

Ticket 6: ABC-2832

Status: Complete
Priority: Critical
Summary: Database connection pool exhaustion causing service timeouts during peak load
Reporter: Thomas Anderson (Enterprise Solutions Group)
Assignee: Rachel Kim (Database Engineering)
Created Date: May 22, 2025
Updated Date: May 29, 2025

Description

Customer-Reported Symptoms: Enterprise Solutions Group experienced complete Voice Receptionist outages during peak business hours, with new calls receiving HTTP 504 Gateway Timeout errors. The system became unresponsive for 15-20 minute periods, affecting their multi-tenant deployment serving 12 client companies. Database-dependent features like caller history lookup and personalized greetings failed completely.

Error Messages/Logs:

```
[2025-05-22 14:23:15] ERROR ConnectionPool: All connections exhausted
[2025-05-22 14:23:16] FATAL DatabaseManager: Connection timeout after 30s
[2025-05-22 14:23:17] ERROR CallerHistoryService: Unable to fetch caller history
[2025-05-22 14:23:18] WARN HealthCheck: Database connectivity failed
[2025-05-22 14:23:19] ERROR LoadBalancer: All backend instances rejected requests
```

Environmental Details: - Database: PostgreSQL 15.3 on AWS RDS (db.r6g.2xlarge) - Application Server: Node.js 18.16.0 with pg-pool connection manager - Connection Pool: Default configuration (50 max connections, 30s timeout) - Peak concurrent sessions: 350+ simultaneous voice calls - Database workload: 2,400 queries/minute during peak - Geographic deployment: Multi-region (us-east-1, us-west-2)

Impact on Business Operations: - 12 client companies affected simultaneously - Service unavailable for cumulative 4.5 hours over 3 days - Estimated revenue impact: \$85,000 in SLA penalties - Emergency escalation to C-suite level - Customer retention risk for 3 major accounts

Resolution Details: Root Cause Analysis: The default PostgreSQL connection pool size of 50 was insufficient for the application's peak load patterns. Each voice session required multiple database connections for caller lookup, session state management, and audit logging. The connection pool configuration hadn't been updated since the initial deployment when the customer base was 1/7th the current size.

Applied Fixes: 1. **Database Connection Pool Optimization:** - Increased max connections from 50 to 200 per application instance - Implemented connection pooling with PgBouncer for additional layer - Configured statement-level pooling to reduce connection hold time - Set aggressive idle connection timeout (5 minutes vs previous 30 minutes)

1. Database Server Scaling:

2. Upgraded RDS instance from db.r6g.2xlarge to db.r6g.4xlarge
3. Increased max_connections parameter from 100 to 500
4. Optimized shared_buffers and work_mem for increased concurrent load






5. Enabled connection logging for ongoing monitoring

6. Application Architecture Improvements:

7. Implemented Redis caching layer for frequently accessed caller data
8. Added database query batching to reduce connection requirements
9. Deployed connection health monitoring with automatic failover
10. Introduced graceful degradation mode when database load exceeds thresholds

11. Monitoring and Alerting Enhancements:

- 12. CloudWatch alarms for connection pool utilization >80%
- 13. Real-time database performance dashboard
- 14. Automated scaling triggers for connection pool exhaustion
- 15. Customer-facing status page integration

Verification Steps: - Load testing with 500 concurrent sessions for 2 hours:  Passed - Database failover testing during peak load: 
Passed - Connection pool exhaustion simulation:  Graceful degradation confirmed - Customer acceptance testing with all 12 client companies:  Approved - 7-day production monitoring with no connection issues:  Stable