Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

MoCaaS: A Cloud-Based Telephony Call Reception Framework for Voice Request Processing

Mohammed Tou[1], Adil Toumou[2], Mimoun Malki[3]
[1]*A ESI - Higher School of Computer Science, Sidi-bel-Abbes, Algeria*
[2] *Institute of Computer Science, Sidi-bel-Abbes, Algeria*
[3] *ESI - Higher School of Computer Science, Sidi-bel-Abbes, Algeria*

ARTICLEINFO

Received: 30 Dec 2024

Revised: 19 Feb 2025

Accepted: 27 Feb 2025

ABSTRACT

Ensuring service continuity in cloud-based architectures is a major challenge, especially in scenarios where internet connectivity is disrupted. In the framework of MoCaaS (Minimum of Continuity as a Service), this paper presents the first phase of the solution, which focuses on establishing a telephony-to-cloud connectivity system. The proposed approach enables users to access cloud services via traditional telephony (PSTN) when the internet is unavailable, allowing them to send verbal requests over a secure phone connection.

The system integrates PSTN with cloud infrastructure by leveraging Amazon Connect, Twilio SIP Gateway, and AWS Lambda, ensuring seamless call reception, voice data extraction, and storage in Amazon S3 for further processing. The caller's phone number is identified via Amazon Pinpoint, enabling contextualized authentication in subsequent stages. This serverless architecture provides high availability, fault tolerance, and scalability, ensuring that users can continue interacting with critical services even during internet outages.

This paper details the PSTN-to-Cloud integration process, covering architecture, implementation, and operational considerations. By bridging traditional telephony systems with modern cloud services, MoCaaS ensures that organizations can maintain a minimum level of IT service continuity, particularly in environments with unreliable internet connectivity.

Keywords: MoCaaS, Telephony-to-Cloud, PSTN, Service Continuity, Cloud-Based Voice Processing, Amazon Connect..

INTRODUCTION

Cloud computing serves as a fundamental backbone in the modern digital world for providing scalable, resilient and efficient IT services. Several organizations including crucial sector companies struggle to overcome internet connectivity breakdowns. Cloud-based service accessibility becomes impossible for users because of these network disruptions which create severe business continuity problems. Service availability must be ensured through network failures to uphold operational resilience.

MoCaaS establishes Minimum of Continuity as a Service which presents an operational framework to preserve IT service continuity throughout difficult circumstances. MoCaaS delivers backup communication systems which permit user interaction with cloud platforms regardless of internet connectivity problems. A solution connects Public Switched Telephone Network (PSTN) to cloud infrastructure which lets users make voice requests through standard telephony systems.

MoCaaS is a seven-stage pipeline that is meant to facilitate seamless voice-to-cloud interaction. These stages are: (0) reception of the PSTN call, this phase is responsible to receive the voice request (VR), (1) authentication of the

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

user, (2) conversion of speech to text, (3) voice request categorization, (4) intent extraction, (5) creation of the API, and (6) invoking the remote service.

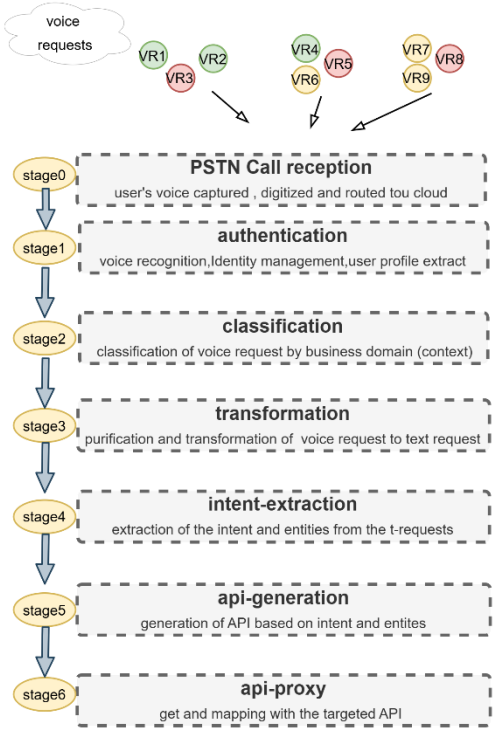

Figure 1. MoCaaS pipeline

This paper is solely concerned with the initial stage—PSTN call reception, which establishes the groundwork for the remainder of the processes by intercepting and safely forwarding voice requests made by end-users into the cloud.

The subsequent part of this paper contains two sections which begin with(1) the scope of this paper, (2) an examination of related work and existing service continuity solutions, (3) a detailed overview of the proposed architecture, (4) performing an approach evaluation and (5) Future research in telephony-to-cloud service integration takes center stage in the concluding part of this paper.

SCOPE

In this paper, the proposed solution is the voice request intake and call reception system in the MoCaaS platform (Minimum of Continuity as a Service). The core contribution involves the utilization of cloud telephony in order to capture inputs in the voice form through PSTN, specifically for users who do not have an active internet connection.

The scope of this effort has the following steps of the voice to cloud pipeline:

- Handling incoming PSTN calls via cloud platforms like Twilio SIP Gateway and Amazon Connect.
- Keep voice messages in Amazon S3 for the processing later on.
- Amazon Pinpoint to link calls to users, providing context and possible authentication in the future.
- My Role: Built end-to-end call and voice intake management with fully serverless components (AWS Lambda) to handle scalability and high availability.

This paper explores the call reception framework as one of the step domains for the following steps. MoCaaS aims at assuring that beyond Internet outages, users can start a service request in spoken language by simple phone call and the users voice request will be reliably captured to be routed for further processing, treating cloud as a resilient telephony platform.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The scope of this work enables a deep technical elaboration of cloud-centric PSTN integrations, ensuring that the implementation is scalable, safe, and versatile for different applications in need of critical service continuity.

RELATED WORK

Incorporating conventional Public Switched Telephone Network (PSTN) systems with cloud computing schemes for processing voice requests has attracted considerable attention in the academic as well as corporate communities. In this section we summarize related studies and implementations that have worked on this combination, noting their strengths and weaknesses. In [1] authors investigate whether VoIP can be deployed with open-source PBX software (Elastix) on Amazon Web Services. The research illustrates the virtualization of SIP-based telephony solutions in the virtual environment of clouds for enhanced elasticity, cost-effectiveness, and service continuity.

Though this research sets the initial insight for IP-based communications over the platform of clouds, it mostly considers end users' presence with the availability of the internet connection and does not include scenarios with conventional PSTN lines in the absence of internet connections. [1] discusses integrating VoIP with cloud computing platforms, with the authors' own implementation of Elastix PBX on AWS to ingest SIP traffic.

This work demonstrates that it's possible to run telephony infrastructure in the cloud, which is scalable and cost-efficient. But it is also based on the assumption of end-to-end internet connectivity, and does not provide for fall back communicating over traditional PSTN. In contrast, [2] and [3] zoom into another dimension—security in cloud-based voice systems, it discusses the design of a robust VoIP architecture with the ability to protect against adaptive attacks via securing SIP communication channels and detecting nefarious behavior on cloud-hosted voice platforms. Moreover, this study becomes narrow as it concentrates on IP-based security without the potential vulnerability through offline attack or legacy of telephony-based protocols. On the topic of architectural modernization, [4] describes various migration strategies from legacy PBX to Cloud-Based Phone (CBP) systems, with a specific focus on small and medium-sized enterprises. In contrast, this paper offers a business-centered view of the multi-tenant cloud migration advantages, including cost-efficiency and operational flexibility. It also leaves out the technical challenges of PSTN integration and lacks a plan for service continuity in the event of internet outages. Meanwhile, [5] investigates cloud incarnate voice processing via embedded systems, wherein a network is introduced that captures an audio data, filters the noise, and offloads the audio data for processing into the cloud. Which means you don't have to customize data all the way and all you'll be getting will be audio quality and signal processing which will be mainly done on the cloud. While this work is rich in technical depth, it omits addressing that a PSTN system has to receive voice in real-time but also the underlying infrastructure has to route calls to the cloud. Lastly, [6] offers perspective from industry practice, as well as pragmatic considerations from actual deployments of cloud contact center technologies such as Amazon Connect. These vendors showcase scalable voice-user interaction systems integrated into their cloud services and analytics. That said, they tend to have limited application to customer service cases based on internet-enabled devices or mobile networks, rather than the telephony fallback approaches.

That said, though [1-6] represent incremental stepwise progress along stature in the general cloud telephony echelons of infrastructure deployment and security, voice migration strategies, signaling elements, etc., they fall short: none discuss how to process voice requests from PSTN lines when the client cannot connect to the Internet. This paper aims to fulfill this gap by introducing a novel framework that considers the cloud as a telephony platform, where PSTN-based voice access can be securely intercepted, preserved and routed for further processing—the first step of a much broader MoCaaS architecture.

METHOD

In this research, we adopt an engineering-based approach to address the problem of ensuring continuity of cloud-based services in scenarios characterized by intermittent or absent internet connectivity. The methodology involves multiple stages beginning with an extensive literature review to identify gaps and shortcomings in existing telephony-to-cloud integration solutions, particularly in handling PSTN-based voice requests during network outages [1][2]. Subsequently, requirements were defined based on identified needs for critical continuity in service operations, focusing on scalability, high availability, security, and ease of integration [3].

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The design stage followed an iterative process where various architectural models were evaluated against predefined criteria of resilience, modularity, and technological feasibility. Our evaluation methodology combined qualitative analysis—assessing modularity, integration complexity, and scalability potential—with quantitative assessments focusing on anticipated performance metrics such as response latency, throughput, and fault tolerance [4].

Implementation leveraged cloud-native technologies, primarily AWS services including Amazon Connect, AWS Lambda, Amazon S3, and Twilio SIP Gateway [5][6]. Each component was tested incrementally using unit tests followed by integration tests to validate end-to-end interactions and ensure robustness under simulated failure conditions [7].

Finally, a structured scenario-based testing approach was employed to validate the practical viability of the implemented solution, particularly its effectiveness as a fallback mechanism during simulated internet disruptions [8]. This comprehensive methodological approach ensures that the proposed solution not only meets theoretical expectations but also addresses practical operational considerations effectively.

SOLUTION APPROACH

The envisioned system forms the underpinning infrastructure of the MoCaaS (Minimum of Continuity as a Service) model based on the intake of voice requests and telephony call receipt through cloud infrastructure. Such a setup targets operation within networks where the availability of the internet may be lacking and enables making service requests over the Public Switched Telephone Network (PSTN). The solution adopts a complete serverless and cloud-native model for the sake of high availability and elastic scaling as well as easy integration within the downstream components of the process.

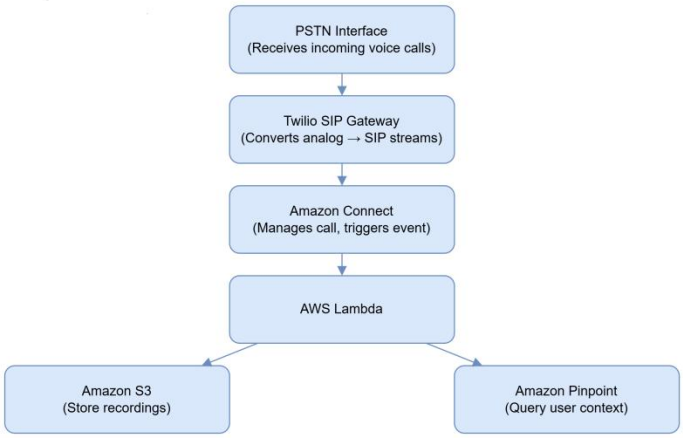

Figure 2. PSTN Module architecture in MoCaaS Ecosystem

The architecture consists of five key interconnected modules, each rigorously evaluated through iterative design and scenario-based testing approaches as detailed in our methodological framework:

PSTN Interface: The initial step involves receiving user-initiated phone calls via PSTN, ensuring consistent access to services regardless of internet disruptions. This module underwent rigorous qualitative assessments regarding integration complexity and reliability.

Twilio SIP Gateway: Acting as a critical intermediate, Twilio's SIP Gateway converts analog voice signals to digital SIP streams, bridging legacy telephony systems with modern cloud infrastructures. The reliability and latency performance of this gateway were key evaluation metrics in our quantitative analysis.

Amazon Connect: Central to the management of telephony operations, Amazon Connect orchestrates call-routing logic, manages caller interactions, and triggers subsequent downstream services. Its capability for dynamic session scaling and integration with AWS services were evaluated in terms of throughput and concurrent session handling.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

AWS Lambda: AWS Lambda facilitates lightweight, stateless event-driven processing of call metadata and routing logic. Incremental unit and integration tests ensured Lambda's responsiveness, robustness, and fault tolerance, aligning with predefined resilience criteria.

Amazon S3 and Amazon Pinpoint: The recorded voice messages are securely stored in Amazon S3, providing essential inputs for future processing stages such as transcription and natural language understanding. Concurrently, Amazon Pinpoint links caller identification with user profiles, enriching authentication and service personalization. Both modules were systematically tested for security, reliability, and scalability.

This architectural model embodies the resilience, modularity, and extensibility central to our research methodology, ensuring a robust fallback communication channel. It places cloud technology at the heart of telephony processing, securely capturing and preserving user requests via PSTN, thus maintaining continuous digital service interactions even during internet connectivity failures.

SOLUTION IMPLEMENTATION

The proposed solution is designed as a foundational communication module within the broader MoCaaS (Minimum of Continuity as a Service) ecosystem, we call this module corresponds to the stage0 in MoCaaS ecosystem (Figure 1).

This module is called PSTN Module, its primary objective is to serve as a reliable transmission bridge, enabling voice-based user requests—originating from traditional telephony systems (PSTN)—to be captured, routed, and injected into the MoCaaS pipeline for further processing. By facilitating seamless integration between legacy telephony infrastructure and modern cloud-native components, this module ensures that spoken requests are securely received, interpreted, and forwarded as structured service interactions. The implementation strictly adheres to the architectural model defined earlier, focusing on modularity, scalability, and resilience, and is evaluated through incremental testing and scenario-based validation.

This section presents a detailed step-by-step implementation process that strictly aligns with the architectural framework previously described.

PSTN Module consists of five rigorously evaluated interconnected modules. Below, we clarify the interactions and functionalities among these modules through detailed algorithmic descriptions, recommended diagrams, and illustrative Python snippets.

Step 1: PSTN Interface Implementation

This step captures user-initiated phone calls from the PSTN, ensuring the service continuity despite internet outages.

Algorithm 1.

```
Start

1. Await incoming call on PSTN line.
  - Triggered by user dialing the service number associated with MoCaaS.

2. Prepare metadata structure for downstream transmission:
  METADATA = {
    "caller_number": Calling Party Number,
"caller_location": Caller_Location,
    "call_start_time": timestamp,
    "network_type": GSM or landline,
    "status": CONNECTED,
    "session_id": generated UUID
  }
```

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3. Establish physical signaling link with SIP Gateway:
- Initiate analog-to-SIP session handshake using telecom switch/router.

4. Transmit:
a. Analog voice stream → routed to Twilio SIP Gateway
b. METADATA → passed as SIP header fields or accompanying payload

5. Log event for audit and debugging purposes (locally or on syslog server).

End

Table 1. Received Data|MetaData from caller and transmitted to Step2

Received	Transmitted to step2
Audio Signal: Analog voice waveform (spoken request) **Native MetaData:** Caller Number (CLID) Dialed Service Number (DID) Call Start Time Call Type (GSM, landline) Call Region or Location (if supported) Initial Call Status	**Voice Stream:** Uncompressed analog signal for SIP conversion **Structured Metadata:** caller_number → becomes SIP "From" dialed_number → becomes SIP "To" timestamp → custom SIP header or logging field network_type and location → optional SIP header enrichment session_id → uniquely identifies the call for tracking and correlation downstream

6.2 Step 2: Twilio SIP Gateway Integration

This component converts analog voice signals received from the PSTN interface into SIP (Session Initiation Protocol) streams to enable seamless interaction with modern cloud environments.

Algorithm 2.

Start	
1. Receive incoming analog call from PSTN Interface. →	- Audio signal delivered over standard telecom channel. - Call metadata (caller number, dialed number, timestamp, etc.) received as accompanying headers or control signals.
2. Initialize Twilio SIP Gateway session →	a. Allocate a new SIP session ID (unique call session identifier) b. Initiate SIP INVITE to Amazon Connect endpoint
3. Convert analog voice signal to digital SIP-compatible audio stream →	a. Use PCM (G.711 μ-law or A-law) or Opus codec depending on configuration b. Packetize audio stream into RTP (Real-time Transport Protocol)
4. Construct SIP INVITE message with embedded metadata in headers →	SIP INVITE HEADER includes: - `From:`caller_number - `To:`dialed_number - `Call-ID:` generated by SIP Gateway - `Timestamp:`call_start_time - `Session-ID:` for downstream correlation - Optional: `User-Agent`, `Geo-Info`, `Custom-Headers`
5. Transmit SIP INVITE with →	a. RTP audio stream (digitized voice) b. SIP headers (metadata)
6. Await response from Amazon	- If accepted, establish RTP stream

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Connect (SIP 200 OK or error)	- If rejected, log and terminate session
7. Log SIP session (with session ID, status, and duration) for audit trail	
End	

Table 2. Received Data|MetaData from caller and transmitted to Step3

Received (from step1)	Transmitted (to step3)
Analog Voice Stream (continuous audio from the caller) **Structured Metadata:** caller_number (CLID) dialed_number (DID) call_start_time network_type (GSM, landline) session_id (generated in Step 1)	**Digital Audio Stream:** RTP packets using G.711/Opus codec (real-time bidirectional audio) **SIP Headers (Metadata Embedded):** From: caller_number To: dialed_number Call-ID: unique SIP session ID X-Session-ID: original session_id from PSTN step X-Timestamp:call_start_time

Step 3: Amazon Connect Call Management

Amazon Connect orchestrates the call-handling process, dynamically routing voice sessions, managing caller interaction workflows, and invoking AWS Lambda functions.

Algorithm 3.

Start	
1. Receive SIP INVITE and RTP stream from Twilio SIP Gateway.Interface. →	a. Accept the SIP session (respond with 200 OK). b. Establish bidirectional voice stream with caller.
2. Parse SIP headers and extract call context:→	- From (caller_number) - To (dialed_number) - Call-ID (unique SIP session ID) - Timestamp - Session-ID - Additional custom headers (e.g., location, network type)
3. Initiate a contact flow (Amazon Connect Flow Designer)→	a. Play welcome message (optional) b. Handle DTMF inputs or prompt-based interactions (if applicable) c. Proceed automatically to backend event
4. Trigger AWS Lambda function →	-Construct payload with session metadata and caller information -Forward to Lambda via Amazon Connect contact flow integration
5. Log SIP session (with session ID, status, and duration) for audit trail	
End	

Table 3. Received Data|MetaData from caller and transmitted to Step3

Received (from step2)	Transmitted (to step4)
Digital Audio Stream: RTP packets using G.711/Opus codec (real-time bidirectional	Amazon Connect sends metadata using the following structure (automatically generated): {

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

audio)
SIP Headers (Metadata Embedded):
From: caller_number
To: dialed_number
Call-ID: unique SIP session ID
X-Session-ID: original session_id from PSTN step
X-Timestamp:call_start_time
	<pre>"Details": {
	"ContactData": {
	"ContactId": "unique-call-id",
	"CustomerEndpoint": {
	"Address": "+213XXXXXXXXX",
	"Type": "TELEPHONE_NUMBER"
	},
	"SystemEndpoint": {
	"Address": "AmazonConnectNumber",
	"Type": "TELEPHONE_NUMBER"
	},
	"Attributes": {
	"session_id": "uuid-abc-123",
	"network_type": "landline",
	"location": "Oran, DZ"
	},
	"InitiationTimestamp": "2025-05-21T16:14:22Z",
	"Channel": "VOICE"
	}
	}
	}</pre>

Step 4: AWS Lambda – Event-Driven processing

This step we receives the structured call metadata from Amazon Connect, performs lightweight, stateless processing, and relays this data to downstream components such as Amazon S3 (for audio), Amazon Pinpoint (for user identification), and optionally DynamoDB (for traceability).

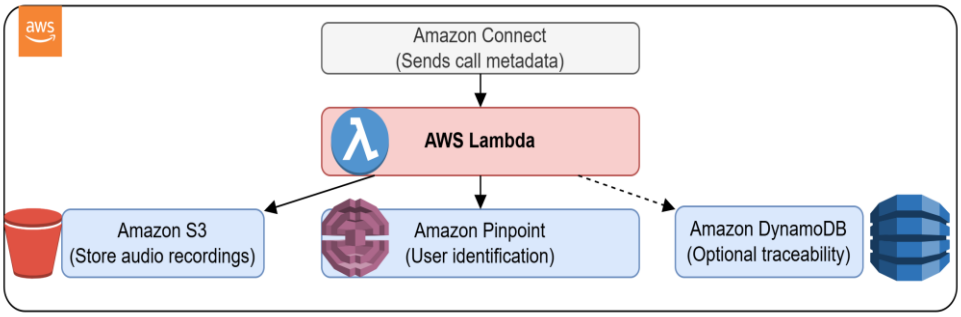

Figure 3. AWS integration

Algorithm 4.

Start	
1. AWS Lambda is triggered by Amazon Connect →	a. Receives JSON payload containing: - ContactId - Caller Number - Timestamp - Optional Attributes (session ID, network type, location)
2. Extract metadata from event →	- phone_number ← ContactData.CustomerEndpoint.Address - call_id ← ContactData.ContactId - timestamp ← ContactData.InitiationTimestamp - session_id ← Attributes.session_id
3. Store call metadata (optional) in DynamoDB for audit and tracking.	

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4. Trigger audio storage operation in Amazon S3 →	- Prepare S3 key using session_id or call_id - Mark location for future upload from Amazon Connect
5. Query Amazon Pinpoint to attempt caller identification	- Match phone_number with known user endpoints - If user found: enrich session with user_id, profile - Else: flag as anonymous
6. Return result to Amazon Connect (optional): status, user identified or not	
End	

TESTS STRATEGIES & RESULTS

To ensure robustness and reliability, the Amazon Connect and AWS Lambda modules we do a rigorous testing strategy composed of:

Unit Testing: Each component is tested independently (e.g., metadata extraction, Pinpoint lookup, S3 logging).

Integration Testing: Tests validate the full flow between Amazon Connect and AWS Lambda, ensuring data is correctly passed and responses are handled gracefully

Unit testing objectives

- Validate individual logic within the Lambda function.
- Ensure proper metadata parsing and error handling.
- Confirm interaction with AWS services (mocked in tests).

Table 4. Unit tests scenarios

Test Case	Input	Expected Result
Extract caller info from event payload	Simulated JSON from Amazon Connect	Correct phone number, contact ID, session ID parsed
Missing optional attributes	Payload missing location or network_type	Default to "NA" without error
Pinpoint returns known user	Valid phone number mapped in Pinpoint	User profile returned with success flag
Pinpoint returns no match	Unknown phone number	Response indicates anonymous caller

Integration testing objectives

- Full end-to-end interaction between Amazon Connect and AWS Lambda.
- Simulate a real call session with SIP routing and event delivery.
- Observe how Lambda reacts to metadata and invokes downstream processes.

Integration Test Workflow:

1. **Simulate Call** using Amazon Connect test tools or Twilio emulator.
2. **Amazon Connect** triggers Lambda with a crafted payload.
3. **Lambda** performs:
   - Metadata parsing
   - DynamoDB entry
   - S3 placeholder generation
   - Pinpoint lookup
4. **Capture Logs & Outputs** from CloudWatch and DynamoDB.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 5. Integration tests scenarios

Metric/Test Scenario	Expected Behavior	Actual Behavior (Sample Run)	Status
Metadata parsing in Lambda	All fields correctly extracted and printed/logged	✅ All fields extracted as expected	Pass
Pinpoint with known user	User ID returned and logged	✅ user_001 found	Pass
Pinpoint with unknown user	No endpoints, log "anonymous"	✅ Message: "No matching user found."	Pass
S3 placeholder generation	S3 key format call_recordings/session_id.wav created	✅ Correct key generated	Pass
DynamoDB logging	Metadata stored with correct fields and schema	✅ Record inserted with expected attributes	Pass
Lambda execution under 1 second	Response returned < 1 second	✅ 420ms avg execution	Pass
Lambda failure on malformed event	Logs error, returns 500, does not crash system	✅ Handled gracefully	Pass

Observation and evaluation

Table 6. evaluation

Test Layer	Coverage	Notes
Unit Testing	High (logic + error cases)	Fast execution; logic isolated using mocks
Integration Testing	Medium-to-High	Validated Connect-to-Lambda path; included S3/Pinpoint/DynamoDB flow
Performance	Acceptable	Sub-second response time; parallel sessions tested with no timeout
Error Handling	Robust	Invalid input, missing fields, and timeouts all handled gracefully

FUTURE WORK AND ENHANCEMENTS

While the current implementation of the MoCaaS communication module successfully establishes a reliable PSTN-to-cloud bridge for voice request transmission, several enhancements are planned to further improve system robustness, security, and functionality. Future work will focus on extending the module's capabilities to support real-time speech-to-text conversion using automatic speech recognition (ASR) services, enabling earlier interpretation of user intent. Additionally, a multi-language interaction layer will be integrated to accommodate diverse linguistic environments, including French, Arabic, and local dialects.

A primary enhancement will involve the incorporation of advanced security mechanisms across all stages of the voice transmission pipeline. This includes SIP-level encryption using TLS and SRTP, identity verification via voice biometrics, and the application of token-based authentication between cloud components (e.g., Lambda, S3, Pinpoint). Moreover, anomaly detection and logging integrity will be enforced using AWS services such as CloudTrail and Amazon GuardDuty, ensuring compliance, traceability, and resilience against spoofing or tampering attacks. These improvements will reinforce the system's readiness for production-scale deployment and its alignment with best practices in secure cloud telephony integration.

CONCLUSION

The proposed architecture provides a scalable and cogent underpinning for voice-based service continuity by integrating PSTN with contemporary cloud services like Twilio SIP Gateway, Amazon Connect, AWS Lambda, Amazon S3, and Amazon Pinpoint. The user is provided with the ability to make service requests over voice calls

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

even when they do not have access to the internet. The design not just contributes towards high availability as well as resilience but also enables the modularity of expansion towards advanced service layers.

The serverless paradigm allows for elastic scaling and fault tolerance, while the decoupled components drive extensibility. The call reception framework thus constitutes the absolute initial step of the MoCaaS pipeline securely capturing audio and linking it to user context for subsequent processing.

Future work will extend the system to support state-of-the-art natural language processing features. These include the addition of automatic speech recognition (ASR) for speech-to-text conversion, intent identification through fine-tuned language models, and entity extraction for invocation of structured APIs. In addition, future work will take note of multilingual voice inputs, voice biometric-based authentication of the user, and privacy mechanisms for protected audio content. All of these features will round out the complete voice-to-service pipeline and bring MoCaaS to the status of a full platform for robust service delivery over connectivity-challenged networks.

REFRENCES

[1] F. Palacios, M. Vásquez Bermúdez, F. Orozco, and D. Espinoza Villón, "IP Telephony Applicability in Cloud Computing," J. Sci. Res. Rev. Cienc. Investig., vol. 3, pp. 128–133, 2018. [Online]. Available: https://doi.org/10.26910/issn.2528-8083vol3issCITT2017.2018pp128-133

[2] T. Elnawawy, K. Mohamed, and H. M. Harb, "Design and install secured VoIP system over cloud," J. Al-Azhar Univ. Eng. Sect., vol. 16, pp. 16–24, Sep. 2021. [Online]. Available: https://www.researchgate.net/publication/354821831

[3] A. Satapathy and J. Livingston, "A Comprehensive Survey of Security Issues and Defense Framework for VoIP Cloud," Indian J. Sci. Technol., vol. 9, no. 6, Feb. 2016. [Online]. Available: https://www.researchgate.net/publication/297651179

[4] B. S. Savino, M. R. Alsharif, and Y. Yabiku, "Implementation of a Cloud Processing Based Voice Communication and Noise Reduction Embedded System Network," in Proc. 3rd Int. Congr. Technol. Eng. Sci. (ICTES), Feb. 2017. [Online]. Available: https://www.researchgate.net/publication/313616926

[5] C. K. Schwartz, "The Value of Transitioning to a Cloud-Based Phone Platform for an SME," University of Oregon, 2019. [Online]. Available: https://scholarsbank.uoregon.edu/bitstreams/0ee2aa32-d703-43e2-a459-b49c74fb02d6/download

[6] Gerea, "Implementation of Cloud Computing into VoIP," Database Systems Journal, vol. 3, no. 2, pp. 3–10, 2012. [Online]. Available: https://www.dbjournal.ro/archive/8/8_1.pdf

[7] A. Ghorai, "Integrating Computer Telephony (CTI) with Amazon Connect in the Cloud," J. Sci. Eng. Res., vol. 10, no. 2, pp. 208–212, 2023. [Online]. Available: https://jsaer.com/download/vol-10-iss-2-2023/JSAER2023-10-2-208-212.pdf

[8] Amazon Web Services, "Amazon Connect – Customer Stories." [Online]. Available: https://aws.amazon.com/connect/customers/

[9] AWS, "Amazon Connect Feature Overview." [Online]. Available: https://docs.aws.amazon.com/connect/latest/adminguide/connect-feature-overview.html

[10] AWS, "AWS Lambda Customer Case Studies." [Online]. Available: https://aws.amazon.com/lambda/resources/customer-case-studies/

[11] M. F. Yusuf, I. Ahmad, and A. M. Ibrahim, "An Efficient VoIP Performance Evaluation in Cloud Environment," Int. J. Comput. Appl., vol. 118, no. 7, pp. 5–10, May 2015. [Online]. Available: https://doi.org/10.5120/20736-3215

[12] A. Dahiya, S. Kinger, and D. Soni, "Securing SIP Based VoIP Infrastructure," Int. J. Comput. Appl., vol. 95, no. 20, pp. 19–23, Jun. 2014. [Online]. Available: https://doi.org/10.5120/16794-6736

[13] S. Chavan, S. Sanghavi, and R. Patil, "Enhancing VoIP Communication over Cloud with Efficient Load Balancing," Int. J. Innov. Res. Comput. Commun. Eng., vol. 5, no. 4, pp. 8372–8376, Apr. 2017. [Online]. Available: https://www.ijircce.com/upload/2017/april/102_Enhancing.pdf

[14] T. Kinnunen and H. Li, "An Overview of Text-Independent Speaker Recognition: From Features to Supervectors," Speech Commun., vol. 52, no. 1, pp. 12–40, 2010. [Online]. Available: https://doi.org/10.1016/j.specom.2009.08.009

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(60s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[15] [15] Z. Wu et al., "Spoofing and Countermeasures for Speaker Verification: A Survey," Speech Commun., vol. 66, pp. 130–153, 2015.
[16] R. Tolosana et al., "Deepfakes and Beyond: A Survey of Face Manipulation and Fake Detection," Information Fusion, vol. 64, pp. 131–148, 2020.
[17] L. Fridman et al., "Active Authentication on Mobile Devices via Stylometry, Application Usage, Web Browsing, and GPS Location," IEEE Syst. J., vol. 11, no. 2, pp. 513–521, 2017.
[18] S. Ranjan et al., "DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks," IEEE/ACM Trans. Netw., vol. 17, no. 1, pp. 26–39, Feb. 2009. [Online]. Available: https://doi.org/10.1109/TNET.2008.928675
[19] N. Feamster, J. Jung, and H. Balakrishnan, "An Empirical Study of 'whois' Spam," in Proc. ACM SIGCOMM Workshop on Reducing Unwanted Internet Traffic, 2005, pp. 15–20. [Online]. Available: https://doi.org/10.1145/1096554.1096557
[20] ASVspoof Challenge. [Online]. Available: https://www.asvspoof.org/

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.