /incoming (Twiml) → /media
- /media → start Realtime session; create calls row
- Agent → pricing → availability → booking (+ Stripe Checkout)
- DB: customers upsert, jobs insert; SMS deposit link
- Stripe webhook → mark deposit paid
- Call ends → update calls outcome; metrics already emitted