Exotel AppBazaar Integration – Issue & Resolution Notes

This document records the Exotel-specific issues encountered during Week 2 Day 4 integration.

The system successfully placed outbound calls using Exotel APIs. Phones rang correctly and provider_call_id values were persisted in the database. However, call lifecycle events such as answered, no_answer, and completed were not received by the backend.

Root Cause Analysis:

Exotel AppBazaar does not expose a standalone status callback configuration in the current UI. Instead, it relies on a Flow Builder / Primary URL model where the provided URL is treated as a call-flow controller, not a pure status webhook.

This means:
- The URL is called to control dialing logic
- Exotel does not automatically POST lifecycle events to arbitrary backend endpoints
- Any lifecycle data must be forwarded manually by the AppBazaar app itself

Incorrect Assumption Identified:

It was initially assumed that setting a public URL (via ngrok) would allow Exotel to POST call lifecycle events directly to the backend. This assumption is valid for providers like Twilio but is not valid for Exotel AppBazaar.

As a result, calls remained in CLAIMED state because no_answer events were never delivered to the backend.

Correct Models Identified:

Model 1: AppBazaar Forwarding
The AppBazaar application (often implemented in Python) must explicitly forward call events to the backend via HTTP.

Model 2: Provider-Agnostic Timeout
The backend treats CLAIMED calls exceeding a time threshold as no_answer and applies retry/failure logic without provider input.

For Week 2 scope, Model 2 was selected to avoid provider lock-in and UI limitations.

Final Conclusion:

The backend design is correct.
The scheduler, database schema, and retry logic are correct.

The limitation exists in Exotel AppBazaar’s event delivery model.

This document exists to prevent re-investigation of the same issue in the future.