Agentic Virtual Agent Architecture

Isocktank Insurance — Eligibility, FNOL, Claim Status

Assumptions

Customer Context

- P&C insurer with US + EU operations
- 3,000 contact center agents (in-house + BPO)
- Three use cases: Eligibility, FNOL, Claim Status

Technology Stack

- Cloud: AWS
- Back-office AI: Bedrock Agents (Strands SDK + AgentCore) — already in production
- Claims platform: REST APIs, ~800ms latency
- Knowledge: Policy docs + procedures in S3 (greenfield vector DB)
- Vector DB: OpenSearch Serverless + Nova multimodal embeddings
- Identity: Okta (employees/BPO), custom portal auth (members)
- Contact center: Genesys Cloud

Regulatory & Compliance

- HIPAA, CCPA, GDPR
- 7-year data retention
- Human-in-the-loop: claim decisions, sensitive FNOL, disputes
- SOC2 Type II environment

Operational

- Voice latency budget: <300ms round-trip
- Digital: web chat, mobile, SMS, WhatsApp (EU)
- BPO uses same systems (no data isolation)
- Broker access: delegated OAuth, scoped to their book

High-Level Flow

1. Member initiates contact (voice or digital) → Genesys Cloud routes to virtual agent
2. Authentication (ANI + OTP for voice, session token for digital)
3. Intent classification → route to appropriate use-case agent
4. Agent reasons, retrieves knowledge, calls tools, responds
5. Escalation triggers → warm handoff to human with full context
6. All interactions logged for audit + 7-year retention

1. Agent Reasoning (Orchestration)

Architecture: Bedrock Agents with Strands SDK for orchestration

Why This Pattern

- Strands provides structured agent loops with tool use, retry logic, and trace logging
- Consistent with existing back-office agents — reuse patterns, reduce learning curve
- Claude on Bedrock for reasoning — long context (200k) handles complex policy docs

Agent Structure

- Supervisor agent: Intent classification, routing, escalation decisions
- Specialist agents: One per use case (Eligibility, FNOL, Claim Status)
- Each specialist has scoped tools + knowledge (principle of least privilege)

Reasoning Guardrails

- Max reasoning steps (prevent infinite loops)
- Mandatory tool confirmation for state-changing actions (e.g., file FNOL)
- Bedrock Guardrails: PII redaction, denied topics (medical advice, legal advice)

2. Knowledge Retrieval (RAG)

Ingestion Pipeline

- Source: S3 (policy PDFs, procedure docs, FAQ)
- Processing: Textract for PDFs, chunking (512 tokens, 20% overlap)
- Embeddings: Amazon Nova multimodal (handles text + images in docs)
- Storage: OpenSearch Serverless (vector + keyword hybrid search)
- Metadata: doc_type, effective_date, state, product_line (for filtering)

Retrieval Strategy

- Hybrid search: semantic (vector) + keyword (BM25) with RRF fusion
- Metadata pre-filtering: member's state, product, policy effective date
- Reranking: Cross-encoder rerank top 20 → return top 5
- Citation tracking: Every retrieved chunk tagged with source doc + page

Knowledge Drift Mitigation

- S3 event triggers re-ingestion on doc update
- Effective date filtering ensures current policy version used
- Monthly freshness audit: flag docs not updated in 90 days

3. Tool Use (APIs & Actions)

Tool Inventory

`get_eligibility` [Read] — Check member coverage, benefits, deductibles
`get_claim_status` [Read] — Retrieve claim state, timeline, next steps
`create_fnol` [Write] — File first notice of loss (requires confirmation)
`get_policy_docs` [Read] — RAG retrieval from knowledge base
`transfer_to_agent` [Action] — Escalate to human with context

`schedule_callback` [Write] — Book callback slot

Latency Handling

- Claims API ~800ms — too slow for voice without mitigation
- Strategy: Predictive prefetch on authentication (get eligibility + recent claims)
- Cache: ElastiCache (Redis) with 5-min TTL for read operations
- Fallback: "Let me check that for you" + streaming partial response

Write Operation Safety

- create_fnol requires explicit member confirmation before execution
- Idempotency keys prevent duplicate submissions on retry
- Compensation pattern: if downstream fails, log for manual review

4. Authentication & Authorization

Member Authentication

- Voice: ANI lookup + OTP to registered mobile (or security questions fallback)
- Digital: Session token from member portal (OAuth 2.0)
- Step-up auth: Sensitive actions (FNOL, payment) require re-verification

Broker Authentication

- Delegated OAuth via Okta
- Scope: Limited to their book of business (agency_id claim in token)
- Audit: All broker actions logged with broker_id + member_id

Tool-Level Authorization

- Virtual agent has service account with scoped permissions
- Member context (member_id) passed to every API call — claims API enforces access
- No ambient authority: agent can't access data outside current session context

Session Management

- Session state in DynamoDB (TTL: 24 hours)
- Encrypted at rest (KMS), member_id as partition key
- Session includes: auth level, conversation history, retrieved docs, tool calls

5. Conversation State & Handoff

State Management

- Conversation history: DynamoDB (short-term) → S3 (long-term, 7-year retention)
- Channel switching: Unified session ID across voice/digital — context persists
- Genesys interaction attributes: Pass session_id to enable context lookup

Escalation Triggers

- Explicit: Member requests human
- Sentiment: Frustration/anger detected (Genesys sentiment analysis)

- Complexity: Agent confidence below threshold, or >3 failed tool calls
- Policy: FNOL involving injury/fatality always escalates
- Compliance: Member disputes AI answer, requests review

Handoff Context Package

- Summary: LLM-generated 3-sentence summary of conversation
- Intent: What member was trying to accomplish
- Actions taken: Tools called, results returned
- Open questions: What the agent couldn't resolve
- Retrieved docs: Links to source documents referenced
- Delivered via: Genesys agent desktop screen pop

6. Platform Services (Genesys)

Note: *High-level mapping; will refine during Genesys ramp-up*

- Routing: Genesys Predictive Routing to select best human agent on escalation
- Voice: Genesys ASR/TTS or BYOT (Bring Your Own Transcription) with Amazon Transcribe
- Digital: Genesys Web Messaging / Open Messaging for chat channels
- Agent Desktop: Genesys Agent Workspace — screen pop with context package
- Analytics: Genesys Interaction Analytics for sentiment, topic, CSAT
- Workforce Management: Forecast volume shifts from AI containment

7. Audit & Observability

Logging Strategy

- Every LLM call: prompt, response, latency, tokens, trace_id
- Every tool call: input, output, latency, success/failure
- Every retrieval: query, chunks returned, sources, rerank scores
- Destination: CloudWatch Logs → Kinesis → S3 (7-year retention)

Explainability

- Trace ID links conversation → agent reasoning → tool calls → sources
- On audit request: Reconstruct full decision chain from logs
- Source attribution: Every AI statement tagged with doc_id + chunk_id

Monitoring & Alerting

- Latency: P99 response time by channel (voice vs digital)
- Errors: Tool failure rate, LLM timeout rate
- Quality: Hallucination flags (human review sample), containment rate
- Alerts: PagerDuty integration for SLA breaches

8. Multi-Region (US + EU)

- Separate deployments: us-east-1 (US), eu-west-1 (EU)

- Data residency: EU member data never leaves EU region
- Shared: Model (Claude via Bedrock available in both), code artifacts
- Separate: DynamoDB tables, S3 buckets, OpenSearch clusters, logs
- Routing: Genesys routes by member region; failover is NOT cross-region (compliance)

9. Graceful Degradation

- LLM timeout: Fall back to scripted IVR / transfer to human
- Claims API down: Inform member, offer callback, log for retry
- RAG failure: Proceed with agent reasoning but flag low confidence
- Full outage: Genesys routes directly to human queue (bypass virtual agent)
- Key principle: Never leave member in dead-end state