Product Requirements Document - agentic-ivr

Author: Edu **Date:** February 14, 2026

Executive Summary

This product is a **conversational IVR service** that replaces traditional DTMF phone menus with natural-language voice interaction powered by Azure AI. It acts as a **real-time audio bridge** between Azure Communication Services (ACS) Call Automation and the Azure AI Voice Live API — translating between ACS's binary audio framing and Voice Live's OpenAI Realtime-compatible JSON events over two concurrent WebSocket connections per call.

Vision: A caller dials the company number, hears an AI greeting within 2 seconds, and has a natural conversation — no menus, no hold queues, no "press 1 for..."

Target users: Telecom customers calling for account inquiries, billing, and service requests (pt-BR). The system prompt — authored by domain specialists, not developers — is the only per-client customization.

Differentiator: Production-ready in 5 days (1 developer + Copilot), security-hardened from Day 1, fully testable without Azure credentials, and designed for retirement if ACS adds native Voice Live integration.

Tech stack: Java 21 + Spring Boot 4.x + Spring MVC with Virtual Threads (Tomcat), JSR 356 WebSocket server, `java.net.http.WebSocket` client, `DefaultAzureCredential` everywhere, OpenTelemetry for distributed tracing.

Scale: 50 concurrent calls per instance, 99.9% availability, ≥99% call completion rate.

Reading Guide

This PRD was built iteratively through 10 discovery and specification steps. Sections appear in creation order rather than the canonical PRD reading order. Use this map for logical navigation:

Canonical Section	Location
Executive Summary	*You are here*
Success Criteria	Success Criteria
Product Scope (MVP + Growth)	Product Scope
User Journeys (5 journeys, 4 personas)	User Journeys
Functional Requirements (50 FRs, 10 capabilities)	Functional Requirements
Non-Functional Requirements (47 NFRs, 6 categories)	Non-Functional Requirements
Architecture & Constraints	Architectural Principles
Security (STRIDE + Attack Surface + 18 Requirements)	STRIDE Threat Model, Security Requirements
Observability (23 events, 16 metrics, 5 spans)	Observability Requirements
Configuration (61 parameters, zero magic numbers)	Configuration Catalog
Call State Machine (12 states, 33 transitions)	Call State Machine
Sprint Plan (5-day, 30 items)	Scoping & MVP Strategy

Resilience & Fault Tolerance Requirements (Non-Functional)

Derived from Chaos Monkey failure analysis during discovery.

FM-1: ACS WebSocket Drop Mid-Call (Severity: Critical)

- **REQ-R1.1:** Both WebSocket connections (ACS ↔ Service, Service ↔ Voice Live) MUST have a **linked lifecycle** — closure of either side MUST trigger graceful teardown of the other within 3 seconds.
- **REQ-R1.2:** Implement **heartbeat/keepalive** on both WebSocket connections; detect staleness within 2–3 seconds.
- **REQ-R1.3:** If ACS reconnects within a 5-second window, attempt session resumption with the existing Voice Live session.
- **REQ-R1.4:** If reconnection fails or times out, play a "sorry, we lost the connection" prompt via ACS Call Automation SDK and hang up gracefully.

FM-2: Voice Live API Unavailable (Severity: Critical)

- **REQ-R2.1:** Implement a **circuit breaker** pattern on Voice Live WebSocket connections. After N consecutive failures (configurable, default 3), trip the circuit.
- **REQ-R2.2:** When circuit is open, new calls MUST receive a **fallback audio prompt** ("We're experiencing difficulties, please try again later") played via ACS SDK, followed by graceful hangup.
- **REQ-R2.3:** Voice Live WebSocket connection MUST be established within **3 seconds** or fail fast.
- **REQ-R2.4:** A **background health probe** MUST periodically test Voice Live connectivity and preemptively trip the circuit breaker if the probe fails.
- **REQ-R2.5:** Track Voice Live connection success rate; alert when < 95%.

FM-3: Pod Termination During Active Calls (Severity: High)

- **REQ-R3.1:** On SIGTERM, **stop accepting new calls immediately** (readiness probe → unhealthy).
- **REQ-R3.2:** For active calls, send a polite prompt via Voice Live ("One moment please"), then gracefully close both WebSocket connections per call.
- **REQ-R3.3:** Set Kubernetes `terminationGracePeriodSeconds` to **120+ seconds** to allow connection draining.
- **REQ-R3.4:** Rolling deployments MUST use `maxUnavailable: 0` to ensure capacity is never reduced during rollout.
- **REQ-R3.5:** Enforce a **maximum call duration** (configurable, default 10 minutes) to bound drain time.

FM-4: Memory Pressure / OOMKill (Severity: High)

- **REQ-R4.1:** Implement **admission control** — track active call count per instance and reject new calls (return 503) when approaching the configured limit (default: 80% of max capacity).
- **REQ-R4.2:** Autoscaling MUST use a **custom metric** (active WebSocket connection count or memory usage), not CPU alone.
- **REQ-R4.3:** Container memory limit set conservatively (1 GB per instance, ~100 concurrent calls max). Prefer horizontal scaling over vertical.
- **REQ-R4.4:** When nearing capacity, mark readiness probe as **unhealthy** to stop new call routing.

FM-5: Event Grid Delivery Delay (Severity: Medium)

- **REQ-R5.1:** Monitor **answer latency** (time between call start and answer). Alert if p95 exceeds 5 seconds.
- **REQ-R5.2:** Configure **dead-letter destination** on the Event Grid subscription to capture failed deliveries.
- **REQ-R5.3:** Set ACS `ringTimeoutInSeconds` to **45 seconds** (extended from default ~30s) to provide headroom for Event Grid delays.
- **REQ-R5.4:** *(Stretch/V2)* Consider a redundant Event Grid subscription to a Service Bus queue as fallback.

Resilience Summary

ID	Failure Mode	Severity	Key Hardening
FM-1	ACS WebSocket drop mid-call	**Critical**	Linked lifecycle, heartbeat, reconnection window
FM-2	Voice Live API unavailable	**Critical**	Circuit breaker, fallback prompts, health probe
FM-3	Pod termination during calls	**High**	Graceful drain, SIGTERM handler, rolling deploy
FM-4	Memory OOM	**High**	Admission control, memory-based autoscaling
FM-5	Event Grid delivery delay	**Medium**	Answer SLA monitoring, dead-letter, extended ring timeout

---

Architectural Principles & Constraints (First Principles)

Derived from First Principles Analysis during discovery.

AP-1: Protocol Translator, Not Audio Processor

The service is a **protocol translator and auth bridge** between ACS and Voice Live. It MUST NOT buffer, inspect, or transform audio data — only forward bytes immediately between the two WebSocket connections. This is the core architectural identity of the service.

AP-2: Two-WebSocket Model Is Inherent

The two concurrent WebSocket connections per call (ACS ↔ Service, Service ↔ Voice Live) are an **inherent architectural constraint**, not a design choice. ACS media streaming uses ACS-specific binary framing (`StreamingData.parse()`). Voice Live uses OpenAI Realtime-compatible JSON events. The service exists precisely to translate between these two protocols. Plan capacity around **2N connections for N concurrent calls**.

AP-3: Thin API Gateway Integration for MVP

The MVP includes a **thin tool call framework** that routes Voice Live `tool_call` events to backend APIs via Azure API Management (APIM). The bridge service makes outbound HTTPS calls to APIM using `java.net.http.HttpClient` (JDK built-in, same module as the Voice Live WebSocket client) running on virtual threads — blocking `HttpClient.send()` is effectively non-blocking under Project Loom.

The scope is deliberately narrow: 3 predefined tool calls (GET user data, send invoice to registered email, send invoice to user-provided email). The bridge handles tool call dispatch and result forwarding only — **no business logic** lives in the bridge. Backend API implementation behind APIM is out of scope.

Auth model: `DefaultAzureCredential` acquires an OAuth2 bearer token scoped to the APIM audience (`C-57b`). APIM validates the token via `validate-jwt` inbound policy. No client secrets — managed identity in prod, `az login` locally.

Key constraint: APIM infrastructure is pre-provisioned (same as ACS, Foundry, etc.). The bridge only needs the gateway base URL (`C-57`) and auth scope (`C-57b`).

AP-4: Container-Portable Deployment

The application MUST be deployable on **any OCI-compatible container platform** (AKS, Azure Container Apps, App Service for Containers) without code changes. Deployment-platform-specific configuration (Kubernetes manifests, ACA YAML, Bicep/Terraform) is maintained **separately** from application code. No platform-specific APIs (e.g., Kubernetes client libraries) in application code.

AP-5: Virtual Threads + Spring MVC over WebFlux

The project uses **Java 21 virtual threads with Spring MVC** (servlet stack + Tomcat) instead of Spring WebFlux (reactive stack). This decision was made through First Principles Analysis and Developer Deep Dive during discovery:

Rationale:

- **ACS Java SDK compatibility:** The official ACS audio streaming quickstart uses **Jakarta WebSocket (JSR 356)** annotations (`@ServerEndpoint`, `@OnMessage`, `Session`). This works natively on Tomcat (servlet container) but requires non-trivial adaptation for WebFlux's `WebSocketHandler` + `Flux<WebSocketMessage>` model. Using Spring MVC means SDK quickstart code works **as-is** with zero adaptation.
- **Backpressure is unnecessary:** Real-time audio streams at a fixed, predictable rate (~48 KB/sec per direction for PCM 24K mono). This is bounded by physics (real-time speech), not by variable throughput. Reactive backpressure solves a problem this system doesn't have.
- **Virtual threads eliminate the scalability concern:** Java 21 virtual threads handle hundreds of concurrent connections without thread exhaustion — the primary reason WebFlux was historically chosen for WebSocket workloads.
- **Simpler programming model:** Imperative code is easier to write, debug, and maintain. No `Mono`/`Flux` operators, no reactive stack traces, no scheduler complexity.
- **Tool call HTTP calls on virtual threads:** Tool call handlers make blocking `java.net.http.HttpClient.send()` calls to APIM. Virtual threads handle these naturally — no `WebClient`, no reactive chains, no blocking workarounds. The same JDK module (`java.net.http`) provides both the WebSocket client (Voice Live) and the HTTP client (APIM gateway).

Stack:

- Web server: **Tomcat** with `spring.threads.virtual.enabled=true`
- ACS WebSocket server: **JSR 356** `@ServerEndpoint` (ACS SDK quickstart pattern)
- Voice Live WebSocket client: `java.net.http.WebSocket` (JDK built-in) or equivalent
- REST controllers: **Spring MVC** `@RestController` on virtual threads
- ACS SDK client: `CallAutomationClient` (synchronous) — natural fit for virtual threads

> **TODO (Architect):** Create a formal ADR documenting the decision to use Spring MVC + Virtual Threads over Spring WebFlux, with explicit trade-offs and conditions under which this decision should be revisited (e.g., if variable-rate audio streaming or non-audio streaming workloads are added).

AP-6: Designed for Retirement

The bridge service should remain **thin and stateless**, designed to be gracefully retired if Azure Communication Services adds native Voice Live API integration in the future. No business logic should accumulate in the bridge layer. Business logic belongs in Voice Live's system prompt configuration or in backend services (V2+).

First Principles Summary

ID	Principle	Key Implication
AP-1	Protocol translator, not audio processor	Zero audio buffering/inspection — forward bytes immediately
AP-2	Two-WebSocket model is inherent	Capacity planning: 2N connections for N calls
AP-3	Thin API gateway integration for MVP	3 tool calls routed via APIM; `java.net.http.HttpClient` on virtual threads; `DefaultAzureCredential` for gateway auth
AP-4	Container-portable	No K8s-specific code; deploy on AKS, ACA, or App Service
AP-5	Virtual Threads + Spring MVC over WebFlux	JSR 356 SDK compatibility, fixed-rate audio eliminates backpressure need, simpler model

ID	Principle	Key Implication
AP-6	Designed for retirement	Keep bridge thin; no business logic accumulation

Test Infrastructure & Developer Experience

Derived from joint Developer + QA deep dive during discovery.

Rationale

Both ACS and Voice Live are cloud-only services with no local emulators. To enable offline development, fast CI pipelines, and deterministic resilience testing, the project ships with **in-process test simulators** that replicate the WebSocket behavior of both external systems.

Test Simulators (P0 — Required for Any Testing)

FakeAcsClient

A WebSocket client that pretends to be ACS, connecting to our `@ServerEndpoint`:

- Connects to the service's WebSocket endpoint (just like ACS would)
- Sends `AudioMetadata` packet (JSON, configuring the stream)
- Sends `AudioData` packets from pre-recorded PCM fixtures
- Receives outbound audio (AI responses forwarded by the bridge)
- **Fault injection:** disconnect mid-stream, send malformed data, send silence-only packets
- **Assertions:** verify what audio was received back, verify timing constraints

FakeVoiceLiveServer

A WebSocket server that pretends to be Voice Live, accepting our outbound connection:

- Accepts outbound WSS connection from the service
- Receives `session.update` → validates config (including tool definitions), sends back `session.created`
- Receives `input_audio_buffer.append` → collects audio data
- Sends `response.audio.delta` events (pre-recorded AI audio)
- Sends `input_audio_buffer.speech_started` (barge-in simulation)
- Sends `tool_call` events (tool call simulation — e.g., `get_user_data`, `send_invoice`)
- Receives `tool_call_output` → validates tool call ID + response payload
- **Fault injection:** configurable delay, disconnect, refuse connection (503 simulation), malformed events, slow responses
- **Assertions:** verify what audio/events were received, verify session configuration, verify tool call outputs

WireMock Gateway Stub

A WireMock instance simulating the APIM gateway for tool call HTTP requests:

- Responds to `POST /api/v1/users/{id}` (GET user data tool call)
- Responds to `POST /api/v1/invoices/send` (send invoice tool calls)
- **Fault injection:** configurable response delay (> C-58 for timeout testing), 4xx/5xx HTTP errors, connection refused
- **Assertions:** verify request includes `Authorization: Bearer <token>` header, verify `correlationId` propagation
- Runs on `http://localhost:8082` in `test` profile (matches C-57 test override)

Test Fixture Library (P0)

Pre-recorded JSON fixtures for deterministic, repeatable tests:

```
src/test/resources/fixtures/
├── acs/
│   ├── audio-metadata.json            # AudioMetadata packet (stream config)
│   ├── audio-hello-pcm24k.json        # 3-second "hello" as AudioData packets
│   ├── audio-silence.json             # Silent AudioData packet
│   ├── audio-malformed.json           # Invalid packet for error testing
│   └── dtmf-digit-3.json             # DTMF event
├── voicelive/
│   ├── session-created.json           # Response to session.update
│   ├── response-audio-greeting.json   # AI greeting (response.audio.delta)
│   ├── speech-started.json            # Barge-in event
│   ├── response-done.json             # response.done event
│   ├── tool-call-get-user-data.json   # tool_call event: GET user data
```

```
│   ├── tool-call-send-invoice.json    # tool_call event: send invoice
│   └── tool-call-output-success.json  # Expected tool_call_output response
├── gateway/
│   ├── user-data-response.json        # APIM GET user data response
│   └── send-invoice-response.json     # APIM send invoice response
└── scenarios/
    ├── happy-path.yaml                # Full call: ACS sends → VL responds → ACS receives
    ├── barge-in.yaml                  # User interrupts AI mid-response
    ├── silence-timeout.yaml           # User says nothing, VAD fires
    ├── tool-call-success.yaml         # AI triggers tool call → gateway responds → result returned
    ├── tool-call-timeout.yaml         # AI triggers tool call → gateway timeout → error returned, call continues
    └── websocket-drop.yaml            # ACS disconnects mid-stream
```

Scenario files define scripted interaction sequences with expected outcomes. The integration test runner loads a scenario and orchestrates both simulators accordingly.

(P2) Fixture recording tool: debug mode that captures real ACS/Voice Live packets during E2E tests and saves them as JSON fixtures for the library.

Spring Profiles (P0)

Profile	ACS Source	Voice Live Source	Auth	Use Case
`test`	FakeAcsClient (in-process)	FakeVoiceLiveServer (in-process)	None	CI/CD, unit & integration tests
`local`	Real ACS (via devtunnel)	Real Voice Live	`az login` / `DefaultAzureCredential`	Local dev with real services
`prod`	Real ACS (direct)	Real Voice Live	Managed identity	Production deployment

Configuration files:

- `application.yml` — shared config (audio format, timeouts, defaults)
- `application-test.yml` — localhost URLs for simulators, no auth
- `application-local.yml` — devtunnel URLs, local ACS credentials
- `application-prod.yml` — production config (Key Vault refs, managed identity)

Developer Setup Guide (P0 — Required Deliverable)

A step-by-step guide from `git clone` to first call, covering:

1. Prerequisites: JDK 21, Maven, Azure CLI, devtunnel CLI
2. Azure resource provisioning: ACS resource (dev tier) + AI Foundry resource with Voice Live model
3. `az login` for `DefaultAzureCredential` locally
4. Start devtunnel to expose webhook + WebSocket endpoints
5. Configure Event Grid subscription to point to devtunnel URL
6. `./mvnw spring-boot:run -Dspring.profiles.active=local`
7. Place a test call to the ACS phone number

CI/CD Test Strategy (P1)

PR Pipeline (cloud-free, fast):

1. `./mvnw test` — Unit tests (mocks, no network)
2. `./mvnw test -Pintegration` — Integration tests with FakeAcsClient + FakeVoiceLiveServer (no cloud credentials needed)
3. Static analysis (SpotBugs, Checkstyle, etc.)
4. `./mvnw org.owasp:dependency-check-maven:check` — Dependency vulnerability scan (fail on CVSS ≥ 7) (SEC-REQ-16)
5. Container image scan (Trivy or Grype) — fail on Critical/High CVEs (SEC-REQ-16)

Nightly Pipeline (cloud resources required): 4. Deploy to staging environment (AKS or ACA) 5. E2E smoke test (real phone call via ACS test number) 6. Chaos tests: kill pods (FM-3), throttle Voice Live (FM-2), simulate Event Grid delay (FM-5)

Caller Experience (UX) Requirements

Derived from Caller Journey Mapping during discovery.

These requirements define what the **caller hears** at every phase of the call. Resilience engineering (FM-x, REQ-Rx) ensures the system survives failures; UX requirements ensure the **caller** survives them gracefully.

Phase 1: Ring → Answer (Caller waits for pickup)

- **UX-REQ-1:** After answering the call, if no audio is sent to the caller within **2 seconds**, play a brief comfort tone or "Please wait" prompt via ACS Play action (not Voice Live) as a fallback. This bridges the gap between call answer and Voice Live session readiness.

Phase 2: Connection Setup (Answer → First AI Audio)

- **UX-REQ-2:** The Voice Live `session.update` MUST include `turn_detection` configuration so that Voice Live proactively sends a greeting without waiting for user audio. The system prompt must instruct the AI to greet the caller immediately.
- **UX-REQ-3:** If Voice Live does not produce a `response.audio.delta` within **5 seconds** of `session.created`, the service MUST play a fallback greeting via ACS Play action and log a `WARN` event. This prevents the "silent deadlock" scenario.
- **UX-REQ-4:** Total time from **call answer to first AI audio reaching the caller** MUST be < **3 seconds** (p95). This is the single most critical UX metric. Track as `ivr.time_to_first_audio`.

Phase 3: Conversation (Natural dialogue)

- **UX-REQ-5:** The barge-in implementation MUST handle **rapid successive interruptions** gracefully. If the caller interrupts, then the AI starts responding, and the caller interrupts again within 500ms — the bridge must not enter an inconsistent state. Design barge-in as an **idempotent operation** (sending `StopAudio` when no audio is playing is harmless).
- **UX-REQ-6:** Voice Live VAD configuration (semantic VAD, threshold, silence duration, noise suppression, echo cancellation) MUST be **externalized as configuration** (not hardcoded). Different caller environments (call center, mobile, landline) may need tuning.
- **UX-REQ-7:** Add a metric `ivr.ai.response.latency` — time from end of caller speech (VAD silence detection) to first AI audio byte reaching ACS. Target: **< 1 second** p95. This is the "conversational responsiveness" metric.

Phase 4: Barge-In (Caller interrupts AI)

- **UX-REQ-8:** The bridge MUST prioritize `speech_started` events over all other event processing. When a `speech_started` event arrives, `StopAudio` MUST be sent to ACS **before** any queued audio packets. This is the single highest-priority event in the system.

Barge-in timing budget:

```
T+0.0s  Caller starts talking over AI
T+0.1s  Voice Live detects speech → sends speech_started
T+0.1s  Bridge receives speech_started → sends StopAudio to ACS
T+0.2s  ACS stops playing AI audio — caller now hears silence
T+0.5s  Caller finishes speaking — VAD detects silence
T+1.0s  New AI audio starts streaming to caller
```

Phase 5: Call End (Conversation concludes → hangup)

- **UX-REQ-9:** When max call duration is approaching (e.g., 30 seconds before limit), the service MUST inject a notification to Voice Live (via a system-level message or context update) prompting the AI to wrap up the conversation naturally. The caller should NEVER experience an abrupt mid-sentence cutoff.
- **UX-REQ-10:** When Voice Live disconnects unexpectedly during a call (linked lifecycle FM-1 scenario), the service MUST play a brief apology prompt via ACS Play action before hanging up. The caller should never hear dead silence followed by a disconnection tone.

Phase 6: Error Recovery Caller Experience

- **UX-REQ-11:** When admission control rejects a call (REQ-R4.1), the service SHOULD answer the call briefly, play a "All agents are busy, please try again shortly" prompt via ACS Play, then hangup — rather than letting the call ring unanswered to timeout.

Caller Experience Summary

Phase	UX Requirements	Critical Metric
Ring → Answer	UX-REQ-1 (comfort tone)	Answer latency
Setup → First Audio	UX-REQ-2 (proactive greeting), UX-REQ-3 (fallback), UX-REQ-4 (< 3s target)	`ivr.time_to_first_audio`
Conversation	UX-REQ-5 (idempotent barge-in), UX-REQ-6 (externalized VAD), UX-REQ-7 (< 1s response)	`ivr.ai.response.latency`
Barge-In	UX-REQ-8 (StopAudio priority)	`ivr.bargein.latency`
Call End	UX-REQ-9 (graceful max duration), UX-REQ-10 (apology on disconnect)	—
Error Recovery	UX-REQ-11 (busy message vs. unanswered ring)	—

Acceptance Criteria

Derived from QA analysis and Caller Journey Mapping during discovery.

ID	Criteria	Test Method	Priority
AC-1	Call answered within 3 seconds of Event Grid delivery	Integration test (timing) + E2E monitoring	P0
AC-2	Audio round-trip forwarded correctly (byte-for-byte ACS → Voice Live and back)	Integration test (fixture comparison)	P0
AC-3	Barge-in: `StopAudio` sent to ACS within 100ms of `speech_started` event	Integration test (timing)	P0
AC-4	Linked lifecycle: both WebSockets closed within 3s of either side disconnecting	Integration test (FM-1 scenario)	P0
AC-5	Circuit breaker trips after 3 consecutive Voice Live connection failures	Integration test (FM-2 scenario)	P1
AC-6	Graceful shutdown drains active calls before pod terminates	Integration test (SIGTERM signal)	P1
AC-7	Admission control rejects new calls at 80% capacity	Integration test (concurrent connections)	P1
AC-8	50 concurrent calls sustained for 5 minutes without errors	Load test (nightly, simulators or staging)	P1
AC-9	Time from call answer to first AI audio < 3 seconds (p95)	Integration test (timing) + E2E monitoring	P0
AC-10	Barge-in is idempotent: rapid successive interruptions do not cause state corruption	Integration test (rapid barge-in scenario)	P0
AC-11	Fallback prompt plays if Voice Live produces no audio within 5s of session creation	Integration test (FakeVoiceLiveServer delayed response)	P0
AC-12	Admission-rejected calls hear "busy" prompt instead of unanswered ringing	Integration test (capacity limit + call attempt)	P1
AC-13	Unexpected Voice Live disconnect plays apology prompt before hangup	Integration test (FM-1 + FakeVoiceLiveServer disconnect)	P1
AC-14	Max call duration approached: AI wraps up naturally (no abrupt cutoff)	Integration test (timer + FakeVoiceLiveServer)	P2
AC-15	WebSocket connection without valid JWT → rejected at handshake (HTTP 401, never upgrades)	Integration test (FakeAcsClient without JWT)	P0
AC-16	Event Grid event with `eventTime` older than 5 min → rejected	Integration test (stale event replay)	P0
AC-17	No audio bytes appear in any log output at any log level (DEBUG, TRACE included)	Static analysis + integration test (log capture)	P0
AC-18	No full phone number appears in log output — only last 4 digits visible	Integration test (log capture + regex assertion)	P0
AC-19	`/actuator/env` on application port returns 404	Integration test (HTTP GET)	P0
AC-20	HTTP 500 error response contains no stack trace, class names, or Spring-specific details	Integration test (trigger error + assert response body)	P0
AC-21	OWASP dependency-check passes with no Critical/High CVEs (CVSS ≥ 7)	CI pipeline verification	P1
AC-22	Production config contains no `connection-string` property — only `endpoint` + `DefaultAzureCredential`	Config validation + deployment test	P1
AC-23	Tool call round-trip: `tool_call` event → APIM gateway HTTP → `tool_call_output` returned to Voice Live within 5s (p95)	Integration test (WireMock gateway stub, timing assertion)	P0
AC-24	Tool call timeout: gateway does not respond within C-58 → error `tool_call_output` returned to Voice Live, call continues (no termination)	Integration test (WireMock delayed response > C-58)	P0

Observability Requirements

Derived from Stakeholder Round Table — SRE/Ops perspective.

Observability is layered into three tiers: structured logging (P0), metrics (P1), and distributed tracing (P2). Every log entry and span MUST include a `correlationId` (the ACS call ID) for end-to-end troubleshooting.

Structured Logging (P0)

All log events use structured JSON format via SLF4J + Logback. Every entry includes `correlationId`, `timestamp`, `level`, and `component`.

#	Log Event	Level	Component	Key Fields
1	Incoming call received (Event Grid)	INFO	webhook	`correlationId`, `callerNumber`, `acsResourceId`
2	Call answered	INFO	webhook	`correlationId`, `answerLatencyMs`
3	ACS WebSocket connected	INFO	websocket-server	`correlationId`, `remoteAddr`
4	ACS WebSocket disconnected	INFO	websocket-server	`correlationId`, `closeCode`, `closeReason`, `durationMs`
5	Voice Live WebSocket connected	INFO	websocket-client	`correlationId`, `endpoint`, `connectLatencyMs`
6	Voice Live WebSocket disconnected	INFO	websocket-client	`correlationId`, `closeCode`, `closeReason`, `durationMs`
7	Voice Live session created	INFO	websocket-client	`correlationId`, `sessionId`, `model`
8	Audio forwarded (ACS → Voice Live)	DEBUG	bridge	`correlationId`, `packetSize`, `sequenceNum`
9	Audio forwarded (Voice Live → ACS)	DEBUG	bridge	`correlationId`, `packetSize`, `eventType`
10	Barge-in detected	INFO	bridge	`correlationId`, `latencyMs` (time to send StopAudio)
11	Circuit breaker state change	WARN	resilience	`previousState`, `newState`, `failureCount`
12	Admission control: call rejected	WARN	resilience	`correlationId`, `activeCallCount`, `maxCapacity`
13	Graceful shutdown initiated	INFO	lifecycle	`activeCallCount`, `drainTimeoutMs`
14	Call completed	INFO	bridge	`correlationId`, `totalDurationMs`, `audioPacketsForwarded`, `bargeInCount`
15	Error (catch-all)	ERROR	*	`correlationId`, `errorType`, `errorMessage`, `stackTrace`
16	WebSocket JWT validation failed	WARN	security	`remoteAddr`, `reason` (expired/invalid/missing), `path`
17	Event Grid token validation failed	WARN	security	`remoteAddr`, `reason`
18	Event Grid replay rejected (stale event)	WARN	security	`eventTime`, `ageSeconds`, `maxAgeSeconds`
19	ACS callback JWT validation failed	WARN	security	`remoteAddr`, `callId`, `reason`
20	WebSocket rate limit exceeded	WARN	security	`remoteAddr`, `connectionCount`, `windowSeconds`
21	Tool call dispatched	INFO	tool-call	`correlationId`, `toolName`, `gatewayUrl`
22	Tool call succeeded	INFO	tool-call	`correlationId`, `toolName`, `latencyMs`, `httpStatus`
23	Tool call failed	WARN	tool-call	`correlationId`, `toolName`, `latencyMs`, `httpStatus`, `errorType`

Metrics (P1)

Exposed via **Micrometer** to Prometheus (or Azure Monitor via OpenTelemetry Collector). All metrics prefixed with `ivr.`.

#	Metric Name	Type	Description
1	`ivr.calls.active`	Gauge	Current number of active calls (WebSocket pairs)
2	`ivr.calls.total`	Counter	Total calls answered since startup
3	`ivr.calls.rejected`	Counter	Calls rejected by admission control
4	`ivr.call.duration`	Timer	Call duration distribution (p50, p95, p99)

#	Metric Name	Type	Description
5	`ivr.answer.latency`	Timer	Time from Event Grid delivery to call answered
6	`ivr.voicelive.connect.latency`	Timer	Time to establish Voice Live WebSocket
7	`ivr.voicelive.circuit.state`	Gauge	Circuit breaker state (0=closed, 1=half-open, 2=open)
8	`ivr.audio.packets.forwarded`	Counter	Audio packets forwarded (tagged by direction: `acs_to_vl`, `vl_to_acs`)
9	`ivr.bargein.latency`	Timer	Time from `speech_started` to `StopAudio` sent
10	`ivr.time_to_first_audio`	Timer	Time from call answer to first AI audio byte reaching ACS (target: < 3s p95)
11	`ivr.ai.response.latency`	Timer	Time from end of caller speech (VAD silence) to first AI audio byte (target: < 1s p95)
12	`ivr.bargein.effectiveness`	Gauge	% of barge-ins where StopAudio arrives < 100ms (target: > 95%)
13	`ivr.security.auth.rejected`	Counter	Rejected inbound connections/requests (tagged by reason: `jwt_invalid`, `jwt_expired`, `jwt_missing`, `replay`, `rate_limit`)
14	`ivr.security.websocket.rate.limited`	Counter	WebSocket connection attempts rejected by per-IP rate limiter
15	`ivr.tool_call.latency`	Timer	Tool call round-trip time: `tool_call` received → `tool_call_output` sent (p50, p95, p99)
16	`ivr.tool_call.errors`	Counter	Tool call failures (tagged by `reason`: `timeout`, `http_4xx`, `http_5xx`, `connection_error`)

Distributed Tracing (P2)

Spans emitted via **OpenTelemetry** SDK. Enable correlation from Event Grid → webhook → ACS WebSocket → Voice Live WebSocket.

#	Span Name	Parent	Key Attributes
1	`ivr.call`	(root)	`correlationId`, `callerNumber`, `callDurationMs`
2	`ivr.call.answer`	`ivr.call`	`answerLatencyMs`
3	`ivr.call.voicelive.connect`	`ivr.call`	`endpoint`, `connectLatencyMs`, `sessionId`
4	`ivr.call.audio.bridge`	`ivr.call`	`packetsForwarded`, `bargeInCount`
5	`ivr.call.tool_call`	`ivr.call`	`correlationId`, `toolName`, `latencyMs`, `httpStatus`, `success`

Priority Tiers

All requirements organized by implementation priority.

P0 — 5-Day MVP (Launch Blocker)

These are required for the system to function correctly and safely in production. No release without all P0 items passing. **Scope: 1 developer, Copilot-assisted, 5-day sprint.**

Area	Requirements
Call Flow	End-to-end: Event Grid → Answer → ACS WebSocket → Voice Live → audio round-trip → barge-in → hangup
Audio Bridge	Bidirectional audio forwarding (AP-1), linked lifecycle (REQ-R1.1), PCM 24K mono passthrough
Caller Experience	Comfort tone fallback (UX-REQ-1), proactive AI greeting (UX-REQ-2), silent deadlock prevention (UX-REQ-3), time-to-first-audio < 3s (UX-REQ-4), idempotent barge-in (UX-REQ-5), StopAudio priority (UX-REQ-8), admission-rejected callers hear busy prompt (UX-REQ-11), apology prompt on unexpected disconnect (UX-REQ-10)
Security	WebSocket JWT validation (SEC-REQ-1), Event Grid Entra ID validation (SEC-REQ-2), ACS callback JWT validation (SEC-REQ-3), non-guessable callback URLs (SEC-REQ-4), zero audio storage (SEC-REQ-5), actuator lockdown (SEC-REQ-9), error response hardening (SEC-REQ-10), ACS managed identity — no connection strings (SEC-REQ-11), Event Grid replay protection (SEC-REQ-14), WebSocket per-IP rate limiting (SEC-REQ-15), OWASP dependency-check in CI (SEC-REQ-16), wss:// only, `DefaultAzureCredential` everywhere, zero hardcoded secrets
Resilience	Admission control (REQ-R4.1, R4.4), graceful shutdown / drain (REQ-R3.1–R3.2)
Observability	All 23 log events with `correlationId` + structured JSON logging, all 16 Micrometer metrics (auto-config Day 1, custom metrics incremental), health/readiness probes, 5 OTel spans

Area	Requirements
Config	Externalized config via Spring profiles (`test` / `local` / `prod`), system prompt loaded from external file (C-12)
Test Infrastructure	FakeAcsClient + FakeVoiceLiveServer + fixture library + WireMock (gateway stub) + `test` profile
CI/CD	PR pipeline: `./mvnw test` (cloud-free) + OWASP dependency-check (fail CVSS ≥ 7)
Deployment	Container image, health/readiness probes, production deploy on Day 5
Acceptance Criteria	AC-1 through AC-4, AC-9 (time-to-first-audio), AC-10 (idempotent barge-in), AC-11 (fallback prompt), AC-12 (busy prompt), AC-13 (apology prompt), AC-15 (WebSocket JWT), AC-16 (replay protection), AC-17 (no audio in logs), AC-19 (actuator lockdown), AC-20 (error hardening), AC-21 (OWASP), AC-22 (no connection strings), AC-23 (tool call round-trip), AC-24 (tool call timeout)
Dev Experience	Developer setup guide, `./mvnw test` works with zero cloud credentials
Tool Call Framework	3 tool calls (GET user data, send invoice registered email, send invoice provided email) via APIM gateway; `java.net.http.HttpClient` on virtual threads; `DefaultAzureCredential` → APIM `validate-jwt`; tool definitions file (C-60); WireMock in tests

P1 — Growth (Post-MVP, Incremental)

Enhancements delivered after the 5-day MVP is in production. Prioritized by risk reduction and operational maturity.

Area	Requirements
Security	PII masking in logs (SEC-REQ-6), Voice Live managed identity + key disable (SEC-REQ-12), abuse monitoring opt-out (SEC-REQ-13), system prompt guardrails (SEC-REQ-17)
Security Compliance	Data residency — Brazil South (SEC-REQ-7), LGPD data processing record (SEC-REQ-8)
Resilience	Circuit breaker (REQ-R2.1–R2.4), heartbeat/keepalive (REQ-R1.2)
Caller Experience	Externalized VAD config (UX-REQ-6)
Config	Key Vault integration for secret management
CI/CD	Container image scanning (Trivy/Grype) + nightly E2E pipeline
Deployment	Kubernetes manifests or ACA config, auto-scaling prep
Acceptance Criteria	AC-5 through AC-8, AC-18 (PII masking)

P2 — Scale & Observability Polish (Post-Launch)

Enhancements for operational maturity and future growth.

Area	Requirements
Resilience	Reconnection window (REQ-R1.3–R1.4), health probes for Voice Live (REQ-R2.4 advanced), dead-letter queue (REQ-R5.2), extended ring timeout (REQ-R5.3), redundant Event Grid (REQ-R5.4)
Caller Experience	Graceful max-duration wrap-up (UX-REQ-9), AI response latency metric (UX-REQ-7)
Tracing	OpenTelemetry distributed tracing (4 spans, see Observability)
Scaling	Memory-based autoscaling (REQ-R4.2–R4.3), rolling deploy with `maxUnavailable: 0` (REQ-R3.4), max call duration (REQ-R3.5)
Acceptance Criteria	AC-14 (graceful max-duration)
Test Tooling	Fixture recorder (capture real packets → JSON), chaos test automation

Load & Scalability Targets

Capacity planning baselines per deployment tier.

Tier	Concurrent Calls	Instances	WebSocket Connections	Memory (per instance)	Notes
MVP	50	1	100 (2 per call) + APIM HTTP pool	1 GB	WS connections + HttpClient connection pool for tool calls (bounded by C-59)
Growth	200	2–3	400–600	1 GB each	First production rollout
Scale	1,000+	10+	2,000+	1 GB each	Full production, requires autoscaling

Key sizing assumptions:

- Each call = 2 WebSocket connections + ~100 KB memory (buffers, session state) + shared APIM HTTP connection pool
- Audio rate: ~48 KB/sec per direction (PCM 24K, 16-bit mono) → ~96 KB/sec total per call
- Network bandwidth per instance at 50 calls: ~4.8 MB/sec (well within container limits)
- Tool call HTTP requests are short-lived (< 5s) and bounded by C-59 max concurrent per call
- Virtual threads eliminate thread-count bottleneck — JVM can handle 10K+ virtual threads easily
- Primary scaling constraint is **WebSocket connection count per instance** and **memory for connection state**, not CPU

Definition of Done

Applies to every story/task in this project.

A task is considered **Done** when ALL of the following are satisfied:

1. **Code complete** — Implementation matches the task requirements and follows project conventions (AP-1 through AP-6)
2. **Unit tests pass** — All new code has unit tests; `./mvnw test` passes with zero failures
3. **Integration tests pass** — New behavior has integration tests using FakeAcsClient / FakeVoiceLiveServer / WireMock gateway stub; `./mvnw test -Pintegration` passes
4. **No regressions** — Full test suite passes; no previously-passing tests are broken
5. **Structured logging** — All new code paths emit structured log events with `correlationId` (per Observability section)
6. **Documentation updated** — Relevant docs (README, developer guide, API docs) are updated if behavior changes
7. **Code reviewed** — At minimum, the implementing developer has self-reviewed the diff for completeness and correctness
8. **Security check** — No hardcoded secrets, no PII in logs, no audio bytes logged, all new endpoints have auth validation, OWASP dependency-check passes

---

Configuration Catalog

Derived from Configuration Catalog elicitation during discovery.

Every configurable parameter in the system. Zero magic numbers in code — all values externalized via Spring configuration properties with validation.

Webhook Controller

ID	Property	Type	Default	Per-Env	Validation	Ref
C-01	`ivr.acs.callback-base-url`	String	—	**Required**	Valid URL	—
C-02	`ivr.acs.endpoint`	String	—	**Required**	Valid URL	SEC-REQ-11
C-03	`ivr.acs.ring-timeout-seconds`	int	`45`	Optional	15–120	REQ-R5.3

ACS WebSocket Server (JSR 356)

ID	Property	Type	Default	Per-Env	Validation	Ref
C-04	`ivr.websocket.server.path`	String	`/ws`	Rarely	Starts with `/`	—
C-05	`ivr.websocket.server.idle-timeout-ms`	long	`5000`	Optional	1000–30000	REQ-R1.2
C-06	`ivr.websocket.server.max-message-size`	int	`65536`	Optional	1024–1048576	—

Voice Live WebSocket Client

ID	Property	Type	Default	Per-Env	Validation	Ref
C-07	`ivr.voicelive.endpoint`	String	—	**Required**	Valid WSS URL	—
C-08	`ivr.voicelive.api-version`	String	`2025-05-01-preview`	Optional	Non-blank	—

ID	Property	Type	Default	Per-Env	Validation	Ref
C-09	`ivr.voicelive.model`	String	—	**Required**	Non-blank	—
C-10	`ivr.voicelive.connect-timeout-ms`	long	`3000`	Optional	500–10000	REQ-R2.3
C-11	`ivr.voicelive.idle-timeout-ms`	long	`5000`	Optional	1000–30000	REQ-R1.2

Voice Live Session Configuration

ID	Property	Type	Default	Per-Env	Validation	Ref
C-12	`ivr.voicelive.system-prompt`	String	—	**Required**	Non-blank	—
C-13	`ivr.voicelive.voice`	String	`en-US-Aria:DragonHDLatestNeural`	Optional	Valid voice ID	—
C-14	`ivr.voicelive.vad.type`	String	`semantic`	Optional	`semantic` or `server`	UX-REQ-6
C-15	`ivr.voicelive.vad.threshold`	double	`0.3`	Optional	0.0–1.0	UX-REQ-6
C-16	`ivr.voicelive.vad.silence-duration-ms`	int	`200`	Optional	50–2000	UX-REQ-6
C-17	`ivr.voicelive.noise-suppression`	boolean	`true`	Optional	—	UX-REQ-6
C-18	`ivr.voicelive.echo-cancellation`	boolean	`true`	Optional	—	UX-REQ-6

Audio Bridge

ID	Property	Type	Default	Per-Env	Validation	Ref
C-19	`ivr.audio.format`	String	`pcm_24k_mono`	Immutable	Fixed	AP-1
C-20	`ivr.audio.sample-rate`	int	`24000`	Immutable	Fixed	AP-1
C-21	`ivr.audio.silence-packet-threshold`	int	`10`	Optional	1–100	—

Resilience

ID	Property	Type	Default	Per-Env	Validation	Ref
C-22	`ivr.resilience.circuit-breaker.failure-threshold`	int	`3`	Optional	1–20	REQ-R2.1
C-23	`ivr.resilience.circuit-breaker.half-open-delay-ms`	long	`30000`	Optional	5000–300000	REQ-R2.1
C-24	`ivr.resilience.circuit-breaker.success-threshold`	int	`1`	Optional	1–5	REQ-R2.1
C-25	`ivr.resilience.health-probe.interval-ms`	long	`10000`	Optional	5000–60000	REQ-R2.4
C-26	`ivr.resilience.admission.max-calls`	int	`50`	**Per env**	1–1000	REQ-R4.1
C-27	`ivr.resilience.admission.rejection-threshold-pct`	int	`80`	Optional	50–100	REQ-R4.1
C-28	`ivr.resilience.reconnect-window-ms`	long	`5000`	Optional	0–30000	REQ-R1.3

Call Lifecycle

ID	Property	Type	Default	Per-Env	Validation	Ref
C-29	`ivr.call.max-duration-seconds`	int	`600`	Optional	60–3600	REQ-R3.5
C-30	`ivr.call.max-duration-warning-seconds`	int	`30`	Optional	10–120	UX-REQ-9
C-31	`ivr.call.linked-lifecycle-timeout-ms`	long	`3000`	Optional	1000–10000	REQ-R1.1
C-32	`ivr.call.time-to-first-audio-timeout-ms`	long	`5000`	Optional	2000–15000	UX-REQ-3
C-33	`ivr.call.comfort-tone-delay-ms`	long	`2000`	Optional	500–5000	UX-REQ-1

Graceful Shutdown

ID	Property	Type	Default	Per-Env	Validation	Ref
C-34	ivr.shutdown.drain-timeout-seconds	int	90	Optional	30–300	REQ-R3.1

Fallback Prompts

ID	Property	Type	Default	Per-Env	Validation	Ref
C-35	ivr.prompts.comfort-tone-file	String	classpath:prompts/comfort-tone.wav	Optional	Valid resource	UX-REQ-1
C-36	ivr.prompts.fallback-greeting-file	String	classpath:prompts/fallback-greeting.wav	Optional	Valid resource	UX-REQ-3
C-37	ivr.prompts.error-apology-file	String	classpath:prompts/error-apology.wav	Optional	Valid resource	UX-REQ-10
C-38	ivr.prompts.busy-file	String	classpath:prompts/busy.wav	Optional	Valid resource	UX-REQ-11
C-39	ivr.prompts.service-unavailable-file	String	classpath:prompts/service-unavailable.wav	Optional	Valid resource	REQ-R2.2

Security

ID	Property	Type	Default	Per-Env	Validation	Ref
C-40	ivr.security.jwt.enabled	boolean	true	Optional	—	SEC-REQ-1
C-41	ivr.security.jwt.issuer	String	—	Required when enabled	Valid URL	SEC-REQ-1
C-42	ivr.security.jwt.audience	String	—	Required when enabled	Non-blank	SEC-REQ-1
C-43	ivr.security.event-grid.validation-enabled	boolean	true	Optional	—	SEC-REQ-2
C-44	ivr.security.jwt.acs-oidc-metadata-url	String	ACS OIDC well-known URL	Required when enabled	Valid URL	SEC-REQ-1
C-45	ivr.security.event-grid.max-event-age-seconds	int	300	Optional	60–600	SEC-REQ-14
C-46	ivr.security.websocket.rate-limit-per-ip	int	10	Optional	1–100	SEC-REQ-15
C-47	ivr.security.websocket.rate-limit-window-seconds	int	60	Optional	10–300	SEC-REQ-15
C-48	management.server.port	int	8081	Optional	1024–65535	SEC-REQ-9
C-49	ivr.security.pii.phone-mask-visible-digits	int	4	Optional	0–4	SEC-REQ-6
C-50	ivr.security.callback-token-length	int	32	Optional	16–64	SEC-REQ-4

Profile Override Matrix

Property	test	local	prod
C-01 callback-base-url	http://localhost:8080	https://<devtunnel>.devtunnels.ms	https://ivr.example.com
C-02 acs.endpoint	http://localhost:8080	https://<acs>.communication.azure.com	https://<acs>.communication.azure.com
C-07 voicelive.endpoint	ws://localhost:9090	wss://<foundry>.../realtime	wss://<foundry>.../realtime
C-09 voicelive.model	test-model	Dev deployment	Prod deployment
C-12 system-prompt	Short test prompt	Full prompt (dev)	Full prompt (prod)

Property	test	local	prod
C-26 admission.max-calls	5	10	50–100
C-40 jwt.enabled	false	false	true
C-45 event-grid.max-event-age-seconds	300	300	300
C-46 websocket.rate-limit-per-ip	100	100	10
C-48 management.server.port	8080 (same)	8081	8081
C-57 tool-call.gateway-base-url	http://localhost:8082 (WireMock)	https://<apim>.azure-api.net	https://<apim>.azure-api.net
C-57b tool-call.gateway-auth-scope	(not used — WireMock)	api://<apim-app-id>/.default	api://<apim-app-id>/.default

API Backend Lifecycle (from Pre-mortem & First Principles — Step 7)

ID	Property	Type	Default	Per-Env	Validation	Ref	Priority
C-51	ivr.voicelive.session-init-timeout-ms	long	5000	Optional	500–15000	Pre-mortem #1	P0
C-52	ivr.call.orphan-reaper-interval-ms	long	60000	test: 1000	1000–300000	Pre-mortem #2	P0
C-53	ivr.call.allowed-numbers-pattern	String	.* (all)	Optional	Valid regex	Red Team #1	**P1**
C-54	ivr.ws.idle-timeout-ms	long	10000	test: 2000	1000–60000	First Principles + Red Team	P0
C-55	ivr.voicelive.stall-timeout-ms	long	5000	Optional	1000–30000	FR46, NFR-R13	P0
C-56	ivr.callback.dedup-ttl-seconds	int	60	test: 5	10–600	FR5, NFR-R6	P0

P1 configs are wired in code but **not enforced** in MVP. Default values ensure they are inert (.* matches all callers). This avoids test coverage confusion — P1 configs have explicit priority tags.

Tool Call Framework

ID	Property	Type	Default	Per-Env	Validation	Ref	Priority
C-57	ivr.tool-call.gateway-base-url	String	—	**Required**	Valid HTTPS URL	AP-3, FR48	P0
C-57b	ivr.tool-call.gateway-auth-scope	String	—	**Required**	Non-blank (Entra ID audience URI)	SEC-REQ-18, FR48	P0
C-58	ivr.tool-call.response-timeout-ms	long	5000	Optional	1000–30000	FR50, AC-23	P0
C-59	ivr.tool-call.max-concurrent	int	3	Optional	1–10	FR50	P0
C-60	ivr.tool-call.definitions-file	String	classpath:tool-definitions.json	Optional	Valid resource path, valid JSON	FR51, FR13	P0

Total: 61 configurable parameters across 12 components — zero magic numbers in code.

Call State Machine

Derived from State Machine Specification during discovery.

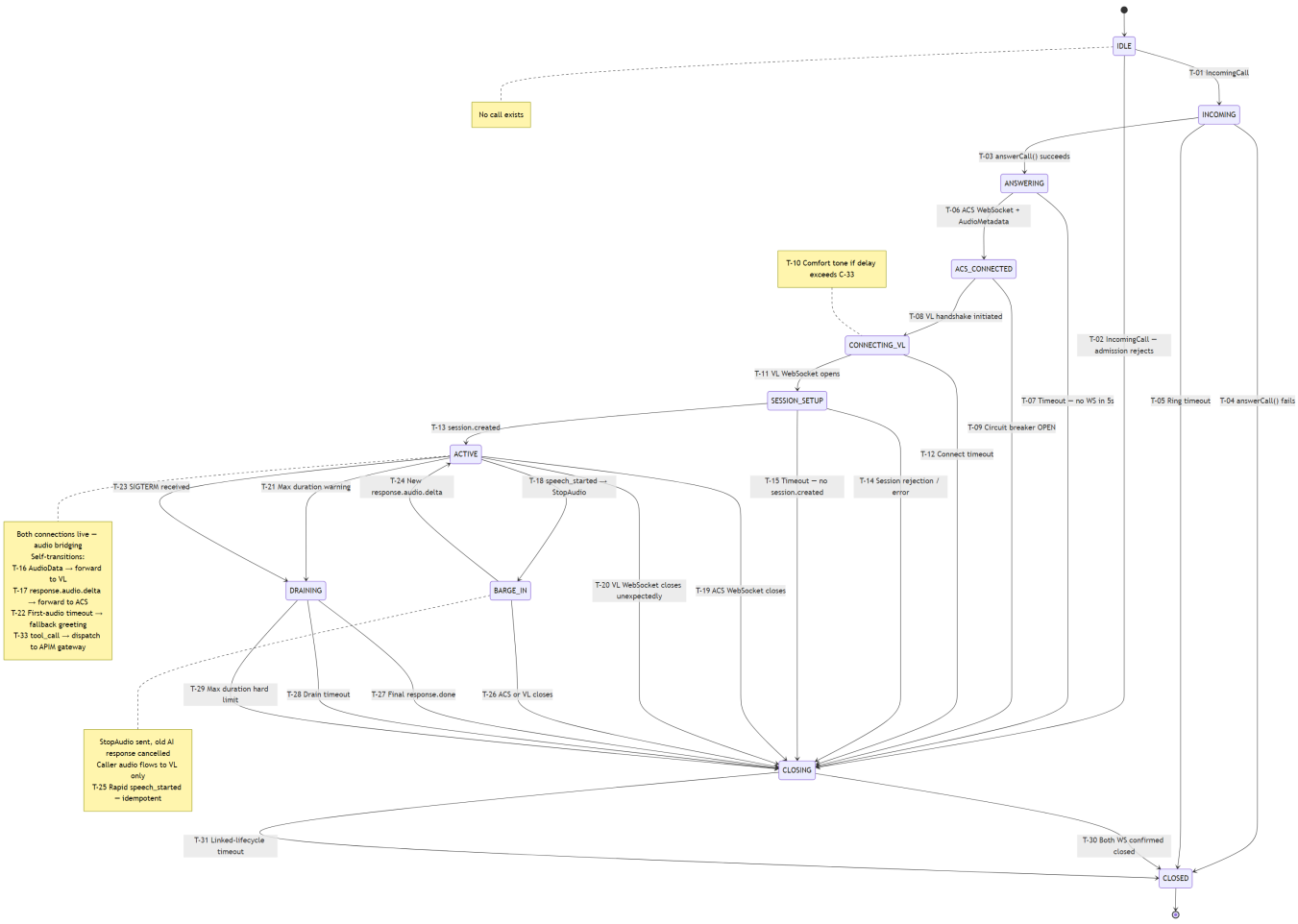
The lifecycle of a single call modeled as a finite state machine. Eliminates ambiguous state transitions and defines exactly what events are valid in each state.

States

State	Description	Active Connections	Audio Flowing?

State	Description	Active Connections	Audio Flowing?
IDLE	No call exists (initial/terminal)	None	No
INCOMING	Event Grid `IncomingCall` received, not yet answered	None	No
ANSWERING	`answerCall()` invoked, waiting for media WebSocket	None (HTTP in progress)	No
ACS_CONNECTED	ACS WebSocket established, AudioMetadata received	ACS ↔ Service	No
CONNECTING_VL	Voice Live WebSocket handshake in progress	ACS ↔ Service	No (comfort tone may play)
SESSION_SETUP	Voice Live open, `session.update` sent, awaiting `session.created`	ACS ↔ Service, Service ↔ VL	No
ACTIVE	Both connections live, audio bridging bidirectionally	ACS ↔ Service, Service ↔ VL	**Yes**
BARGE_IN	`speech_started` received — StopAudio sent, old AI response cancelled	ACS ↔ Service, Service ↔ VL	Caller → VL only
DRAINING	Graceful shutdown or max-duration approaching — wrapping up	ACS ↔ Service, Service ↔ VL	Yes (final response)
CLOSING	One or both connections closing, playing final prompt if needed	Partial	Possibly
CLOSED	Both connections terminated, resources freed	None	No

State Transition Diagram



Transition Table

#	From	Event / Trigger	Guard	To	Actions
T-01	IDLE	`IncomingCall`	Admission allows	INCOMING	Log #1, start answer timer

#	From	Event / Trigger	Guard	To	Actions
T-02	IDLE	`IncomingCall`	Admission rejects	CLOSING	Answer, play busy prompt (UX-REQ-11), hangup
T-03	INCOMING	`answerCall()` succeeds	—	ANSWERING	Log #2, register callback
T-04	INCOMING	`answerCall()` fails	—	CLOSED	Log error
T-05	INCOMING	Ring timeout (C-03)	—	CLOSED	Log warning
T-06	ANSWERING	ACS WebSocket + AudioMetadata	—	ACS_CONNECTED	Log #3, parse audio config
T-07	ANSWERING	Timeout (no WS in 5s)	—	CLOSING	Log error, hangup
T-08	ACS_CONNECTED	VL handshake initiated	Circuit CLOSED/HALF_OPEN	CONNECTING_VL	Start VL connect timer
T-09	ACS_CONNECTED	Circuit breaker OPEN	—	CLOSING	Play service-unavailable (REQ-R2.2), hangup
T-10	CONNECTING_VL	Comfort delay elapsed (C-33)	Still connecting	*(self)*	Play comfort tone (UX-REQ-1)
T-11	CONNECTING_VL	VL WebSocket opens	—	SESSION_SETUP	Log #5, send `session.update`
T-12	CONNECTING_VL	Connect timeout (C-10)	—	CLOSING	Log error, circuit breaker failure, play error, hangup
T-13	SESSION_SETUP	`session.created`	—	ACTIVE	Log #7, start first-audio timer, record VL connect latency
T-14	SESSION_SETUP	Session rejection/error	—	CLOSING	Log error, play error prompt, hangup
T-15	SESSION_SETUP	Timeout (C-32, no `session.created`)	—	CLOSING	Log error, play fallback greeting, hangup
T-16	ACTIVE	`AudioData` from ACS	Not silent	*(self)*	Forward to VL as `input_audio_buffer.append`, log #8 (DEBUG)
T-17	ACTIVE	`response.audio.delta` from VL	—	*(self)*	Convert to `OutStreamingData`, send to ACS, log #9 (DEBUG)
T-18	ACTIVE	`speech_started` from VL	—	BARGE_IN	**Priority:** Send `StopAudio` immediately (UX-REQ-8), log #10
T-19	ACTIVE	ACS WebSocket closes	—	CLOSING	Start linked-lifecycle timer (C-31), close VL
T-20	ACTIVE	VL WebSocket closes unexpectedly	—	CLOSING	Play apology (UX-REQ-10), close ACS, log #6
T-21	ACTIVE	Max duration warning (C-29 − C-30)	—	DRAINING	Inject wrap-up to VL (UX-REQ-9)
T-22	ACTIVE	First-audio timeout (C-32)	No audio received yet	*(self)*	Play fallback greeting (UX-REQ-3), log WARN
T-23	ACTIVE	SIGTERM received	—	DRAINING	Send "one moment" to VL (REQ-R3.2)
T-33	ACTIVE	`tool_call` from VL	—	*(self)*	Dispatch HTTPS to APIM gateway (FR48), return `tool_call_output` (FR49), log #21/#22/#23. Audio bridge continues uninterrupted.
T-24	BARGE_IN	New `response.audio.delta`	—	ACTIVE	Resume audio bridging VL → ACS

#	From	Event / Trigger	Guard	To	Actions
T-25	BARGE_IN	`speech_started` again (rapid)	—	*(self)*	Idempotent StopAudio (UX-REQ-5)
T-26	BARGE_IN	ACS or VL closes	—	CLOSING	Same as T-19/T-20
T-27	DRAINING	Final `response.done`	—	CLOSING	Close both WebSockets gracefully
T-28	DRAINING	Drain timeout (C-34)	—	CLOSING	Force-close both
T-29	DRAINING	Max duration hard limit	—	CLOSING	Force-close
T-30	CLOSING	Both WebSockets confirmed closed	—	CLOSED	Log #14, record `ivr.call.duration`, free resources
T-31	CLOSING	Linked-lifecycle timeout (C-31)	Peer still open	CLOSED	Force-close remaining, log warning
T-32	CLOSED	*(terminal)*	—	IDLE	Decrement `ivr.calls.active`, GC call context

Invalid Transitions (Guards)

If any of these occur, log `ERROR` and force-transition to CLOSING:

From	Invalid Event	Why
IDLE	Audio data	No call exists
INCOMING	Audio data	WebSocket not yet established
ANSWERING	`session.created`	Voice Live not yet contacted
CONNECTING_VL	`response.audio.delta`	Session not yet created
SESSION_SETUP	Forward audio to VL	Only after ACTIVE
CLOSED	Any event	Call already terminated

Concurrency Model

- Each call has its **own state machine instance** — no shared mutable state between calls
- State transitions MUST be **atomic** — use `AtomicReference<CallState>` or synchronized block per call
- The BARGE_IN state prevents sending stale audio from the cancelled AI response — bridge checks state before forwarding VL → ACS audio
- `StopAudio` is **idempotent** — sending it multiple times is harmless (UX-REQ-5)

State × Configuration Cross-Reference

State	Key Config Properties
INCOMING	C-03 (ring timeout), C-26/C-27 (admission)
ANSWERING	C-01 (callback URL)
ACS_CONNECTED	C-04 (ws path), C-05/C-06 (ws config)
CONNECTING_VL	C-07/C-08 (VL endpoint), C-10 (connect timeout), C-33 (comfort delay)
SESSION_SETUP	C-09/C-12–C-18 (session config), C-32 (first-audio timeout)
ACTIVE	C-05/C-11 (idle timeouts), C-19–C-21 (audio), C-29/C-30 (max duration)
DRAINING	C-34 (drain timeout)
CLOSING	C-31 (linked lifecycle timeout), C-35–C-39 (prompt files)

STRIDE Threat Model

Method: STRIDE per trust boundary — systematic identification of Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege threats across 6 trust boundaries.

Trust Boundaries Analyzed

#	Boundary	From → To
TB-1	Internet → ACS Platform	Caller PSTN/SIP → Azure Communication Services
TB-2	ACS Platform → IVR Service	Event Grid webhooks, mid-call callbacks, WebSocket audio
TB-3	IVR Service → Voice Live API	Outbound WSS to Azure AI Foundry
TB-4	IVR Service → Key Vault	Secret retrieval at startup
TB-5	IVR Service → APIM → Backends	Outbound HTTPS to Oracle BRM, CRM, etc. via API gateway (MVP — 3 tool calls)
TB-6	Container Platform → IVR Service	Health probes, management endpoints

Findings

ID	Trust Boundary	STRIDE Category	Threat	Severity	Mitigation	Status
S-1	TB-2 (ACS→IVR)	**Spoofing**	WebSocket upgrade request without JWT — anyone who discovers WS URL can connect and receive/inject audio	**CRITICAL**	JWT validation MUST be P0, not P1. Validate `Authorization` header *during handshake* before upgrading. Use ACS OIDC metadata for key rotation.	**P0 → SEC-REQ-1**
S-2	TB-2 (ACS→IVR)	**Information Disclosure**	Audio bytes logged at DEBUG/TRACE level would expose caller voice data (biometric PII under LGPD)	HIGH	Never log audio payload content. Log only metadata (size, sequence). Zero-storage guarantee.	P0 → SEC-REQ-5
S-3	TB-2 (ACS→IVR)	**Information Disclosure**	Phone numbers logged in full expose caller identity	MEDIUM	Mask all phone numbers to last 4 digits in every log appender. Use structured logging with a PII-masking filter.	P0 → SEC-REQ-6
S-4	TB-2 (ACS→IVR)	**Spoofing**	ACS callback URL is predictable (e.g., `/api/callbacks/12345`) — attacker can forge callback requests	MEDIUM	Callback URL must include a cryptographically random token (`/api/callbacks/{callId}?token={random}`).	P0 → SEC-REQ-4
S-5	TB-2 (ACS→IVR)	**Tampering / Replay**	Event Grid event replayed to trigger duplicate call answering	MEDIUM	Validate `eventTime` — reject events older than configurable max age (default: 5 min). Idempotent call handling.	P0 → SEC-REQ-14
S-6	TB-3 (IVR→VL)	**Elevation of Privilege**	Vulnerable transitive dependency (e.g., Log4Shell-class) could allow remote code execution	HIGH	OWASP dependency-check in CI pipeline. Fail build on CVSS ≥ 7. Container image scanning (Trivy/Grype).	P1 → SEC-REQ-16
S-7	TB-3 (IVR→VL)	**Denial of Service**	AI model abuse (prompt injection, resource exhaustion) — attacker keeps calls open to drain Foundry quota	MEDIUM	Max call duration limit (C-29/C-30). Abuse monitoring via Azure AI content safety. Opt-out flag where applicable.	P1 → SEC-REQ-13
S-8	TB-4 (IVR→KV)	**Elevation of Privilege**	Key Vault RBAC grants more than GET — service could modify/delete secrets	LOW	RBAC: `Key Vault Secrets User` (GET-only). Never `Key Vault Administrator`. Audit log via Key Vault diagnostics.	P1 (operational)
S-9	TB-1 (Internet→ACS)	**All**	Should we add WAF or Azure Firewall?	N/A	**No.** All inbound is from known Microsoft IP ranges (ACS, Event Grid). WAF would break Event Grid validation and can't handle binary WS framing. Azure Firewall costs ~$1K/mo with minimal benefit. NSGs for IP allowlisting are sufficient for V1.	**Decision: No WAF/Firewall**

LGPD Impact Assessment (Simple Items — In Scope)

#	LGPD Requirement	Implementation	Complexity
1	Zero audio storage guarantee	Audio is forwarded in real-time; no buffer persisted to disk or storage. `tmpdir` is ephemeral container memory only.	Simple ☑
2	PII masking in logs	Structured logging with phone-number masking filter (last 4 digits only). No caller name in logs.	Simple ☑
3	Data residency — Brazil South	All Azure resources deployed in `brazilsouth` region. No cross-region replication for V1.	Simple ☑
4	Data processing record (Art. 37)	Documented in operational runbook: what data, why, how long (zero retention), who has access.	Simple ☑

Deferred (Complex): Explicit consent for AI voice processing (Art. 11), Data Processing Agreement (DPA) with Azure AI services.

Attack Surface Map

Method: Systematic enumeration of all entry points, outbound connections, and data flows — identifying gaps not covered by other discovery methods.

Entry Points (Inbound)

ID	Surface	Protocol	Path	Auth Mechanism	Data Sensitivity	Threat Level
EP-1	Event Grid Webhook	HTTPS POST	`/api/events`	Entra ID bearer token (validated via Event Grid SDK)	Low (event metadata: callId, from, to phone numbers)	Medium
EP-2	ACS Mid-Call Callbacks	HTTPS POST	`/api/callbacks/{callId}?token={random}`	Signed JWT in `Authorization` header	Medium (call state transitions, callId)	Medium
EP-3	ACS WebSocket Audio	WSS	`/ws`	JWT in `Authorization` header during upgrade	**Critical** (raw PCM audio — biometric data)	**High** ⚠
EP-4	Health / Readiness Probes	HTTP GET	`/actuator/health`, `/actuator/ready`	None (internal only — not exposed externally)	None	Low
EP-5	Spring Boot Actuator	HTTP	`/actuator/*`	⚠ **UNADDRESSED** — must be locked down	**Critical** (can expose secrets via `/actuator/env`)	**Critical**

Outbound Connections

ID	Target	Protocol	Auth	Data Sensitivity
OC-1	Voice Live API	WSS (Private Endpoint)	`DefaultAzureCredential` → `Cognitive Services User` RBAC	Critical (audio + AI session)
OC-2	ACS Call Automation API	HTTPS	`DefaultAzureCredential` → ACS RBAC	Medium (call control commands)
OC-3	Key Vault	HTTPS (Private Endpoint)	Managed Identity	High (secrets retrieval — GET only)
OC-4	APIM API Gateway	HTTPS	`DefaultAzureCredential` → Entra ID bearer token (scope C-57b)	Medium (tool call request/response: user data, invoice operations)

Data Sensitivity Inventory

Data Type	Classification	Where It Flows	Storage	LGPD Category
Raw PCM Audio	**Critical / Biometric**	ACS→IVR→Voice Live (real-time)	**Zero** — forwarded in memory only	Sensitive Personal Data (Art. 5-II)
Phone Numbers	**PII**	Event Grid events, callback URLs, logs	Masked in logs (last 4 digits)	Personal Data (Art. 5-I)
Call IDs	Internal Identifier	All components (correlation)	In-memory + logs	Non-personal (system-generated UUID)
AI Conversation	Sensitive	IVR↔Voice Live JSON events	Transient (Voice Live session only)	Personal Data context

Data Type	Classification	Where It Flows	Storage	LGPD Category
JWT Tokens	Secret	HTTP headers	Never logged	Security credential

Gaps Found

ID	Gap	Severity	Remediation	New Requirement
AS-1	Spring Boot Actuator endpoints not addressed in any PRD section — `/actuator/env` can expose all config including secrets	**Critical**	Bind actuator to separate management port (8081). Expose only `/health` and `/ready` on management port. Disable all other actuator endpoints.	SEC-REQ-9
AS-2	Error responses may contain stack traces, class names, Spring internal details	High	Configure `server.error.include-stacktrace=never`, custom error handler returns generic JSON	SEC-REQ-10
AS-3	ACS connection string stored in Key Vault is itself a secret that could be leaked	High	**Eliminated** — use `DefaultAzureCredential` with `ivr.acs.endpoint` (non-secret URL). No connection string anywhere.	SEC-REQ-11
AS-4	Voice Live API key could be used as fallback auth	Medium	Disable API key access on Foundry resource. Force managed identity only.	SEC-REQ-12
AS-5	No rate limiting on WebSocket connections — single IP could exhaust connection pool. Gateway HTTP connection pool not bounded.	Medium	Per-IP rate limiter on WebSocket handshake endpoint. Default: 10 connections per 60 seconds. `java.net.http.HttpClient` connection pool bounded by C-59 max concurrent tool calls.	SEC-REQ-15
AS-6	No mechanism to detect or prevent prompt injection in system prompt or user audio	Medium	System prompt guardrails + Azure AI Content Safety integration for abuse detection.	SEC-REQ-17
AS-7	Dependency vulnerabilities not checked in CI pipeline	Medium	Add OWASP dependency-check + container image scan to PR pipeline.	SEC-REQ-16

Security Requirements

Synthesized from STRIDE Threat Model (S-1 through S-9) + Attack Surface Mapping (AS-1 through AS-7). Total: 18 security requirements.

P0 — Must Have for MVP

ID	Requirement	Source	Acceptance Criteria	Config
SEC-REQ-1	**WebSocket JWT Validation** — Validate JWT in `Authorization` header during WebSocket upgrade handshake. Reject with HTTP 401 before upgrading if invalid/expired/missing. Use ACS OIDC metadata endpoint for key rotation.	S-1	AC-15	C-40, C-41, C-42, C-44
SEC-REQ-2	**Event Grid Entra ID Validation** — Validate bearer token on `/api/events` using Event Grid SDK.	S-5	(existing AC-5)	C-43
SEC-REQ-3	**ACS Callback JWT Validation** — Validate signed JWT on all mid-call callback requests.	S-4	(existing AC-6)	C-40
SEC-REQ-4	**Non-Guessable Callback URLs** — Include cryptographically random token in callback URL path/query. Minimum 32 characters (configurable).	S-4	(implicit in AC-6)	C-50
SEC-REQ-5	**Zero Audio Storage** — Audio bytes are forwarded in real-time via in-memory buffers only. No disk writes, no temporary files, no blob storage for audio. Container filesystem is ephemeral.	S-2	AC-17	—
SEC-REQ-6	**PII Masking** — All phone numbers in log output masked to last N digits (default: 4). Implemented as a structured logging filter applied to all log appenders.	S-3	AC-18	C-49
SEC-REQ-9	**Actuator Lockdown** — Bind Spring Boot Actuator to separate management port (default: 8081). Expose only `/actuator/health` and `/actuator/ready`. Disable all other actuator endpoints. Management port is internal-only (no external load balancer routing).	AS-1	AC-19	C-48

ID	Requirement	Source	Acceptance Criteria	Config
SEC-REQ-10	**Error Response Hardening** — HTTP error responses contain no stack traces, no class names, no Spring-specific internal details. Configure `server.error.include-stacktrace=never`. Custom error handler returns generic JSON `{"error": "Internal Server Error"}`.	AS-2	AC-20	—
SEC-REQ-11	**ACS Managed Identity Auth** — Use `DefaultAzureCredential` for all ACS Call Automation API calls. No connection strings anywhere (code, config, Key Vault). Config property is `ivr.acs.endpoint` (non-secret URL).	AS-3	AC-22	C-02
SEC-REQ-14	**Event Replay Protection** — Validate `eventTime` on Event Grid events. Reject events older than configurable max age (default: 5 minutes).	S-5	AC-16	C-45

P1 — Important, Post-MVP OK

ID	Requirement	Source	Acceptance Criteria	Config
SEC-REQ-7	**Data Residency** — All Azure resources deployed in `brazilsouth` region. No cross-region replication for V1.	LGPD	—	—
SEC-REQ-8	**LGPD Data Processing Record** — Maintain Art. 37 record documenting what personal data is processed, purpose, retention (zero), and access controls.	LGPD	—	—
SEC-REQ-12	**Voice Live Managed Identity + Key Disable** — Authenticate to Voice Live API using `DefaultAzureCredential`. Disable API key access on the Foundry resource to prevent key-based auth fallback.	AS-4	—	—
SEC-REQ-13	**Abuse Monitoring Opt-Out** — Where applicable, opt out of Azure AI abuse monitoring that would store audio/transcripts for review. Document opt-out in operational runbook.	S-7	—	—
SEC-REQ-15	**WebSocket Rate Limiting** — Per-IP rate limiter on WebSocket handshake endpoint. Default: 10 connections per 60 seconds. Configurable.	AS-5	—	C-46, C-47
SEC-REQ-16	**Dependency Vulnerability Scanning** — OWASP dependency-check in PR pipeline (fail on CVSS ≥ 7). Container image scanning (Trivy or Grype) for Critical/High CVEs.	AS-7, S-6	AC-21	—
SEC-REQ-17	**System Prompt Guardrails** — System prompt includes explicit instructions to refuse off-topic requests and never reveal internal system details. Azure AI Content Safety integration for abuse detection.	AS-6	—	C-12
SEC-REQ-18	**Gateway Auth Scope Validation** — Tool call HTTP requests to the APIM gateway SHALL authenticate using `DefaultAzureCredential` with the configured Entra ID scope (C-57b). The acquired bearer token SHALL be sent in the `Authorization` header. If token acquisition fails, the tool call SHALL fail gracefully (FR50 error path) rather than sending an unauthenticated request.	TB-5, OC-4	AC-23	C-57b

Security × Config Cross-Reference

SEC-REQ	Config Properties
SEC-REQ-1	C-40 (jwt.enabled), C-41 (jwt.issuer), C-42 (jwt.audience), C-44 (acs-oidc-metadata-url)
SEC-REQ-2	C-43 (event-grid.validation-enabled)
SEC-REQ-4	C-50 (callback-token-length)
SEC-REQ-6	C-49 (phone-mask-visible-digits)
SEC-REQ-9	C-48 (management.server.port)
SEC-REQ-11	C-02 (acs.endpoint)
SEC-REQ-14	C-45 (max-event-age-seconds)
SEC-REQ-15	C-46 (rate-limit-per-ip), C-47 (rate-limit-window-seconds)

SEC-REQ	Config Properties
SEC-REQ-18	C-57b (gateway-auth-scope)

Security Decision Log

Decision	Rationale
No WAF or Azure Firewall for V1	All inbound is from known Microsoft IP ranges (ACS, Event Grid). WAF breaks Event Grid validation headers and can't handle binary WebSocket framing. Azure Firewall costs ~$1K/mo with minimal benefit. NSGs for IP allowlisting are sufficient.
No ACS Connection String	Both connection string and `DefaultAzureCredential` require Azure connectivity — no offline benefit. Eliminating the connection string removes a secret from Key Vault and simplifies rotation. `DefaultAzureCredential` works everywhere: managed identity in prod, `az login` locally.
JWT validation is P0, not P1	Without JWT on WebSocket, anyone who discovers the URL can connect and receive live caller audio (biometric PII). This is the highest-impact security gap in the entire system.
Actuator on separate port	`/actuator/env` can expose all configuration values including secrets resolved from Key Vault. Binding to a separate internal-only port prevents external access without disabling health probes.
LGPD: 4 simple items in, 2 complex deferred	Zero audio storage, PII masking, Brazil South residency, and data processing record are straightforward. Explicit consent for AI processing and DPA with Azure AI are complex legal/product decisions — deferred to post-MVP.

Success Criteria

User Success (Caller)

- Caller hears AI greeting within **2 seconds** of call being answered — no dead air
- Caller can **interrupt the AI mid-sentence** and be heard immediately (barge-in latency < 100ms)
- If the system can't connect to the AI, caller hears a **polite apology** — never silence or a dial tone
- Conversation feels natural — **no IVR menus**, no awkward pauses, no "press 1 for..."

Technical Success

- Bidirectional **PCM 24K mono** audio stream between ACS and Voice Live works end-to-end
- **50 concurrent calls** per instance sustained without degradation
- Call completion rate **≥ 99%** (calls that are answered reach a graceful end)
- **Zero audio data at rest** — audio forwarded in real-time through memory only
- All P0 security requirements pass validation (JWT, replay protection, rate limiting, actuator lockdown, error hardening)
- Full integration test suite runs **without Azure credentials** (FakeAcsClient + FakeVoiceLiveServer + WireMock gateway stub)
- OWASP dependency-check passes (no CVSS ≥ 7)
- Single `./mvnw clean package` produces a deployable artifact
- Micrometer metrics (auto-config + custom) + structured JSON logging from Day 1

Business Success (Accelerator)

- **Day 3 milestone:** First real phone call through ACS reaches AI and gets a spoken response — tool call round-trip with APIM gateway proven (placeholder system prompt OK)
- **Day 5 milestone:** Sustained production traffic with real callers, all security hardened, production system prompt deployed
- **Clone-to-first-call < 30 minutes** — developer experience for onboarding new deployers
- System prompt is the **only per-client customization** — loaded from external config (C-12), authored by domain specialists (not the dev)

Measurable Outcomes

Metric	Target	Measurement
Time-to-first-audio	< 2 seconds	From `CallConnected` event to first audio byte sent to ACS
Barge-in latency	< 100ms	From `speech_started` event to `StopAudio` sent to ACS
Call completion rate	≥ 99%	Calls answered / calls reaching graceful end
Concurrent calls	50 per instance	Load test with simulated audio streams
Cloud-free test suite	100% pass	`./mvnw test` with no Azure credentials configured
Tool call round-trip	< 5 seconds (p95)	From `tool_call` event received to `tool_call_output` sent back to Voice Live
Auth rejection accuracy	100%	Invalid/expired/missing JWT → 401, connection never upgrades

Product Scope

MVP — 5-Day Security-Hardened Production

One developer, Copilot-assisted. Azure resources pre-provisioned. System prompt authored by domain specialists.

Core Audio Bridge:

- Event Grid webhook → answer call with `MediaStreamingOptions` (bidirectional, PCM 24K mono)
- ACS WebSocket server (`@ServerEndpoint`, JSR 356) — receives/sends audio from/to ACS
- Voice Live WebSocket client (`java.net.http.WebSocket`) — connects to `wss://<foundry>/voice-agent/realtime`
- Bidirectional audio forwarding (ACS ↔ Voice Live) in real-time
- Barge-in support (`speech_started` → `StopAudio`)
- Graceful error handling with fallback audio prompts (busy, apology, timeout)

Tool Call Framework (3 predefined tool calls via APIM):

- Voice Live sends `tool_call` events → bridge dispatches HTTPS request to APIM gateway (`C-57`)
- `java.net.http.HttpClient` on virtual threads — blocking `.send()` is effectively non-blocking
- Auth: `DefaultAzureCredential` acquires bearer token scoped to APIM audience (`C-57b`)
- Response timeout per tool call (`C-58`, default 10s); call stays ACTIVE during tool execution
- Tool call result forwarded back to Voice Live as `tool_call_output` event
- 3 tool calls: (1) GET user data, (2) send invoice to registered email, (3) send invoice to user-provided email
- WireMock stub server in test profile for gateway simulation
- Tool definitions file (`C-60`) loaded at session init — externalized, not hardcoded

Security (all in-scope for 5 days):

- JWT validation on WebSocket handshake + ACS callbacks (SEC-REQ-1, 3)
- Non-guessable callback URLs with crypto-random tokens (SEC-REQ-4)
- Zero audio storage guarantee (SEC-REQ-5)
- Event Grid replay protection — reject events > 5 min old (SEC-REQ-14)
- Actuator lockdown — management port 8081, health/ready only (SEC-REQ-9)
- Error response hardening — no stack traces to clients (SEC-REQ-10)
- WebSocket per-IP rate limiting (SEC-REQ-15)
- OWASP dependency-check in CI pipeline (SEC-REQ-16)
- `DefaultAzureCredential` everywhere — no connection strings (SEC-REQ-11)
- Event Grid Entra ID validation via SDK (SEC-REQ-2)

Observability:

- Micrometer auto-config on Day 1, custom metrics added incrementally per feature
- Structured JSON logging (Logback + JSON encoder)
- All 23 log events defined in Observability Requirements
- All 14 metrics defined in Observability Requirements
- Health + readiness probes via Actuator

Infrastructure:

- Externalized config via Spring profiles (test / local / prod)
- Cloud-free test suite (FakeAcsClient + FakeVoiceLiveServer)
- Call admission control (reject when at capacity)
- Graceful shutdown (drain active calls before termination)

Day-by-Day Plan: See the Authoritative Day-by-Day Sprint Plan in the Scoping section for the detailed 30-item breakdown with per-day traceability.

Key Constraint: Day 3 milestone uses a placeholder system prompt. Production prompt (authored by domain specialists) swapped in via config property `C-12` on Day 5 deploy.

Growth — Post-MVP

- PII masking in logs (SEC-REQ-6) — custom Logback filter with phone-number regex
- Extended tool calls — additional Oracle BRM, CRM operations beyond initial 3 via APIM
- Multi-language support (voice + system prompt per locale)
- Call transfer to human agent (warm/cold)
- LGPD full compliance — consent for AI processing (Art. 11), DPA with Azure AI
- Voice Live API key disable (SEC-REQ-12) + abuse monitoring opt-out (SEC-REQ-13)
- System prompt guardrails (SEC-REQ-17)

- Auto-scaling with KEDA based on active call count
- A/B testing of system prompts
- Analytics dashboard (call duration, resolution rates, drop-off points)
- Container image scanning (Trivy/Grype) in CI

User Journeys

Derived from Step 4 — User Journey Mapping + Critique & Refine elicitation. 5 journeys covering 4 user types, exercising 22 of 24 acceptance criteria.

Journey 1: Maria — The Caller (Happy Path)

Persona: Maria, 34, telecom customer calling from her mobile phone during lunch break.

Maria dials the company number. The call rings once. In **1.2 seconds**, ACS answers her call — Event Grid delivers `IncomingCall` (T-01), the service calls `answerCall()` (T-03), and the ACS WebSocket connects with `AudioMetadata` (T-06). The state machine advances: IDLE → INCOMING → ANSWERING → ACS_CONNECTED.

Meanwhile, the service opens a WebSocket to Voice Live (T-08 → T-11). The connection establishes in 800ms — well under the 2-second comfort tone threshold (`C-33`). Had it taken longer than 2 seconds, Maria would have heard a brief comfort tone bridging the silence (UX-REQ-1, T-10). But today, the connection is fast. Voice Live receives `session.update` with the client's system prompt (`C-12`), semantic VAD configuration (`C-14`–`C-18`), and the voice model (`C-13`). `session.created` arrives (T-13) — the state reaches ACTIVE.

At T+1.8 seconds from answer, the AI greeting begins streaming. Maria hears: "*Olá! Bem-vinda à [empresa]. Como posso te ajudar?*" The `ivr.time_to_first_audio` metric records **1.8 seconds** — well under the 3-second target (UX-REQ-4, AC-9).

Maria says: "*Quero saber o saldo da minha fatura.*" The audio flows through the bridge — `AudioData` packets forwarded to Voice Live as `input_audio_buffer.append` (T-16). Voice Live's semantic VAD detects end of speech.

Now the tool call round-trip (T-33): Voice Live determines it needs Maria's account data to answer. It fires a `tool_call` event — function `get_user_data`, argument `{"phone": "+5511999887766"}`. The bridge receives the event, dispatches an HTTPS request to the APIM gateway (FR48) using `DefaultAzureCredential` for auth (SEC-REQ-18, C-57b). WireMock (in test) or the real APIM gateway (in prod) responds with Maria's account JSON in **1.2 seconds** — well under the 5-second timeout (C-58, NFR-P10). The bridge wraps the response as `tool_call_output` and sends it back to Voice Live (FR49). The `ivr.tool_call.latency` timer records 1.2s. Within **600ms** of receiving the tool output, the AI responds with her balance information. The `ivr.ai.response.latency` metric ticks at 0.6 seconds (UX-REQ-7).

Then the AI starts explaining late payment options. Maria already knows — she **interrupts mid-sentence**: "*Não, já sei disso. Quero pagar agora.*"

Voice Live detects her speech (T-18) and fires `speech_started`. **Within 80ms**, the bridge sends `StopAudio` to ACS (UX-REQ-8, AC-3). The AI's voice cuts off immediately — Maria hears silence, then herself talking. The state transitions to BARGE_IN. She finishes her question.

Now the critical round-trip (T-24): Voice Live processes Maria's interruption and generates a new response. The first `response.audio.delta` arrives — the state transitions BARGE_IN → ACTIVE (T-24). The AI says: "*Claro! Vou gerar o código de pagamento agora.*" Maria hears the direct response to what she actually asked, not a continuation of the previous topic. The conversation flows naturally — she interrupted, was heard, and got a relevant answer.

Maria gets her payment code, says "*Obrigada!*", and hangs up. The ACS WebSocket closes (T-19), triggering linked lifecycle — Voice Live connection closes within 3 seconds (C-31). State: CLOSING → CLOSED (T-30). Total call: **25 seconds**. Log event #14 records: `totalDurationMs=25000, audioPacketsForwarded=1200, bargeInCount=1, toolCallCount=1`.

Requirements exercised: AC-1, AC-2, AC-3, AC-9, AC-10, AC-23, UX-REQ-1–5, UX-REQ-8, T-01 through T-19, T-24, T-30, T-33 | **Config touched:** C-01, C-02, C-07, C-09, C-12–C-18, C-29, C-31, C-33, C-57, C-57b, C-58

Journey 2: Carlos — The Caller (Failure Paths)

Persona: Carlos, 52, calling from a landline on a busy weekday afternoon.

Path A: Admission Rejection

Carlos calls during a spike. The service is at 42 active calls — **84% of its 50-call capacity** (`C-26`, `C-27`). The admission control guard fires (T-02): state goes IDLE → CLOSING. But Carlos doesn't hear silence — the service answers briefly, plays the "*Nossos atendentes estão todos ocupados. Por favor, tente novamente em alguns instantes.*" prompt (`C-38`), then hangs up gracefully (UX-REQ-11, AC-12).

Carlos waits 2 minutes and calls again. The spike has passed — his call goes through normally.

Requirements exercised: AC-7, AC-12, UX-REQ-11, REQ-R4.1, T-02 | **Config:** C-26, C-27, C-38

Path B: Voice Live Failure Mid-Call

Carlos's second call gets through. He's in conversation (ACTIVE state) when Voice Live's WebSocket drops unexpectedly (T-20). The bridge detects the closure immediately. Before Carlos can wonder why the AI went silent, he hears: "*Desculpe, tivemos um problema técnico. Vamos encerrar a ligação. Por favor, ligue novamente.*" (`C-37`, UX-REQ-10, AC-13). The service closes the ACS connection (T-30), and Carlos hears the normal disconnect tone — not dead silence.

Log event #6 records the Voice Live disconnection with `closeCode=1006` (abnormal). The circuit breaker increments its failure counter. If Carlos's call was the 3rd consecutive failure (`C-22`), the circuit trips (REQ-R2.1) — subsequent callers hit Path A-style rejection with the service-unavailable prompt (`C-39`) instead of waiting for a doomed Voice Live connection.

Requirements exercised: AC-4, AC-5, AC-13, UX-REQ-10, FM-1, FM-2, T-20 | **Config:** C-22, C-31, C-37, C-39

Path C: Max Duration — The Long Conversation

Carlos calls a third time — this one goes well. He's chatting with the AI about multiple billing questions. The conversation is productive but long. At **9 minutes and 30 seconds** (C-29 = 600s minus C-30 = 30s), the state transitions from ACTIVE → DRAINING (T-21).

The service injects a context message to Voice Live, prompting the AI to wrap up naturally (UX-REQ-9). The AI says: "*Carlos, foi ótimo te ajudar com essas questões. Tem mais alguma coisa rápida que eu possa resolver?*" Carlos says no, thanks the AI, and hangs up. He never noticed the nudge — it blended seamlessly into the conversation.

Had Carlos kept talking past 10 minutes, the drain timeout (`C-34`) would have triggered a force-close (T-28, T-29) — but with the natural wrap-up, it never came to that.

Requirements exercised: AC-14, UX-REQ-9, REQ-R3.5, T-21, T-27 | **Config:** C-29, C-30, C-34

Path D: Security Rejection (Invisible to Caller)

An automated script discovers the WebSocket URL and attempts to connect without a valid JWT. The handshake fails at HTTP 401 — the WebSocket never upgrades (SEC-REQ-1, AC-15). Log event #16 fires with `reason=jwt_missing`. The `ivr.security.auth.rejected` metric increments with tag `jwt_missing`.

Separately, a replayed Event Grid event (eventTime 8 minutes ago) arrives at `/api/events`. The service validates the timestamp, finds it exceeds `C-45` (300s max age), and rejects it (SEC-REQ-14, AC-16). Log event #18 fires.

No caller is affected — these are attacker interactions, not legitimate calls. The legitimate caller (Carlos, routed through ACS with valid credentials) was never at risk. The rejections are visible only to Renata on her security dashboard.

Requirements exercised: AC-15, AC-16, SEC-REQ-1, SEC-REQ-14, SEC-REQ-15 | **Config:** C-40–C-42, C-44, C-45, C-46–C-47

Journey 3: Renata — The Operations Engineer

Persona: Renata, 29, SRE responsible for monitoring the IVR system in production. Working her afternoon shift.

Scene 1: Routine Performance Monitoring

Renata opens her Grafana dashboard. The `ivr.calls.active` gauge shows 23 concurrent calls — well within the 50-call capacity (`C-26`). The `ivr.time_to_first_audio` timer shows p95 at 2.1 seconds — healthy. Barge-in effectiveness (`ivr.bargein.effectiveness`) is at 97% — 97% of barge-ins have StopAudio arriving within 100ms. She drills into `ivr.call.duration` and sees a healthy distribution: median 45 seconds, p95 at 3 minutes.

She glances at the **security panel**. The `ivr.security.auth.rejected` counter shows a steady low baseline — about 5 `jwt_invalid` rejections per hour, likely from automated scanners probing the endpoint. The `ivr.security.websocket.rate.limited` counter shows zero — no rate limiting triggers today. Normal day.

Scene 2: Latency Incident

At 14:30, PagerDuty fires: `ivr.voicelive.connect.latency p95 > 3000ms`. Renata checks the dashboard — Voice Live connection latency spiked to 4.2 seconds. Calls are still completing, but callers are hearing the comfort tone (`C-33`) while waiting for the AI. The `ivr.time_to_first_audio` p95 jumped to 4.8 seconds.

She clicks into the structured JSON logs (filtered by `component=websocket-client`) and sees log event #5 with `connectLatencyMs` values clustering around 4000–4500ms. No full disconnects — the circuit breaker hasn't tripped (log event #11 is absent). She checks Azure AI Foundry's status page — regional degradation in Brazil South. She sends a heads-up to the team and continues monitoring. Latency returns to normal within 20 minutes.

Scene 3: Security Alert Spike

At 16:00, her security alert fires: `ivr.security.auth.rejected` spiked to 200 rejections in the last 5 minutes, all tagged `jwt_invalid`, all from a single IP address. She drills into the logs — event #16 repeats 200 times with the same `remoteAddr` and `reason=jwt_invalid`. Then she notices event #20

(rate limit exceeded) starting for that same IP — the per-IP rate limiter (`C-46` = 10 per 60s) kicked in.

She cross-references with the `ivr.calls.active` gauge — no impact on legitimate callers. The attacker never got past the handshake. She files an incident report, considers adding the IP to the NSG blocklist, and continues monitoring.

Scene 4: Deployment Verification

A new version deploys via rolling update. Renata verifies:

- `ivr.calls.active` didn't spike to zero (rolling deploy with `maxUnavailable: 0` — REQ-R3.4)
- Log event #13 (graceful shutdown) appears for old pods, followed by normal startup for new pods
- She runs a quick `curl` against the old actuator endpoint URL on port 443 — gets 404 (AC-19, SEC-REQ-9). The health probe on port 8081 (`C-48`) returns 200. Actuator lockdown confirmed.
- The OWASP dependency-check report in the CI pipeline shows **0 critical, 0 high** (AC-21, SEC-REQ-16). No new CVEs introduced.

Total incident resolution: **8 minutes** from alert to root cause.

Requirements exercised: All 14 metrics, log events #5, #6, #11, #13, #16, #20, AC-19, AC-21, SEC-REQ-9, SEC-REQ-15, SEC-REQ-16 | **Config:** C-26, C-33, C-46, C-48

Journey 4: Thiago — The Domain Specialist

Persona: Thiago, 45, customer experience manager. Knows the business deeply. Writes well. Cannot code.

Thiago receives the system prompt template from the development team — a plain text file with instructions for the AI persona, tone, conversation boundaries, and domain knowledge. He writes a comprehensive prompt in Portuguese, specifying:

- The AI persona ("Assistente virtual da [empresa], simpático e eficiente")
- Greeting format and tone
- Topics the AI should handle (billing, payments, service status, plan changes)
- Topics the AI should refuse ("Não posso ajudar com reclamações jurídicas — vou transferir")
- Escalation rules
- Language and formality level

He saves the file and hands it to the ops team. They update config property `C-12` (`ivr.voicelive.system-prompt`) in the production config — a **config-only deploy**. No code changes, no build, no developer required. The new prompt goes live within minutes (AP-6: business logic belongs in the system prompt, not the code).

The Feedback Loop

Thiago dials the ACS number from his desk phone. The AI answers with his custom greeting. He tests a billing question — the response is accurate but too formal. He tests an off-topic question — the AI correctly deflects. He tests an interruption (barge-in) — the AI handles it gracefully.

He opens his prompt file, adjusts the formality ("mais informal, como uma conversa entre amigos"), and sends the update. Config deploy #2 goes out. He calls again — better, but the escalation phrasing is awkward. Third iteration — nailed it.

Over his first week, Thiago makes **3 prompt iterations**, each validated by calling the number himself and listening. He never touches code, never opens a terminal, never sees a log file. His tool is the system prompt file and a telephone.

By Week 2, he's confident enough to write prompts for two other client deployments — each just a different `C-12` value. Same code, different conversations. The accelerator model validated.

Requirements exercised: C-12, AP-6 (retirement-ready), accelerator business model | **Config:** C-12 (only param that changes between clients)

Journey 5: Lucas — The Developer

Persona: Lucas, 27, backend developer joining the project. First day.

Phase 1: Zero-Config Tests (Minutes 0–7)

Lucas clones the repository and reads the one-page developer guide.

```
git clone <repo>                    # Minute 0
cd agentic-ivr
./mvnw test                         # Minute 3 (Maven downloads deps ~2 min)
```

48 tests pass. Zero failures. Zero cloud credentials configured. The `test` profile activates automatically — FakeAcsClient and FakeVoiceLiveServer run in-process (all at `localhost`, `C-40` = `false`). Lucas sees tests for:

- Happy path: call answered → audio bridged → call ended
- Barge-in: `speech_started` → StopAudio within 100ms
- Linked lifecycle: one WebSocket drops → both close within 3s
- Admission control: 6th call rejected when `C-26` = 5
- JWT validation: connection without token → 401 (even with `C-40=false` in test, a specific security test overrides this)
- **No audio in logs**: a test captures all log output during a call and asserts zero audio byte content appears at any log level (AC-17)

Lucas is impressed. He hasn't configured anything and the full test suite tells him the system works.

Phase 2: Real Services (Minutes 7–22)

Now he wants to hear it for real. He runs:

```
az login                            # Minute 7
devtunnel create --allow-anonymous  # Minute 8
devtunnel port create -p 8080
```

He edits `application-local.yml` with 3 values:

- `C-01`: his devtunnel URL as callback base
- `C-02`: the ACS endpoint (non-secret URL — SEC-REQ-11)
- `C-07`: the Voice Live WSS endpoint

```
./mvnw spring-boot:run -Dspring.profiles.active=local  # Minute 12
```

Spring Boot starts in **4 seconds** on virtual threads. Tomcat binds port 8080. Actuator binds port 8081 (`C-48`). Micrometer auto-config registers all 14 metrics. The logs stream in structured JSON.

Lucas registers the Event Grid subscription pointing to his devtunnel. He calls the ACS phone number from his mobile — **Minute 22**. He hears the AI greeting. He asks a question, gets an answer, interrupts (barge-in works!), and hangs up.

In the terminal, structured JSON log events #1 through #14 scroll past. He sees his call's full lifecycle — each entry tagged with `correlationId`. He checks `localhost:8081/actuator/health` — returns `UP`. He tries `localhost:8081/actuator/env` — **404** (AC-19). Actuator lockdown works.

Clone to first real call: 22 minutes. Exactly what the Success Criteria promised.

Requirements exercised: Developer setup guide, Spring profiles (test/local), FakeAcsClient, FakeVoiceLiveServer, WireMock gateway stub, AC-17 (no audio in logs), AC-19 (actuator lockdown), all 23 log events, DefaultAzureCredential (SEC-REQ-11) | **Config:** C-01, C-02, C-07, C-40, C-48, C-57, C-57b

---

Journey Requirements Summary

Journey	User Type	Key Capabilities Demonstrated	AC Coverage
1: Maria (Happy)	Caller	End-to-end audio, barge-in round-trip (BARGE_IN→ACTIVE), tool call round-trip (T-33), time-to-first-audio, comfort tone timing	AC-1, AC-2, AC-3, AC-9, AC-10, AC-23
2A: Carlos (Admission)	Caller	Admission control, busy prompt	AC-7, AC-12
2B: Carlos (VL Failure)	Caller	Linked lifecycle, circuit breaker, apology prompt	AC-4, AC-5, AC-13
2C: Carlos (Max Duration)	Caller	DRAINING state, natural wrap-up, graceful max-duration	AC-14
2D: Carlos (Security)	Attacker/Caller	JWT rejection, replay protection, rate limiting, zero caller impact	AC-15, AC-16
3: Renata (Ops)	Operations	All 14 metrics, security dashboard, log events, OWASP, actuator lockdown	AC-19, AC-21
4: Thiago (Specialist)	Domain	Config-only deploy, system prompt iteration, feedback loop	C-12, AP-6
5: Lucas (Dev)	Developer	Zero-config tests, clone-to-call < 30m, test profile, local profile, no audio in logs	AC-17, AC-19

AC coverage: 22 of 24 — AC-18 (PII masking) and AC-22 (no connection strings) are verified programmatically in CI/deployment, not through user journeys.

API Backend Specific Requirements

Endpoint Specification (MVP)

Endpoint	Method	Protocol	Auth	Purpose
`/api/v1/events`	POST	HTTPS	Entra ID bearer token (Event Grid)	Incoming call notifications + subscription validation
`/api/v1/callbacks/{callId}`	POST	HTTPS	Signed JWT (ACS)	Mid-call event callbacks (call connected, disconnected, etc.)
`/ws/v1`	GET → WSS upgrade	WSS	JWT in handshake headers	Bidirectional audio stream (ACS ↔ Service)
`:8081/actuator/health/liveness`	GET	HTTP (internal only)	None (port-isolated)	Container liveness probe — Spring context alive
`:8081/actuator/health/readiness`	GET	HTTP (internal only)	None (port-isolated)	Traffic routing probe — checks **cached** credential status (not live refresh) + Voice Live DNS resolution
`:8081/actuator/health/startup`	GET	HTTP (internal only)	None (port-isolated)	Startup probe — validates first managed identity token acquisition (runs once)

Health probe design (Party Mode — Winston): Readiness must NOT trigger a live `DefaultAzureCredential.getToken()` call. `DefaultAzureCredential` caches tokens internally; if the cache expires during an Azure AD hiccup, a live-refresh readiness check would flap, removing pods from the LB exactly when capacity is most needed. Instead: readiness checks a `CredentialHealthIndicator` that reports the cached token's status. The **startup** probe validates the first token acquisition. The credential health check must be behind a testable abstraction (`CredentialHealthChecker` interface) so the test profile can substitute a fake without touching `DefaultAzureCredential`.

Not in MVP: Admin endpoints (active call list, force-disconnect, metrics query). Deferred to P1 if operational need arises.

Architecture note: The service is an **Audio Protocol Translator**, not a transparent bridge. It performs binary ↔ JSON + base64 translation at every audio hop. This distinction drives error handling requirements (encoding failures, partial frames, malformed payloads).

Versioning note: `/ws/v1` is a **deliberate change** from the .NET reference repo which uses `/ws`. This supports blue-green deployment safety for in-flight calls (see API Versioning section). The path is an internal contract — ACS discovers it via `MediaStreamingOptions` set when answering the call.

Authentication Model

Already fully specified in Security Requirements (SEC-REQ-1 through SEC-REQ-17). Summary by endpoint:

Endpoint	Auth Mechanism	Validation
`/api/v1/events`	Entra ID bearer token	SEC-REQ-2: Validate issuer, audience, signature
`/api/v1/callbacks/{callId}`	ACS-signed JWT in headers	SEC-REQ-3: Validate signature; SEC-REQ-4: `{callId}` is non-guessable UUID (defense-in-depth — acts as shared secret between ACS and service)
`/ws/v1`	JWT in auth header on upgrade	SEC-REQ-1: Validate before upgrade; SEC-REQ-15: Rate limit connections
Outbound → Voice Live	`DefaultAzureCredential`	SEC-REQ-11: Managed identity, RBAC `Cognitive Services User`
Outbound → APIM/backends	Managed identity + OAuth2 bearer	SEC-REQ-11

Data Schemas

Data Flow	Format	Schema
Event Grid → `/api/v1/events`	JSON	`EventGridEvent[]` — **batched array**; each event processed concurrently. **Critical:** return HTTP 200 even if individual events fail — failed events logged, individual calls not answered. Never let one bad event poison the batch (Event Grid retries the *entire* batch on 5xx)

Data Flow	Format	Schema
Event Grid subscription validation	JSON	`SubscriptionValidationEventData` → respond with `validationResponse`
ACS → `/api/v1/callbacks/{callId}`	JSON	`CloudEvent[]` — `CallConnected`, `CallDisconnected`, `PlayFailed`, etc.
ACS ↔ `/ws/v1` (inbound audio)	Binary	PCM 24K mono, parsed via `StreamingData.parse()` → `AudioData` → extract PCM bytes → base64 encode → wrap in JSON for Voice Live
ACS ↔ `/ws/v1` (outbound audio)	Binary	Voice Live JSON `response.audio.delta` → base64 decode → PCM bytes → `OutStreamingData.getStreamingDataForOutbound()` → binary frame
ACS ↔ `/ws/v1` (control)	JSON	`StopAudio` for barge-in via `OutStreamingData.getStopAudioForOutbound()`
Service → Voice Live (audio)	JSON	`input_audio_buffer.append` with base64-encoded PCM
Voice Live → Service (audio)	JSON	`response.audio.delta` with base64-encoded PCM
Voice Live → Service (control)	JSON	`session.created`, `speech_started`, `response.done`, `tool_call`, etc.

Parsing rules:

- All Voice Live event parsing must use **lenient deserialization** (`DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES = false`) — Azure may add fields within a version
- Parse error counter required — spikes indicate schema drift
- Base64 encode/decode errors and partial/malformed audio frames must be handled explicitly (not silently dropped)

Error Codes & Responses

Scenario	HTTP Status	Response Body	Action
Invalid Event Grid token	401	`{"error": "unauthorized"}`	Reject, log SEC event
Invalid callback JWT	401	`{"error": "unauthorized"}`	Reject, log SEC event
Replayed callback (dup `jti` or expired `iat`)	401	`{"error": "unauthorized"}`	Reject; cache processed `jti` claims for JWT lifetime + 60s buffer
Unknown `callId` in callback	404	`{"error": "not_found"}`	Log warning
Malformed Event Grid payload	400	`{"error": "bad_request"}`	Reject, log error
WS upgrade without valid JWT	403	Connection rejected	SEC-REQ-1
WS rate limit exceeded	429	Connection rejected	SEC-REQ-15
WS idle timeout (no audio frames received)	—	Connection terminated	`C-54` timeout exceeded
Voice Live session init timeout	—	Graceful disconnect; caller hears apology prompt if possible	`C-51` timeout exceeded
Base64 decode failure (Voice Live → ACS)	—	Log error, skip frame, increment error counter	Lossy — single frame lost, not fatal
Partial/malformed audio frame (ACS → Service)	—	Log warning, skip frame	Graceful degradation
Service at max concurrent calls	503	`{"error": "service_unavailable"}`	Caller hears busy/retry
Internal error	500	`{"error": "internal_error"}`	No details leaked (SEC-REQ-10)

Rate Limiting Strategy

Layer	Mechanism	Rationale
WebSocket connections	SEC-REQ-15: Connection rate limiting per source IP	Prevent WS flood
HTTP callbacks (`/api/v1/callbacks`)	**Trust ACS** + Caffeine cache deduplication (key: `callId` + `eventType` + `correlationId`, `expireAfterWrite` TTL)	ACS controls callback frequency; dedup catches retries/bugs without full rate limiting. Use Caffeine (already in Spring Boot starter) — not a custom data structure

Layer	Mechanism	Rationale
Event Grid (`/api/v1/events`)	**Trust Event Grid** — built-in retry with exponential backoff	Event Grid self-throttles; 429 would cause unnecessary retries
Concurrent call admission	Config param `C-35` (`ivr.call.max-concurrent`)	Reject new calls with 503 when at capacity

Resilience & Lifecycle

Concern	Mechanism	Config
Voice Live session init timeout	If `session.created` not received within timeout → disconnect caller gracefully, log `VL_SESSION_TIMEOUT`	`C-51: ivr.voicelive.session-init-timeout-ms` (default 5000)
Orphan session reaper	Periodic sweep terminates sessions older than max-call-duration + buffer. Catches calls where neither WS close nor callback arrived	`C-52: ivr.call.orphan-reaper-interval-ms` (default 60000; test: 1000). 60s is proportional to typical call durations — 30s was too aggressive (60 sweeps per 30-min call)
Dual cleanup path	Call state cleaned up on *either* WS close *or* callback, whichever arrives first. Cleanup is idempotent — safe to trigger from both	—
Idle WebSocket detection	If no audio frames received within timeout after WS upgrade → terminate connection. Catches malicious/broken connections	`C-54: ivr.ws.idle-timeout-ms` (default 10000)
Lenient JSON parsing	Jackson `FAIL_ON_UNKNOWN_PROPERTIES = false` for all Voice Live event deserialization	—
Parse error monitoring	Counter metric for JSON parse failures on Voice Live events — spikes indicate schema drift	Metric: `ivr.voicelive.parse_errors_total`

API Versioning

Decision	Value	Rationale
Strategy	URL prefix: `/api/v1/` for HTTP, `/ws/v1` for WebSocket	Blue-green deployment safety — in-flight calls on old version survive while new calls route to new version
Scope	Both HTTP endpoints and WebSocket path	WS path versioned because audio framing/schema changes during deployment would break in-flight calls
Breaking changes	Increment to `/api/v2/`, `/ws/v2`	Standard practice; V1 stays running during migration
Current version	`v1`	Only version — no consumers to migrate

SDK & Client Libraries

No SDK required. This service is an internal audio protocol translator with no external API consumers:

- **Inbound callers:** ACS (Azure-managed, configured via ACS portal/SDK)
- **Outbound integration:** Voice Live API (Azure-managed, connected via WebSocket)
- **Operations:** Standard monitoring tools (Azure Monitor, Prometheus) consume metrics endpoints

API Documentation (MVP)

Artifact	Scope	Timeline
OpenAPI 3.0 spec	`/api/v1/events`, `/api/v1/callbacks/{callId}`	Day 1 (scaffold)
WebSocket protocol doc	`/ws/v1` message formats, lifecycle, audio translation pipeline	Day 2 (with audio protocol translator)
Deployment verification	Post-deploy smoke test validates Event Grid subscription URL matches service endpoint; zero-traffic canary alert (`ivr.calls.incoming.total` = 0 for 5 min → alert)	Day 5

Artifact	Scope	Timeline
README quickstart	Clone → configure → run → test call	Day 5
Postman/Bruno collection	Manual testing of HTTP endpoints	P1

Config note: C-51 through C-54 are defined in the main Configuration Catalog (under "API Backend Lifecycle"). Not duplicated here — see that section for full details, validation ranges, and priority tags.

Scoping & MVP Strategy

> **Note:** The Day-by-Day Sprint Plan in this section is the **authoritative version** and supersedes any day-by-day plans in prior sections (Steps 3 and 7). Those earlier plans were working drafts refined through scoping.

MVP Philosophy: Problem-Solving MVP

This MVP targets a **specific, measurable caller pain point**: callers who currently endure hold queues or rigid DTMF menus will get natural-language voice interaction within 30 seconds of dialing. The narrowest slice that proves the concept:

- One phone number → one ACS resource → one Voice Live session → one caller conversation.
- Three tool calls (GET user data, send invoice to registered email, send invoice to user-provided email) routed through APIM to backend APIs.
- No multi-language support.
- The AI agent **listens, understands, and responds** using its system prompt knowledge base, and **executes actions** via tool calls when the conversation requires it.

Team: 1 developer + GitHub Copilot. Azure resources (ACS, Foundry, AKS, Key Vault, APIM) are pre-provisioned. System prompt authored by domain specialists — consumed as config, not coded.

Scope boundary principle: If a capability doesn't directly contribute to a caller hearing an AI voice response within 30 seconds or executing one of the 3 defined tool calls, it's not P0.

P0 — Must-Have (5-Day Sprint)

#	Capability	Day	Trace
1	Spring Boot 4.x scaffold + health probes (startup, liveness, readiness)	1	AP-1
2	Event Grid webhook: receive `IncomingCall`, validate JWT, answer call	1	SEC-REQ-1, SEC-REQ-2
3	ACS mid-call callback handler (CallConnected, CallDisconnected, PlayFailed)	1	REQ-R1.1
4	JSR 356 `@ServerEndpoint` at `/ws/v1` — accept ACS audio stream (PCM 24K mono)	2	AP-2
5	Voice Live WebSocket client (`java.net.http.WebSocket`) — connect to `wss://<foundry>/voice-agent/realtime`	2	AP-2
6	Audio bridge: bidirectional PCM forwarding between ACS ↔ Voice Live	2	AP-3
7	`DefaultAzureCredential` for Voice Live auth (managed identity)	2	SEC-REQ-11
8	Linked lifecycle: closure of either WS triggers teardown of both within 3s	2	REQ-R1.1
9	Heartbeat/keepalive on both WebSocket connections (detect staleness ≤3s)	2	REQ-R1.2
10	Circuit breaker on Voice Live connections (Resilience4j or manual)	2–3	REQ-R2.1–R2.4
11	OpenTelemetry SDK setup + 4 spans (`ivr.call`, `ivr.call.answer`, `ivr.call.voicelive.connect`, `ivr.call.audio.bridge`)	2	OBS
12	Barge-in support: forward `input_audio_buffer.speech_started` / `speech_stopped`	3	UX-REQ-3
13	Reconnection window: 5s window to resume VL session on ACS reconnect	3*	REQ-R1.3–R1.4
14	VAD tuning per environment (silence threshold, end-of-speech delay as config)	3	UX-REQ-6, C-40–C-42
15	Graceful max-duration AI wrap-up (30s before timeout, AI delivers closing message)	3	UX-REQ-9, C-23
16	First real end-to-end call (ACS phone → service → Voice Live → AI voice response)	3	AC-1
17	`CallAutomationClient` with `DefaultAzureCredential` (no connection strings)	4	SEC-REQ-3
18	JWT validation on all inbound webhooks and WebSocket connections	4	SEC-REQ-1

#	Capability	Day	Trace
19	Event Grid `SubscriptionValidationEvent` handshake	4	SEC-REQ-2
20	WSS-only enforcement (never `ws://`)	4	SEC-REQ-5
21	Callback deduplication via Caffeine cache (TTL = C-09)	4	REQ-R3.3
22	`CredentialHealthChecker` abstraction for readiness probe	4	AP-6
23	Three-tier health: startup probe (first token), liveness (Spring context), readiness (cached credential + DNS)	4	AP-1
24	Structured JSON logging with correlation ID propagation	4	OBS
25	Unit + integration tests (≥80% line coverage on bridge + lifecycle code)	5	DoD (see Definition of Done section)
26	AKS deployment manifests + smoke test	5	DoD (see Definition of Done section)
27	Tool call handler: receive `tool_call` from Voice Live, dispatch HTTPS to APIM, return `tool_call_output`	2–3	AP-3, FR48–FR51
28	`DefaultAzureCredential` for APIM gateway auth (OAuth2 bearer token, scope C-57b)	2	SEC-REQ-18, C-57b
29	Tool definitions file loaded at session init (C-60) — externalized, not hardcoded	2	C-60, FR13
30	WireMock gateway stub in test profile + tool call integration tests	3–5	AC-23, AC-24

** Item 13 (reconnection window): Day 3 stretch goal / Day 4 fallback. First real call milestone (item 16) does not depend on it.*

Key Vault & Secrets Strategy

Approach: Kubernetes CRDs — **CSI SecretProviderClass** (preferred) or ExternalSecret Operator (alternative).

The application reads standard environment variables (`AZURE_CLIENT_ID`, `ACS_ENDPOINT`, `VOICE_LIVE_ENDPOINT`, etc.). Kubernetes infrastructure handles the Key Vault → environment variable injection via CRD manifests in the deployment layer.

Zero code impact: No Spring Cloud Azure Key Vault dependency. No `@Value("${azure.keyvault...}")` annotations. The app is unaware that secrets originate from Key Vault — it just reads env vars. This keeps the Spring Boot service portable and testable with plain env vars locally.

Deployment manifests (Day 5) will include:

- `SecretProviderClass` resource referencing the Key Vault
- Pod `volumeMounts` for CSI driver
- `env` / `envFrom` mappings to projected secrets

Explicitly Excluded from MVP (P1/P2)

- Extended tool calls beyond initial 3 (additional Oracle BRM / CRM operations)
- Voice Live tool call configuration (tool definitions are managed in Voice Live system prompt, not bridge code)
- Backend API service implementation behind APIM
- Multi-language support (pt-BR only in MVP)
- Call analytics / CDR pipeline
- KEDA autoscaling
- PII masking / redaction
- A/B testing of system prompts
- Prompt guardrails / content filtering

Explicitly Excluded from All Phases

- LGPD compliance module (handled by separate compliance team)
- Container image scanning (handled by platform team CI/CD)
- Call transfer to human agent (out of product scope)
- Abuse monitoring / rate limiting on caller side (ACS-level concern)
- Fixture recorder for call replay (no test playback system planned)

Parking Lot

Items removed from scope with defined reassessment triggers:

Item	Removed Because	Reassessment Trigger

Item	Removed Because	Reassessment Trigger
LGPD compliance module	Owned by separate compliance team	If compliance team cannot deliver before GA
Container scanning	Platform team CI/CD responsibility	If platform team pipeline not ready by Phase 2
Call transfer	Out of product scope	If business requires warm transfer for escalation
Abuse monitoring	ACS-level concern, not app-level	If ACS native rate limiting proves insufficient
Fixture recorder	No test playback system planned	If regression testing requires deterministic audio replay
PII masking	No PII flows through audio bridge in MVP (tool call responses are logged at INFO level with correlationId only, not response payloads)	If tool call responses introduce structured PII in logs/traces

Phase 2 — Growth (Post-MVP)

#	Capability	Depends On
1	Extended tool calls beyond initial 3 (additional Oracle BRM, CRM operations via APIM)	P0 tool call framework stable
2	Validate Voice Live API key auth is disabled in production config (FR12 enforces managed identity-only in P0; this item confirms no residual API key fallback)	P0 credential infra
3	Prompt guardrails / content filtering	Tool call handler
4	Nightly E2E test suite (synthetic calls)	P0 deployment
5	A/B testing of system prompts (config-driven)	P0 system prompt as config
6	Call detail records (CDR) to Azure Monitor / Log Analytics	P0 structured logging
7	Graceful drain on pod termination (in-flight calls complete)	P0 lifecycle management
8	Connection pool for Voice Live WebSocket (pre-warm N sessions)	P0 circuit breaker
9	Advanced prompt engineering: multi-turn conversation memory	Tool call handler

Phase 3 — Scale

#	Capability	Depends On
1	KEDA autoscaling based on active WebSocket connection count	Phase 2 metrics
2	Multi-language support (es, en beyond pt-BR)	Phase 2 prompt framework
3	Call analytics dashboard (Power BI / Grafana)	Phase 2 CDR pipeline
4	Multi-client/tenant support (different system prompts per tenant)	Phase 2 config framework
5	Dead-letter queue for failed tool calls	Phase 2 tool call handler

Risk Mitigation Strategy

Technical Risks

Risk	Probability	Impact	Mitigation
Voice Live API latency spikes >500ms	Medium	High	Circuit breaker (P0 item 10); fallback: play hold tone while reconnecting
ACS audio format mismatch with Voice Live	Low	Critical	Both use PCM 24K mono — validated in .NET reference. Day 2 integration test confirms.
Voice Live WebSocket drops under load	Medium	High	Heartbeat (P0 item 9) + circuit breaker + reconnection window

Risk	Probability	Impact	Mitigation
`DefaultAzureCredential` token refresh race condition	Low	Medium	`CredentialHealthChecker` (P0 item 22) caches token; readiness probe gates traffic
Spring Boot 4.x + Virtual Threads stability	Low	Medium	Well-tested in Spring 6.2+; fallback: platform threads with bounded pool
Max-duration wrap-up: `conversation.item.create` may not work mid-session	Medium	Medium	**Day 2 spike:** verify Voice Live API supports injecting system message mid-session. Fallback: server-side timer disconnects without AI wrap-up message.

Market Risks

Risk	Probability	Impact	Mitigation
Voice Live API preview breaking changes	Medium	High	Pin `api-version=2025-05-01-preview`; monitor Azure changelog weekly
ACS Call Automation SDK breaking changes	Low	Medium	Pin SDK version 1.6.0; test before upgrading

Resource Risks

Risk	Probability	Impact	Mitigation
Single developer unavailable (illness, etc.)	Low	Critical	All code in Git; comprehensive README; Copilot instructions enable any Java dev to continue
Azure resource quota limits	Low	Medium	Pre-provisioned; quota checked before sprint
Voice Live API rate limits hit during testing	Medium	Low	Use single-call tests; no load testing in MVP

Authoritative Day-by-Day Sprint Plan

Day 1 — Scaffold & Ingress (8h)

Hours	Task	Produces
0–2	Spring Boot 4.x project init: Maven, Java 21, Virtual Threads, health actuator	Running app with `/actuator/health`
2–4	Event Grid webhook controller: `IncomingCall` event handling, `SubscriptionValidationEvent` handshake	POST `/api/v1/events` accepting Event Grid
4–6	`CallAutomationClient` bean + answer call logic + callback handler (`CallConnected`, `CallDisconnected`, `PlayFailed`)	Call answered, callbacks logged
6–8	OpenAPI 3.0 spec for HTTP endpoints + config externalization (`application.yaml` with all C-* params)	Documented API, configurable app

Day 2 — Audio Bridge, Resilience & Tool Call Framework (8h)

Hours	Task	Produces
0–1	**Spike:** Verify Voice Live API supports `conversation.item.create` mid-session for max-duration wrap-up. **No-go impact:** FR6 and FR16 revert to hard disconnect at max-duration (no AI wrap-up message); FR6 acceptance criteria adjusted to require only hard disconnect + log event.	Go/no-go on UX-REQ-9 approach
1–3	JSR 356 `@ServerEndpoint` at `/ws/v1` — accept ACS audio stream, parse `AudioData` via `StreamingData.parse()`	ACS audio flowing into service
3–5	Voice Live WebSocket client (`java.net.http.WebSocket`) — connect, authenticate with `DefaultAzureCredential`, send/receive events	Service connected to Voice Live
5–6	Audio bridge: bidirectional PCM forwarding + linked lifecycle (either-side close → both teardown ≤3s)	End-to-end audio path
6–7	Heartbeat/keepalive on both WS connections + circuit breaker on Voice Live	Resilient connections

Hours	Task	Produces
7–8	Tool call handler: receive `tool_call` from Voice Live → dispatch HTTPS to APIM via `java.net.http.HttpClient` + `DefaultAzureCredential` → return `tool_call_output`. Tool definitions file loading (C-60) + `session.update` tool registration.	Tool call dispatcher + definitions operational

Day 3 — Tool Call Hardening, Conversation Quality & First Call (8h)

Hours	Task	Produces
0–1	Tool call timeout (C-58) + error handling + max concurrent guard (C-59) + WireMock gateway stubs in test profile	Robust tool call error path
1–2	OpenTelemetry SDK setup + 4 call lifecycle spans + tool call spans	Distributed tracing operational
2–4	Barge-in: forward `input_audio_buffer.speech_started` / `speech_stopped` from Voice Live to ACS	Natural conversation flow
4–5	VAD tuning: expose silence threshold + end-of-speech delay as config (C-40–C-42)	Environment-specific voice detection
5–6	Max-duration wrap-up: implement graceful AI closing message 30s before timeout (based on Day 2 spike result)	Graceful call endings
6–8	**First real end-to-end call**: ACS phone number → Event Grid → service → Voice Live → AI voice response heard by caller. Tool call round-trip proven (user data retrieval via APIM gateway stub).	🎯 MVP proof-of-life + tool call proven

Day 4 — Security & Hardening (8h)

Hours	Task	Produces
0–2	JWT validation on all inbound webhooks + WebSocket upgrade requests	Authenticated ingress
2–3	`CallAutomationClient` switched to `DefaultAzureCredential` (remove any connection string usage)	Zero secrets in code
3–4	WSS-only enforcement + callback deduplication (Caffeine cache)	Secure + idempotent
4–5	`CredentialHealthChecker` + three-tier health probes (startup/liveness/readiness)	Production-grade health
5–6	Structured JSON logging + correlation ID propagation across both WS connections	Observable calls
6–8	Reconnection window: 5s window to resume VL session on ACS reconnect (FR20/FR21)	REQ-R1.3–R1.4 complete

Day 5 — Tests & Deployment (8h)

Hours	Task	Produces
0–3	Unit tests: audio bridge, lifecycle management, circuit breaker, credential checker, tool call dispatcher/timeout	≥80% coverage on critical paths
3–5	Integration tests: Event Grid → answer → audio bridge → Voice Live (mocked WS) + tool call round-trip (WireMock gateway stub)	End-to-end flow verified
5–6	AKS deployment manifests: Deployment, Service, Ingress, CSI SecretProviderClass CRD	K8s-ready artifacts
6–7	Deploy to AKS + smoke test (real call through deployed service)	Production deployment verified
7–8	README quickstart + WebSocket protocol doc	Developer onboarding docs

Functional Requirements

Synthesized from all discovery sections (Steps 2–8). 50 requirements across 10 capability areas. Every P0 scoping item, acceptance criterion, and security requirement has at least one corresponding FR. Enhanced via Comparative Analysis Matrix elicitation (13 gaps identified and patched), User Persona Focus Group elicitation (2 new FRs, 2 FR enhancements, 1 preamble addition), Critique & Refine elicitation (3 critical fixes, 4 moderate fixes, 2 low refinements; FR15 absorbed into FR13+FR31), and Edit Step E-3 (tool call framework promotion: FR48–FR51 added, FR13 updated with tool definitions).

Self-Validation

Check	Result
All 30 P0 scoping items traced	☑ Each P0 item maps to ≥1 FR
All 5 user journeys covered	☑ Maria (FR1–FR14, FR16, FR46, FR48–FR51), Carlos (FR17–FR21, FR33–FR35, FR47), Renata (FR28–FR32, FR41), Thiago (FR13), Lucas (FR28–FR32)
Implementation-agnostic	☑ No class names, framework APIs, or library references in FR text (CR-4 fixed FR23/FR24 violations)
Each FR independently testable	☑ Each FR has a verifiable behavior with observable inputs/outputs
Defensive behaviors covered	☑ FR33–FR47 cover admission, SIGTERM, fallback prompts, replay, rate limit, orphan reaper, idle detection, mid-stream stall, PlayFailed terminal

Profile Qualification: Functional requirements describe the system's behavior under the `prod` Spring profile. Security-enforcing FRs (FR22 JWT validation, FR26 WSS-only, FR38 rate limiting) are governed by configuration toggles (C-40, C-46, etc.) that may be relaxed in `test` and `local` profiles to enable cloud-free testing and local development without certificate infrastructure. The `prod` profile SHALL enforce all security FRs unconditionally — no toggle may disable a security FR in production. The Profile Override Matrix in the Configuration Catalog documents the exact per-profile values.

Capability 1: Call Ingestion & Lifecycle Management

ID	Functional Requirement	Trace
FR1	The system SHALL receive `IncomingCall` events from Azure Event Grid via HTTPS webhook and initiate the call-answer sequence within the configured ring timeout (C-03).	T-01, P0 #2
FR2	The system SHALL answer incoming calls using ACS Call Automation with bidirectional media streaming enabled (PCM 24K mono).	T-03, P0 #4
FR3	The system SHALL handle ACS mid-call callback events (`CallConnected`, `CallDisconnected`, `PlayFailed`, `PlayCompleted`) and update internal call state accordingly.	T-06, P0 #3
FR4	The system SHALL perform graceful teardown of all resources (both WebSocket connections, call state, metrics) when a call ends, regardless of which side initiates disconnection.	T-19, T-20, T-30
FR5	The system SHALL deduplicate ACS callback events using a cache keyed on `callId` + `eventType` + `correlationId` with a configurable TTL (C-56), ensuring idempotent processing.	API Backend Rate Limiting
FR6	When the call approaches max duration (C-29 minus C-30 seconds remaining), the system SHALL inject a context message to Voice Live prompting the AI to wrap up the conversation naturally, and enforce a hard disconnect at max duration if the call does not end voluntarily.	UX-REQ-9, T-21, T-29

Capability 2: Real-Time Audio Communication

ID	Functional Requirement	Trace
FR7	The system SHALL accept inbound WebSocket connections from ACS at the versioned path `/ws/v1` and parse incoming binary frames into `AudioMetadata` and `AudioData` packets.	T-06, P0 #4
FR8	The system SHALL send outbound audio to ACS via the same WebSocket connection using the ACS SDK's outbound streaming data format.	T-17, P0 #6
FR9	The system SHALL forward audio bidirectionally between ACS and Voice Live in real-time without buffering, inspecting, or transforming the audio content (protocol translation only).	AP-1, P0 #6
FR10	The system SHALL enforce a linked lifecycle between the ACS and Voice Live WebSocket connections — closure of either side SHALL trigger graceful teardown of both within the configured timeout (C-31).	REQ-R1.1, T-19, T-20

Capability 3: Conversational AI Integration

ID	Functional Requirement	Trace
FR11	The system SHALL establish an outbound WebSocket connection to the Voice Live API endpoint (C-07) with the configured API version (C-08) within the connect timeout (C-10).	T-08, T-11, P0 #5

ID	Functional Requirement	Trace
FR12	The system SHALL authenticate to the Voice Live API using managed identity credentials, with no API keys in code or configuration.	SEC-REQ-11, P0 #7
FR13	The system SHALL send a `session.update` event to Voice Live upon connection, configuring the session with the externalized system prompt (C-12), voice model (C-13), tool definitions loaded from the externalized definitions file (C-60), and turn detection settings — including VAD type, threshold, silence duration, noise suppression, and echo cancellation (C-14 through C-18). If Voice Live rejects the session (error response or no `session.created` within C-51), the system SHALL log the rejection at WARN level including the `correlationId`, the Voice Live error code and message (if provided), and the system prompt size in characters — enabling operators and domain specialists to distinguish prompt-related rejections (e.g., prompt too large, invalid voice model ID) from infrastructure failures (e.g., network timeout, auth failure). This diagnostic detail is critical because the system prompt is the primary per-client customization point, and prompt authors need actionable feedback when their content causes session failures.	T-13, P0 #14, P0 #29, FG-4
FR14	Upon receiving `input_audio_buffer.speech_started` from Voice Live, the system SHALL immediately (i) stop forwarding Voice Live audio frames to ACS and (ii) cancel any in-progress ACS Play operation, treating both actions as idempotent operations safe to invoke redundantly.	UX-REQ-5, UX-REQ-8, T-18, T-25, CR-1
FR16	The system SHALL support injecting system-level messages to Voice Live mid-session (e.g., max-duration wrap-up notification) when triggered by lifecycle events.	UX-REQ-9, T-21, P0 #15

Capability 4: Connection Resilience & Recovery

ID	Functional Requirement	Trace
FR17	The system SHALL implement heartbeat/keepalive on both WebSocket connections and detect connection staleness within the configured idle timeout (C-05, C-11). Upon detecting staleness, the system SHALL force-close the stale WebSocket connection, triggering the linked lifecycle teardown per FR10.	REQ-R1.2, P0 #9, CR-2
FR18	The system SHALL implement a circuit breaker on Voice Live connections that trips to OPEN state after N consecutive connection failures (C-22), preventing new calls from attempting doomed connections.	REQ-R2.1, P0 #10
FR19	The circuit breaker SHALL transition through CLOSED → OPEN → HALF-OPEN → CLOSED states, with configurable half-open delay (C-23) and success threshold for recovery (C-24).	REQ-R2.1, T-08, T-09
FR20	If ACS reconnects within the configured reconnection window (C-28), the system SHALL attempt to resume the existing Voice Live session rather than starting a new one. If the Voice Live protocol does not support session resumption, or the resume attempt fails, the system SHALL fall back to FR21 behavior (play fallback prompt and disconnect gracefully).	REQ-R1.3, P0 #13, CR-8
FR21	When the circuit breaker is OPEN or session resumption fails, the system SHALL play the appropriate fallback audio prompt to the caller before disconnecting gracefully.	REQ-R2.2, REQ-R1.4, T-09, T-12

ID	Functional Requirement	Trace
FR46	The system SHALL detect when a Voice Live audio response stalls mid-stream — defined as receiving one or more `response.audio.delta` events followed by no further audio events and no `response.done` within a configurable timeout (C-55, default 5000ms). This is distinct from the first-audio timeout (FR35b, C-32) which covers pre-conversation silence. On stall detection, the system SHALL log a WARN with the `correlationId`, elapsed silence duration, and last event type received, then play the apology prompt (C-37) and initiate graceful teardown. This prevents callers from sitting in unexplained mid-conversation silence when Voice Live freezes without closing the connection.	UX-REQ-3, REQ-R1.1, FG-1

Capability 5: Security & Authentication

ID	Functional Requirement	Trace
FR22	The system SHALL validate JWT tokens in the `Authorization` header during WebSocket upgrade handshake, rejecting connections with HTTP 401 before upgrading if the token is invalid, expired, or missing.	SEC-REQ-1, AC-15, P0 #18
FR23	The system SHALL authenticate to ACS Call Automation using managed identity credentials, with no connection strings or API keys in code, configuration, or secret stores.	SEC-REQ-11, AC-22, P0 #17, CR-4
FR24	The system SHALL authenticate to Voice Live API using managed identity credentials. *Infrastructure prerequisite: RBAC role `Cognitive Services User` on the Foundry resource.*	SEC-REQ-11, P0 #7, CR-4
FR25	The system SHALL validate Entra ID bearer tokens on Event Grid webhook requests and complete the `SubscriptionValidationEvent` handshake during subscription registration.	SEC-REQ-2, P0 #19
FR26	All WebSocket connections (both ACS-facing and Voice Live-facing) SHALL use `wss://` protocol exclusively; the system SHALL reject or never initiate `ws://` connections.	P0 #20
FR27	The system SHALL implement a testable credential health abstraction that reports cached token status without triggering live token refresh, used by the readiness probe.	P0 #22

Capability 6: Operational Observability

ID	Functional Requirement	Trace
FR28	The system SHALL emit all log events in structured JSON format, with every entry including `correlationId` (ACS call ID for domain-level correlation), `timestamp`, `level`, and `component` fields. When an OpenTelemetry trace context is active, each log entry SHALL additionally include `traceId` and `spanId` fields, enabling direct correlation between distributed tracing spans (FR29) and log lines in observability tooling. This dual-ID approach supports two complementary query paths: searching logs by `correlationId` for a business-level call view, and searching by `traceId` for an infrastructure-level request flow view.	P0 #24, Observability, FG-3
FR29	The system SHALL emit distributed tracing spans for the 4 defined call lifecycle operations (`ivr.call`, `ivr.call.answer`, `ivr.call.voicelive.connect`, `ivr.call.audio.bridge`).	P0 #11, Observability
FR30	The system SHALL expose three-tier health probes: startup (first managed identity token acquisition), liveness (Spring context alive), readiness (cached credential status + Voice Live DNS resolution).	P0 #23, API Backend
FR31	All configurable parameters (per the Configuration Catalog) SHALL be externalized via Spring configuration properties with validation constraints, supporting per-environment overrides via Spring profiles (`test`, `local`, `prod`).	P0 #24, Config Catalog, CR-5
FR32	The system SHALL expose Micrometer metrics for all 14 defined metric names (prefixed `ivr.*`) covering active calls, latency distributions, circuit breaker state, security rejection counts, and audio forwarding counters.	Observability Metrics

Capability 7: Call Admission & Lifecycle Edge Cases

ID	Functional Requirement	Trace
FR33	The system SHALL track instance-local active concurrent calls and reject incoming calls with a spoken busy prompt (C-38) when the instance-local active call count exceeds the per-instance admission threshold (C-26).	REQ-R4.1, UX-REQ-11, T-02, CR-3

ID	Functional Requirement	Trace
FR34	On SIGTERM, the system SHALL mark readiness probe unhealthy immediately, send a wrap-up notification to in-flight Voice Live sessions, and allow active calls to complete within drain timeout (C-34) before forcefully terminating connections.	REQ-R3.1, REQ-R3.2, T-23
FR35	The system SHALL play pre-recorded audio prompts via ACS Play action for the following lifecycle events: (a) comfort tone if VL connection exceeds C-33, (b) fallback greeting if no AI audio within C-32, (c) apology prompt on unexpected VL disconnect, (d) busy prompt on admission rejection, (e) service-unavailable prompt when circuit breaker is open.	UX-REQ-1, UX-REQ-3, UX-REQ-10, UX-REQ-11, REQ-R2.2
FR42	The system SHALL periodically sweep for call sessions exceeding max-duration plus buffer (C-52 interval) and forcefully terminate orphaned sessions where neither WebSocket close nor callback arrived.	C-52, API Backend
FR43	The system SHALL terminate WebSocket connections that receive no audio frames within the idle timeout (C-54) after successful upgrade.	C-54, API Backend
FR47	When an ACS Play action fails (`PlayFailed` callback received for any fallback prompt in FR35a–e), the system SHALL NOT attempt to play a substitute prompt or retry the failed prompt. Instead, it SHALL log the `PlayFailed` event at ERROR level with the `correlationId`, failed prompt identifier, and ACS failure reason, then proceed directly to graceful call termination (close both WebSocket connections and free resources). This prevents infinite-fallback loops where each failed prompt triggers another prompt attempt, and ensures the caller experiences a clean disconnect rather than prolonged silence. The ACS platform's own disconnect tone provides the caller's final audio cue.	UX-REQ-3, T-06, FG-2

Capability 8: Defense-in-Depth Security

ID	Functional Requirement	Trace
FR36	The system SHALL generate a cryptographically random token with a minimum length (C-50) and embed it in each callback URL registered with ACS, validating the token on every callback request.	SEC-REQ-4, S-4, CR-9
FR37	The system SHALL validate the `eventTime` of each Event Grid event and reject events older than the configured maximum age (C-45) to prevent replay attacks.	SEC-REQ-14, AC-16
FR38	The system SHALL enforce per-IP rate limiting on WebSocket handshake attempts, rejecting connections exceeding C-46 per C-47 window with HTTP 429.	SEC-REQ-15, AS-5
FR39	The system SHALL bind Spring Boot Actuator to a separate management port (C-48), exposing only health/liveness/readiness/startup endpoints, returning 404 for all other actuator paths.	SEC-REQ-9, AC-19
FR40	The system SHALL return generic JSON error bodies on all HTTP error responses, with no stack traces, class names, or framework-specific details.	SEC-REQ-10, AC-20

Capability 9: Operational Safety Nets

ID	Functional Requirement	Trace
FR41	The system SHALL periodically probe Voice Live API connectivity (C-25 interval) and preemptively trip the circuit breaker if probes fail.	REQ-R2.4
FR44	The system SHALL process Event Grid event batches individually, returning HTTP 200 for the batch even if individual events fail processing, logging failures per event. *Required test scenario: mixed batch (e.g., 3 events — 1 valid `IncomingCall`, 1 malformed, 1 valid) — verify the 2 valid events are processed and the malformed event is logged without poisoning the batch.*	API Backend Data Schemas
FR45	The system SHALL use lenient JSON deserialization for Voice Live events (ignoring unknown fields) and increment a parse error counter metric on deserialization failures.	API Backend Parsing Rules

Capability 10: Tool Call Framework

ID	Functional Requirement	Trace
FR48	Upon receiving a `tool_call` event from Voice Live, the system SHALL dispatch an HTTPS request to the configured API gateway base URL (C-57) using `java.net.http.HttpClient` on a virtual thread, authenticating with `DefaultAzureCredential` (scope C-57b). The request SHALL include the tool call name, arguments, and the call's `correlationId`.	AP-3, P0 #27, SEC-REQ-18
FR49	Upon receiving a successful HTTP response from the API gateway, the system SHALL forward the result to Voice Live as a `tool_call_output` event containing the tool call ID and the response payload.	P0 #28
FR50	If the API gateway does not respond within the configured timeout (C-58), or returns an HTTP error (4xx/5xx), the system SHALL return a `tool_call_output` event to Voice Live with an error payload indicating the failure reason, log the failure at WARN level with `correlationId`, tool name, HTTP status (if available), and elapsed time, and increment the `ivr.tool_call.errors` metric. The system SHALL NOT terminate the call due to a tool call failure — the AI conversation continues.	AC-24, P0 #30, C-58
FR51	At session initialization, the system SHALL load tool definitions from the externalized definitions file (C-60), validate the JSON structure, and include the tool definitions in the `session.update` event sent to Voice Live (FR13). If the definitions file is missing or contains invalid JSON, the system SHALL log at ERROR level and start the session without tool definitions (degraded mode).	P0 #29, C-60

Non-Functional Requirements

Synthesized from all discovery sections (Steps 2–8) and 50 Functional Requirements. 47 NFRs across 6 categories, enhanced via two elicitation rounds: Comparative Analysis Matrix (13 gaps identified → 13 NFRs added) and Constraint Mapping (8 tensions resolved → 8 NFR refinements).

Self-Validation

Check	Result
All Success Criteria measurable outcomes traced	☑ Each outcome maps to ≥1 NFR
All Resilience requirements (FM-1–FM-5) covered	☑ 5/5
All Architectural Principles (AP-1–AP-6) covered	☑ 6/6
STRIDE threats mitigated by NFRs	☑ 8/9 (S-3 PII masking excluded per scope)
Security Requirements (SEC-REQ) enforced	☑ 15/17 (SEC-REQ-7, SEC-REQ-8 are P1 LGPD)
No mathematical contradictions between NFRs	☑ Resolved via Constraint Mapping (CT-CRITICAL, CT-1–CT-7)
All NFRs independently measurable	☑ Each has a metric name or test assertion

Elicitation History

Round	Method	Findings	Result
1	Comparative Analysis Matrix	13 gaps (G-1–G-13) across 9 discovery categories	13 NFRs added (R8–R13, SEC8–SEC10, P9, I5–I7)
2	Constraint Mapping	1 critical contradiction, 3 high tensions, 4 moderate tensions	8 NFR refinements (R2, R12, P1, P6, S6, SEC10, P9, I5)

Category 1: Performance (10 NFRs)

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-P1	**Time-to-First-Audio** — elapsed time from `CallConnected` event to first audio byte sent to ACS. *Implementation note: JWKS cache MUST be pre-warmed during startup to eliminate cold-call JWT latency from the critical path.*	`ivr.time_to_first_audio` (Timer)	< 3s p95	Success Criteria, UX-REQ-4, AC-9

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-P2	**AI Response Latency** — elapsed time from end of caller speech (VAD end-of-speech) to first `response.audio.delta` from Voice Live.	`ivr.ai.response.latency` (Timer)	< 1s p95	UX-REQ-7, Journey 1
NFR-P3	**Barge-In Latency** — elapsed time from `input_audio_buffer.speech_started` event received from Voice Live to `StopAudio` command sent to ACS.	`ivr.bargein.latency` (Timer)	< 100ms p95	UX-REQ-5, UX-REQ-8, AC-3, Success Criteria
NFR-P4	**Audio Forwarding Latency** — elapsed time for a single audio frame to traverse the bridge (ACS → Service → Voice Live or reverse).	`ivr.audio.forward.latency` (Timer)	< 50ms p99	AP-3, FR9
NFR-P5	**Call Answer Latency** — elapsed time from `IncomingCall` Event Grid event received to `answerCall()` API response.	`ivr.call.answer.latency` (Timer)	< 3s p95	FR1, T-01, T-03
NFR-P6	**Voice Live Connect Latency** — elapsed time from WebSocket handshake initiation to `session.created` received. Tightened from original p99-only target to include p95 — necessary to preserve latency budget for NFR-P1 (time-to-first-audio) since VL connect is sequential in the call setup chain. *(Constraint Mapping CT-1)*	`ivr.voicelive.connect.latency` (Timer)	≤ 2s p95, ≤ 3s p99	FR11, T-08, T-13
NFR-P7	**Per-Call Memory** — maximum heap allocation attributable to a single active call's state (call context, buffers, WS frames).	Profiler / heap dump analysis	≤ 200 KB per call	FM-3, AP-4
NFR-P8	**Zero-Copy Audio Forwarding** — audio bytes SHALL NOT be copied to intermediate buffers, temporary storage, or duplicate data structures during the forwarding path. Protocol translation (binary ↔ base64) is permitted; content duplication is not.	Code review + allocation profiler	Zero unnecessary copies	AP-3, AP-4
NFR-P9	**Application Startup Time** — elapsed time from JVM launch to readiness probe returning HTTP 200. Assumes workload identity (fastest credential path). Credential acquisition capped at 15s — if token not acquired within 15s, startup fails rather than hanging. *(Constraint Mapping CT-4)*	`application.started.time` (Spring metric)	≤ 30s (cold JVM, no CDS)	P0 #23, FR30, API Backend Health Probes
NFR-P10	**Tool Call Round-Trip Latency** — elapsed time from receiving a `tool_call` event from Voice Live to sending the `tool_call_output` event back, including APIM gateway HTTP call, `DefaultAzureCredential` token acquisition, and response parsing.	`ivr.tool_call.latency` (Timer)	≤ 5s p95	FR48, FR49, AC-23, C-58

Category 2: Reliability (14 NFRs)

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-R1	**Service Availability** — percentage of time the service accepts and processes incoming calls (excludes planned maintenance).	Uptime monitoring (health probe success rate)	≥ 99.9%	Success Criteria
NFR-R2	**Linked Lifecycle Teardown** — elapsed time from one WebSocket closing to the peer WebSocket being fully closed + resources freed. Applies to **non-reconnectable closures** only: close codes `1000` (normal), `1001` (going away), or explicit caller/AI hangup. Reconnectable closures (code `1006`, transport errors) defer to NFR-R12's reconnection window. *(Constraint Mapping CT-CRITICAL)*	`ivr.call.teardown.latency` (Timer)	≤ 3s	REQ-R1.1, FR10, T-19, T-20
NFR-R3	**Circuit Breaker Activation** — when Voice Live connections fail consecutively (C-22), the circuit breaker SHALL trip within 1 call of threshold, preventing wasted connection attempts.	`ivr.circuit_breaker.state` (Gauge) + transition log	Trips at exactly C-22 failures	REQ-R2.1, FR18, FR19

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-R4	**Graceful Shutdown Drain** — on SIGTERM, all in-flight calls complete or reach drain timeout (C-34) before process exit. Zero abrupt disconnections during rolling deploys.	`ivr.shutdown.calls_drained` (Counter)	100% of in-flight calls drained	REQ-R3.1, FR34, T-23
NFR-R5	**Zero Race Conditions** — concurrent state transitions on the same call (e.g., simultaneous ACS close + VL close) SHALL NOT produce inconsistent state, resource leaks, or duplicate teardown actions.	Concurrent integration tests + ThreadSanitizer-style analysis	Zero failures under concurrent close scenarios	Concurrency Model, FM-2
NFR-R6	**Callback Deduplication** — duplicate ACS callback events (same `callId` + `eventType` + `correlationId`) SHALL be processed exactly once.	`ivr.callbacks.deduplicated` (Counter)	100% dedup accuracy	FR5, API Backend Rate Limiting
NFR-R7	**Zero Orphaned Sessions** — no Voice Live sessions SHALL remain open after the associated ACS call has ended + linked lifecycle timeout expired.	`ivr.sessions.orphaned` (Gauge) — should be 0	0 at any measurement point	FR42, C-52, API Backend Resilience
NFR-R8	**Call Completion Rate** — ratio of calls that are answered and reach a graceful end (caller hangup, AI-initiated close, or max-duration wrap-up) to total calls answered. Excludes calls rejected by admission control (FR33) which are not "answered."	`ivr.call.completion.rate` (computed: graceful_ends / total_answered)	≥ 99%	Success Criteria (Measurable Outcomes)
NFR-R9	**Heartbeat Detection Latency** — elapsed time from last successful heartbeat response to staleness detection event. Measures how quickly the system detects a dead WebSocket connection.	`ivr.ws.staleness.detection.latency` (Timer)	≤ 3s	FM-4, REQ-R1.2, FR17, P0 #9
NFR-R10	**Configuration Fail-Fast** — the application SHALL fail to start within 5s with a descriptive error message if any required configuration parameter is missing, null, or violates its declared validation range (as defined in the Configuration Catalog). Prevents misconfigured deployments from accepting traffic and failing unpredictably at first call.	Startup fails with descriptive error (integration test)	100% of invalid configs caught at boot	Config Catalog (C-01–C-56), AP-1
NFR-R11	**Observability Signal Completeness** — for each completed call lifecycle (IDLE → ... → CLOSED), all applicable log events (from the 23 defined) and all applicable metrics (from the 16 defined) SHALL be emitted. Missing signals are invisible by definition — this NFR makes their absence detectable.	Per-call audit in integration tests	≥ 99% signal completion rate	Observability Requirements, Journey 3 (Renata)
NFR-R12	**Reconnection Window Compliance** — when an ACS WebSocket closes with an abnormal close code (`1006`) or transport error, the system SHALL hold the Voice Live session open for the configured reconnection window (C-28, default 5s) before teardown. Normal closures (`1000`, `1001`) trigger immediate NFR-R2 teardown instead. *(Constraint Mapping CT-CRITICAL)*	`ivr.reconnection.window.used` (Timer)	Session held ≤ C-28 (5s); resume attempted if ACS reconnects	REQ-R1.3, FR20, P0 #13
NFR-R13	**Stall Detection Accuracy** — mid-stream stall detection (FR46, C-55) SHALL NOT prematurely terminate calls that are actually processing. Validated via post-hoc log analysis correlating stall terminations with Voice Live's actual response behavior. *Operational note: stall timeout (C-55) should be validated against observed Voice Live response latency distribution before tightening below 5s.*	False positive rate (stall terminations that were actually processing) *(production metric — not CI-gated; requires production log analysis)*	< 1% false positives	FR46, FG-1

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-R14	**Tool Call Timeout Handling** — when the API gateway does not respond within C-58, the system SHALL return an error `tool_call_output` to Voice Live and continue the call without interruption. The AI conversation SHALL degrade gracefully (no data fetched) rather than terminate.	`ivr.tool_call.errors` (Counter, tag: `timeout`) + call continuation verified in integration test	100% timeout-handling coverage; 0 call terminations from tool call failures	FR50, AC-24, C-58

Category 3: Scalability (6 NFRs)

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-S1	**Concurrent Call Capacity** — maximum simultaneous active calls per service instance without degradation of NFR-P1 through NFR-P4.	`ivr.calls.active` (Gauge) at load	50 calls per instance	Success Criteria, C-26, Load Targets
NFR-S2	**Sustained Load Stability** — under sustained load at NFR-S1 capacity for ≥30 minutes, zero OOM kills, zero thread starvation, zero connection pool exhaustion.	Load test: 50 calls × 30 min	Zero errors, stable latency percentiles	Load Targets
NFR-S3	**Instance Memory Limit** — total container memory usage (heap + off-heap + OS) at full capacity (50 concurrent calls).	Container metrics (`container_memory_working_set_bytes`)	≤ 1 GB	FM-3, Deployment (512MB–1GB per instance)
NFR-S4	**Stateless Horizontal Scaling** — zero shared mutable state between instances. Each instance manages its own calls independently. New instances accept traffic immediately after readiness probe passes.	Architecture validation + rolling deploy test	No inter-instance state required	AP-4, Concurrency Model
NFR-S5	**Admission Control Isolation** — when one instance reaches capacity and rejects calls (FR33), other instances continue accepting calls normally. Rejection is instance-local.	Multi-instance load test	Zero cross-instance impact	FR33, REQ-R4.1
NFR-S6	**Network Bandwidth** — total inbound + outbound network throughput per instance at full capacity. Accounts for base64 encoding overhead (~37% inflation), JSON framing, control messages, heartbeats, and metric/log egress. *(Constraint Mapping CT-3: raised from 5 MB/s — 50 calls × bidirectional PCM × base64 overhead = ~6.6 MB/s raw)*	Network interface metrics	≤ 10 MB/s per instance	Infrastructure sizing

Category 4: Security (10 NFRs)

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-SEC1	**WebSocket Authentication Coverage** — every WebSocket upgrade request to `/ws/v1` SHALL be authenticated via JWT validation before the connection upgrades. Zero unauthenticated connections.	`ivr.security.auth.rejected` (Counter by reason)	100% validation; 0 unauthenticated upgrades	SEC-REQ-1, FR22, AC-15
NFR-SEC2	**Zero Secrets in Code/Config** — no API keys, connection strings, passwords, or bearer tokens in source code, committed config files, or container images. Secrets resolved at runtime via managed identity or K8s CSI SecretProviderClass.	Static analysis (grep + secret scanner in CI)	Zero findings	SEC-REQ-11, AC-22, FR23, FR24
NFR-SEC3	**WSS-Only Enforcement** — all WebSocket connections (both ACS-facing server and Voice Live-facing client) use `wss://`. Zero `ws://` connections initiated or accepted.	Config validation + integration test	Zero plaintext WebSocket connections	FR26

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-SEC4	**Zero Critical/High CVEs** — no dependencies with CVSS score ≥ 7.0 in the production dependency tree at build time.	OWASP dependency-check report	Zero CVSS ≥ 7.0	SEC-REQ-16, AC-21
NFR-SEC5	**Zero Information Leakage** — HTTP error responses, WebSocket close frames, and log output visible to external consumers SHALL contain no stack traces, class names, internal paths, or framework-specific error details.	Security test suite + penetration test	Zero leakage findings	SEC-REQ-10, FR40, AC-20
NFR-SEC6	**Actuator Isolation** — Spring Boot Actuator endpoints bound to management port (C-48, default 8081), exposing only `/health/liveness`, `/health/readiness`, `/health/startup`. All other actuator paths return 404. Management port not routed through external load balancer.	Integration test: request all actuator paths on both ports	Zero exposed non-health actuator endpoints	SEC-REQ-9, FR39, AC-19
NFR-SEC7	**Event Replay Protection** — Event Grid events with `eventTime` older than C-45 (default 300s) SHALL be rejected. Zero stale events processed.	`ivr.security.events.replayed` (Counter)	Zero stale events accepted	SEC-REQ-14, FR37, AC-16
NFR-SEC8	**Zero Audio in Log Output** — raw audio bytes (PCM samples, base64-encoded audio content) SHALL NOT appear in log output at any level (TRACE, DEBUG, INFO, WARN, ERROR). Only audio metadata (frame size, sequence number, timestamp) may be logged. Validated by test that captures all log output during a full call lifecycle and asserts zero audio byte content.	Integration test assertion (AC-17)	100% pass — zero audio content in any log line	STRIDE S-2, SEC-REQ-5, AC-17
NFR-SEC9	**Zero Audio Bytes at Rest** — audio bytes SHALL exist only in JVM heap memory during active forwarding. No `FileOutputStream`, no disk write syscalls for audio data, no blob storage writes. Container filesystem is ephemeral. Distinct from NFR-P8 (zero-copy is performance); this NFR is a data safety / LGPD biometric-data guarantee.	Code review + runtime verification (no audio file I/O)	Zero audio bytes persisted to any storage	SEC-REQ-5, LGPD Assessment #1
NFR-SEC10	**Zero Rate-Limiter False Positives for ACS Traffic** — connections originating from ACS IP ranges (`52.112.0.0/14`, `52.122.0.0/15`) SHALL never be rejected by the per-IP rate limiter (FR38). Rate limiting applies to non-ACS source IPs only, since ACS may multiplex many callers through shared edge IPs. ACS connections are authenticated via JWT (NFR-SEC1). *(Constraint Mapping CT-2)*	`ivr.security.websocket.rate.limited` with tag `source=acs`	0 ACS-origin rejections	SEC-REQ-15, FR38

Category 5: Integration Reliability (7 NFRs)

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-I1	**Fallback Prompt Coverage** — every failure mode that leaves the caller in silence (VL disconnect, VL session timeout, circuit breaker open, admission rejection, first-audio timeout) SHALL play an appropriate fallback audio prompt before disconnecting.	Integration tests per failure mode	100% coverage for all 5 defined failure modes (FR35a–e)	UX-REQ-1, UX-REQ-3, UX-REQ-10, UX-REQ-11
NFR-I2	**No Reactive Types** — zero usage of `Mono`, `Flux`, `Publisher`, or any Project Reactor / RxJava types in production code. All async I/O via virtual threads (`spring.threads.virtual.enabled=true`).	Static analysis (class usage scan)	Zero reactive type references	AP-2, .github/copilot-instructions.md
NFR-I3	**Lenient Voice Live Deserialization** — Voice Live JSON events with unknown fields SHALL be deserialized without error. Parse failures SHALL increment a counter metric, not crash the session.	`ivr.voicelive.parse_errors_total` (Counter) + test with extra fields	Zero session crashes from unknown fields	FR45, API Backend Parsing Rules

ID	Non-Functional Requirement	Metric	Target	Trace
NFR-I4	**Config-Only Voice Live Version Changes** — updating the Voice Live API version (C-08) SHALL require only a configuration change. Zero code modifications, zero recompilation.	Config change + restart test	Version change = config-only deploy	AP-6
NFR-I5	**Cloud-Free Test Suite** — `./mvnw test` SHALL pass with zero Azure environment variables configured, zero network calls to any Azure service. JWT validation tests SHALL use an **in-process test JWT issuer** (e.g., Nimbus JOSE with generated keys) that exercises the full validation code path (signature verification, expiry checks, claims validation) — "cloud-free" means no external network calls, not untested security logic. *(Constraint Mapping CT-6)*	`./mvnw test` with no Azure env vars; network monitor confirms zero outbound Azure calls	100% pass, zero cloud dependencies	AP-5, Success Criteria, DoD
NFR-I6	**Developer Onboarding Speed** — elapsed time from `git clone` to first real phone call (with pre-provisioned Azure resources, `az login` completed). Measures developer experience quality.	Timed onboarding test (manual or scripted)	≤ 30 minutes	Success Criteria (Business Success)
NFR-I7	**Deterministic Single-Command Build** — `./mvnw clean package` SHALL produce a runnable JAR with zero manual steps, zero external tool installations beyond JDK 21 and Maven wrapper (included in repo).	Build reproducibility test	Single command, deterministic output	Success Criteria (Technical Success), DoD

NFR Constraint Map

Critical tensions between NFRs identified and resolved during Constraint Mapping elicitation. Referenced inline in affected NFRs.

ID	Tension	NFRs Involved	Resolution
CT-CRITICAL	Reconnection window (5s) vs. linked teardown (3s) — mathematically contradictory	NFR-R2, NFR-R12	Disambiguated by close code: normal close → R2 (immediate teardown); abnormal close → R12 (reconnection window)
CT-1	VL connect p99 (3s) consumes entire first-audio budget (3s p95)	NFR-P1, NFR-P6	Tightened P6 to ≤2s p95, ≤3s p99
CT-2	Per-IP rate limiting blocks ACS shared-IP edge nodes	NFR-SEC10, FR38	Rate limiting scoped to non-ACS IPs; ACS authenticated via JWT
CT-3	50 calls × bidirectional PCM × base64 = 6.6 MB/s > 5 MB/s limit	NFR-S1, NFR-S6	Raised S6 to ≤10 MB/s
CT-4	Startup time (30s) thin margin with credential acquisition	NFR-P9, NFR-R10, FR30	Credential acquisition capped at 15s; assumes workload identity
CT-5	Stall detection false positives vs. call completion rate	NFR-R8, NFR-R13	Acceptable at <1% if C-55 ≥ 5s; operational tuning note added
CT-6	Cloud-free tests vs. 100% JWT validation	NFR-I5, NFR-SEC1	In-process JWT issuer exercises full validation logic
CT-7	Cold-start JWKS fetch adds 200–800ms to first call	NFR-P1, NFR-SEC1	JWKS pre-warmed during startup; adds ~500ms to boot (within P9's 30s)

NFR × Priority Tier Cross-Reference

Priority	NFRs
P0 (MVP)	NFR-P1–P6, NFR-P8–P9, NFR-R1–R12, NFR-S1–S6, NFR-SEC1–SEC10, NFR-I1–I5, NFR-I7
P0 (measurable post-launch)	NFR-P7 (profiler needed), NFR-R13 (post-hoc analysis), NFR-I6 (manual onboarding test)
Ongoing	NFR-R8 (call completion rate — monitored continuously), NFR-R11 (observability completeness — validated per release)