HIGH VOLUME OUTAGE SCENARIO

TECHNICAL DESIGN DOCUMENT

Document History

Date	Version	Document Revision Description	Document Author	Email IDs
24-Mar-25	V0	Initial Draft of Technical Design document for HVOS in scope modules	Sukriti Singh	sukriti.singh@fpl.com
11-Apr-25	V1	Added the sections for Convo AI, OTR, Reporting and admin console	Sukriti Singh	sukriti.singh@fpl.com
28-May-25	V2	Updated the sections for No Match No Input scenarios, Updated the transfer flows and exit flow design, updated the test case tables for Lambda and Lex bots, added a logical design doc for contact flows, modules, lambdas and lex bots	Sukriti Singh	sukriti.singh@fpl.com
1-Jun-25	V3	Updated contact attributes as per new design for all contact flows. Added a logical design flow sheet for reference	Vipraj Deshpande	vipraj.pradeep.p.deshpande@pwc.com
5-Jun-25	V4	Updated Error handling section was per latest dev	Vipraj Deshpande	vipraj.pradeep.p.deshpande@pwc.com
6-Jun-25	V5	Updated the OTR section with latest transfer and exit flows	Vipraj Deshpande	vipraj.pradeep.p.deshpande@pwc.com
13-Jun-25	V6	Updated the formatting of document and Table of contents	Vipraj Deshpande	vipraj.pradeep.p.deshpande@pwc.com

Reviewers

Review Date	Version	Reviewed Role	Reviewer	Email IDs
			Ussama Baggili	ussama.baggili@pwc.com
			Ambuj Gupta	Ambuj.Gupta@fpl.com
			Baskar Narayanan	Baskar.Narayanan@fpl.com
			Rakesh Naik	

Table of Contents

Introduction

Florida Power and Light (FPL) is actively working towards enhancing its Customer Service Platform, specifically within its Contact Center, to provide a more efficient and user-friendly experience for customers. This document outlines the ongoing efforts to as a part of XD Program at FPL to expand the platform's capabilities by introducing new intents and refining the existing IVR functionalities. The enhancements are designed to improve customer interactions, streamline processes, and ensure that FPL continues to deliver exceptional service across all customer service channels.

Purpose of Document

This document provides a comprehensive overview of the enhancements made to Florida Power and Light's (FPL) customer service platform during Phase 1 of the XD project. It serves as a technical and functional guide for the integration of Interactive Voice Response (IVR), Chatbot, and Agent Assist systems, designed to improve customer experience and operational efficiency.

The document details the design, implementation, and interaction of key components such as AWS Connect flows, Lex bots, and Lambda functions. It also explains the error handling and fallback mechanisms that ensure system reliability.

The primary objectives are to:

- Provide in-depth explanations of the new and existing system components.
- Serve as a technical blueprint for the customer service platform.
- Outline error handling strategies and future scalability.
- This document is intended for developers, system architects, and customer service managers involved in the platform's development and operation.

Terms, Acronyms & Definitions

Conversational AI: A technology that allows users to interact with systems via natural language, typically using voice or text interfaces.

IVR (Interactive Voice Response): A technology that allows users to interact with a computer-operated phone system through voice and DTMF tones input via a keypad.

Intent: A specific customer request or action that the IVR system can recognize and respond to.

Slot: A data field used with the intent to capture specific information from the user.

Slot Priority: The ordered list of slots in an intent based on their sequence of elicitation.

Lambda Function: A serverless compute service that runs code in response to events and automatically manages the compute resources required by that code.

Storm Mode: An attribute that determines the availability of certain services during severe weather conditions.

AZC- Amazon connect

Naming Conventions

To maintain the consistency during the development across different applications and have a unified way. Please find the below link for the file with details

FPL Phase-1 - Naming Conventions (Refined as of 10-Sept-2024).xlsx

1. Conversational AI

1.1. Overview of Conversational AI

Conversational AI at FPL uses advanced technologies like AWS Lex and AWS Connect to create an intelligent, user-friendly customer interaction platform. This platform enables customers to report issues, request services, and get assistance through natural language interactions, either via voice or chat. The AI platform is integrated with various backend systems through AWS Lambda functions, ensuring data processing and efficient customer service.

1.2. Design Objectives and Goals

The design of the Convo AI system within the High-Volume Outage scenario has been guided by the following key goals to ensure scalability, maintainability, and operational efficiency

- Reusability and Modularity

System components—including Lambda functions, Lex bots, and Amazon Connect flows—are built with a modular architecture to promote reusability across multiple call flows and use cases.

Common assets such as language detection, intent handling, and authentication are abstracted into shared services or reusable components, reducing duplication and easing maintenance

- Efficient Prompt Management

Prompts used in voice interactions are managed as key-value pairs within a centralized Prompt DB, enabling dynamic retrieval based on flow context.

Updates to prompts follow an incremental insert pattern, ensuring that only the latest records are added to the database without the need to overwrite or replace the entire prompt table.

The database supports SSML (Speech Synthesis Markup Language) tags, allowing for rich, customizable voice experiences directly from stored prompt values

- Token Generation Modularity

Token generation logic is implemented as a modular component within Lambda functions and invoked during the initial steps of the interaction.

This modular approach ensures secure, consistent handling of session tokens and authentication workflows across different service flows and user journeys

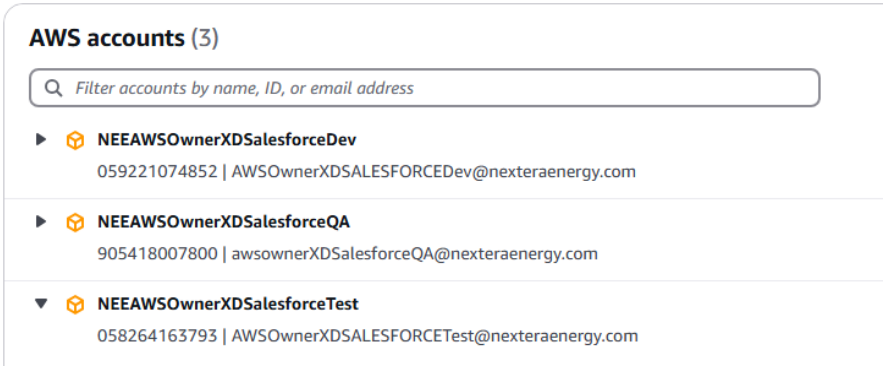
1.3. Key AWS Services

Following services have been used in the build

- Amazon Connect
- Amazon Lex
- AWS Lambda
- AWS DynamoDB
- AWS S3
- AWS Cloudwatch
- AWS Kinesis Data Streams

1.4. General Overview

There are 4 environments – Dev, Test, QA, Prod for the build



Language and channel –

Language-specific prompts are retrieved dynamically using PromptManagement in Connect flows using prompt management lambda listed in each section

Locale is managed via localeId attributes detected in welcome flow

Channel differentiation is managed through $.Channel, and dynamic prompts are used accordingly

Overview of Intents and important links

Intent	Business definition	Functional requirements link (confluence + JIRA)
Welcome & Authentication	Greeting users with universal welcome message & authenticating users to identify users based on their ANI match	https://confluence.nexteraenergy.com/pages/viewpage.action?pageId=664371700
Down Wire	Urgent escalation to agent during a down wire scenario	https://confluence.nexteraenergy.com/display/CISCAMS/4.+Module%3A+Down+wire
Emergency	Emergency escalation based on emergency hot words used by user	https://confluence.nexteraenergy.com/display/CISCAMS/5.+Module%3A+Emergency
Police & Fire	Routing user to Police or Fire based on the PIN entered	https://confluence.nexteraenergy.com/pages/viewpage.action?pageId=664371708
Report Outage	Process of reporting an outage & ticket creation	https://confluence.nexteraenergy.com/display/CISCAMS/7.+Module%3A+Report+Outage
Outage Status	User enquiring about their outage status based on their ticket (Active ticket, no ticket or closed ticket)	https://confluence.nexteraenergy.com/display/CISCAMS/8.+Module%3A+Outage+Status
No Input No Match	System to retry or route if there is no input or no match from user	https://confluence.nexteraenergy.com/display/CISCAMS/9.+Module%3A+No+Input+No+Match
Outage Load Control	Inform user about their On call program usage status	https://confluence.nexteraenergy.com/display/CISCAMS/10.+Module%3A+Outage+Load+Control

1.5. HVOS Module – Welcome

Overview of HVOS Flows/Lambdas/lex bots/API responses invocation

https://nee.sharepoint.com/:x:/r/teams/EXT_CIStoSAP-AWSConnect_Lex/_layouts/15/Doc.aspx?sourcedoc=%7B4B2BB728-6E13-460F-9C69-AF2EA82CD49C%7D&file=HVOS%20-%20Logical%20flow%20design.xlsx&action=default&mobileredirect=true

1.5.1. Flow– 2XSFNV-EI864-Outage-Initial-Flow

Description – The purpose of this flow is to land the customer's call into amazon connect and based on the dialed number route them to welcome message or storm status line

Channels - Chat and Voice

Language - English (Ruth) and Spanish (Lupe).

Attributes - localeId = en_US for English and es_US for Spanish

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/6667252f8469ef0007e9015e

Lex Bots Used –

- Global-UserDetails-LX
- Global-OrchestratorBot-LX
- Outage-IntentDetection-LX
- Global-Confirmation-LX
- Global-LanguageDetection-LX

Lambda Functions invoked –

2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary.

Attributes as inputs – localeId (default = en_US) , Domain, contactId. ExtractMultiBotArn (Default = true)

Global-GetUserAccountDetails-LF – for Fetch account details using phone/account

Global-fetch-admin-console-key-values-LF – for loading admin console related flags (rep options, load control, etc.) from admin console tables

Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

Attributes as inputs

List of attributes –

System / Dynamic:

- Customer endpoint address
- Dialed number
- localeId
- Channel
- ContactId

Contact Attributes

- Admin console Variable such as AC_HVCAMode, AC_WeatherEvent, AC_LoadControl etc.

Prompts Key-Values

- User defined variables - UD_PhoneNumber, UD_AccountNumber, UD_FirstName, etc.

Error Handling –

- Lex Bot Failures - All ConnectParticipantWithLexBot blocks include- NoMatchingCondition → fallback routing, NoMatchingError → fallback routing or disconnect
- Lambda Failures at invocation is attached to timeout fallbacks, invocation failures with retry and graceful disconnect blocks.
- Comparison Blocks - Every Compare block includes:
- Default path when no condition matches
- Fallbacks go to either a generic fallback disconnect or to alternate attribute-setting logic
- Disconnects are handled explicitly using DisconnectParticipant blocks for unhandled paths

1.6. HVOS Module – Storm Status Line

1.6.1. Flow– 2XSFNV-EI864-Outage-StormStatusLine-Flow

Description – Connect flow to inform customers of storm-related messages and confirm receipt of alerts for those who dialed storm status line

Channels – Voice only

Language - English (Ruth) and Spanish (Lupe) flows from Outage-Initial-Flow based on “localeId’ attribute

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/6733c6d1713507277e140745

Lex Bots Used - None

Lambda Functions invoked –

2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary

API or Database Invocations - None

List of attributes –

- $.ContactId
- localeId
- Error Handling - The entire flow has robust fallback handling ensuring a clean exit path in case of:
- No input - A replay of the message, Final disconnect if retries exhausted
- Invalid DTMF entries - A replay of the message, Final disconnect if retries exhausted

1.7. HVOS Module – HVCA Authentication

1.7.1. Flow– 2XSFNV-EI864-Outage-HVCAAuthentication-Flow

Description - This flow authenticates customers for HVCA scenario via ANI matching, last name/business name verification, or address, and routes based on authentication status

Channels – Voice Only

Language - English (Ruth) and Spanish (Lupe) flows from Outage-Initial-Flow based on “localeId’ attribute

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/67339a18713507277e13b4ec

Lex Bots Used –

- Outage-UserDetails-LX
- Global-OrchestratorBot-LX
- Outage-IntentDetection-LX
- Outage-Confirmation-LX

Lambda Functions invoked –

- Outage-HVCAAuthentication-LF - Main Lambda that hits the HVCA Authentication API for ANI matching and address lookup and further preprocessing
- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary

Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration

API or Database Invocations –

- HVCA Authentication API – Via Outage-HVCAAuthentication-LF
- Smart outage inquiry API – Via Outage-DynamicAPIHandler-LF

List of attributes –

Attributes Set by Flow:

- UD_IncomingIntent: HVCAAuthenticated, HVCAUnauthenticated, intent_Emergency, intent_Outage, etc.
- UD_IsAuthenticated: Yes or No
- Auth_*: customer info attributes like Auth_accountNumber, Auth_address, etc.

Attributes Used for Logic:

- UD_MultiAddressCheck, AC_HVCAMode, AC_UniversalStormToggle, AC_ASM, UD_TransferNoMatch

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviours. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes
- Failure to set attributes defaults to exit disconnect

1.7.2. Flow– 2XSFNV-EI864-Outage-HVCAAuthentication-UserDetails-Flow

Description - Authenticates HVCA customers by capturing either account number or phone number, verifies customer identity using Lambdas and Lex bots, and routes users accordingly based on match outcomes and attributes

Channels – Voice Only

Language - English (Ruth) and Spanish (Lupe) flows from Outage-Initial-Flow based on "localeId' attribute

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/67339a18713507277e13b4ec

Lex Bots Used –

- Outage-UserDetails-LX
- Global-OrchestratorBot-LX
- Outage-IntentDetection-LX
- Outage-Confirmation-LX

Lambda Functions invoked –

- Outage-HVCAAuthentication-LF - Main Lambda that hits the HVCA Authentication API for ANI matching and address lookup and further preprocessing
- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration

API or Database Invocations –

- HVCA Authentication API – Via Outage-HVCAAuthentication-LF
- Smart outage inquiry API – Via Outage-DynamicAPIHandler-LF

List of attributes –

Attributes Set by Flow:

- UD_IncomingIntent: HVCAAuthenticated, HVCAUnauthenticated, intent_Emergency, intent_Outage, etc.
- UD_IsAuthenticated: Yes or No for final auth status
- Auth_*: customer info attributes like Auth_accountNumber, Auth_address, etc.

Attributes Used for Logic:

- AC_HVCAMode, AC_UniversalStormToggle, AC_ASM, UD_TransferNoMatch
- AC_noMatch: Controls if unmatched paths should be transferred to agent
- UD_MultiAddressCheck, multiple-address-auth-retry-acct_num, etc

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes
- Failure to set attributes defaults to exit disconnect

1.7.3. Flow– 2XSFNV-EI864-Outage-HVCAAuthentication-IntentCheck-Flow

Description - Post-authentication logic to determine user intent (Outage, Emergency, Police/Fire) after successful HVCA authentication. This flow validates if the user has an open ticket and guides them accordingly, with escalation or exit routes.

Channels – Voice Only

Language - English (Ruth) and Spanish (Lupe) flows from Outage-Initial-Flow based on "localeId' attribute

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/67339a18713507277e13b4ec

Lex Bots Used –

- Outage-UserDetails-LX
- Global-OrchestratorBot-LX
- Outage-IntentDetection-LX
- Outage-Confirmation-LX

Lambda Functions invoked –

- Outage-HVCAAuthentication-LF - Main Lambda that hits the HVCA Authentication API for ANI matching and address lookup and further preprocessing
- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for service orchestration by getting ticket details or creating an outage ticket

APIs invoked - Smart Outage Inquiry, Ticket Submission API

API or Database Invocations –

- HVCA Authentication API – Via Outage-HVCAAuthentication-LF
- Smart Outage inquiry API – Via Outage-DynamicAPIHandler-LF
- Ticket Submission API – Via Outage-DynamicAPIHandler-LF

List of attributes –

Attributes Set by Flow:

- UD_IncomingIntent: HVCAAuthenticated, HVCAUnauthenticated, intent_Emergency, intent_Outage, etc.
- UD_IsAuthenticated: Yes or No for final auth status
- Auth_*: customer info attributes like Auth_accountNumber, Auth_address, etc.

Attributes Used for Logic:

- AC_HVCAMode, AC_UniversalStormToggle, AC_ASM, UD_TransferNoMatch
- AC_noMatch: Controls if unmatched paths should be transferred to agent
- SOI_ticketFoundFlag, SOI_universalStormToggle, SOI_ticketNumber, SOI_ticketStatus, SOI_ticketRestoredDate, SOI_meterStatus

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.7.4. Module– 2XSFNV-EI864-Outage-HVCAAuthentication-MultipleAddress-Mod

Description - Handles HVCA authentication when multiple addresses are associated with a contact. It verifies address input, evaluates confidence score, sets authentication status, and routes users accordingly.

Channels – Voice Only

Language - English (Ruth) and Spanish (Lupe) flows from Outage-Initial-Flow based on “localeId’ attribute

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/67339a18713507277e13b4ec

Lex Bots Used –

- Outage-UserDetails-LX
- Global-OrchestratorBot-LX
- Outage-IntentDetection-LX
- Outage-Confirmation-LX

Lambda Functions invoked –

- Outage-HVCAAuthentication-LF - Main Lambda that hits the HVCA Authentication API for ANI matching and address lookup and further preprocessing
- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration

API or Database Invocations –

- HVCA Authentication API – Via Outage-HVCAAuthentication-LF
- Smart outage inquiry API – Via Outage-DynamicAPIHandler-LF

List of attributes –

Attributes Set by Flow:

- UD_IncomingIntent: HVCAAuthenticated, HVCAUnauthenticated, intent_Emergency, intent_Outage, etc.
- UD_IsAuthenticated: Yes or No for final auth status
- Auth_*: customer info attributes like Auth_accountNumber, Auth_address, Auth_phoneNumber, UD_ConfidenceScore, Auth_confidenceScoreetc.

Attributes Used for Logic:

- AC_HVCAMode, AC_UniversalStormToggle, AC_ASM, UD_TransferNoMatch
- AC_noMatch: Controls if unmatched paths should be transferred to agent
- SOI_ticketFoundFlag, SOI_universalStormToggle, SOI_ticketNumber, SOI_ticketStatus, SOI_ticketRestoredDate, SOI_meterStatus

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.8. HVOS Module – Wiredown

1.8 Flow– 2XSFNV-EI864-Outage-Emergency-Flow

Description - This flow handles emergency reports from customers (e.g. downed wires), gathers context from Lex (slot_clearEmergency = downwire), configures contact attributes, and dynamically routes to the appropriate flow for follow-up or agent escalation (Outage-AgentTransfer-Flow for processing escalation for emergency or police/fire handling and then transfer to Outage-ExitIVR-Flow for graceful exits to agents based on customer journey)

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/668cabbb1ac5cfcab081e053

Lex Bots Used –

- Outage-UserDetails-LX
- Global-OrchestratorBot-LX
- Outage-IntentDetection-LX
- Determine emergency type a downwire using slot_clearEmergency
- Customize prompts
- Set routing logic
- Outage-Confirmation-LX
- Global-LanguageDetection-LX

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary

- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration
- Outage-LambdaHandler-LF – to fetch and validate customer inputs like phone numbers, retrieve dynamic prompts from dynamo db, manage fallback and DTMF-based intent resolution and counters to track the retries.

API or Database Invocations - None

List of attributes –

Lex Slot & Session Attributes:

- slot_ClearEmergency
- Custom Contact Attributes Set
- UD_IncomingIntent: Captures whether it's intent_Emergency or intent_EmergencyDownwire

Error Handling –

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block
- Compare Conditions (Routing Logic)
- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes
- Failure to set attributes defaults to exit disconnect

1.8.2 Flow– 2XSFNV-EI864-Outage-EmergencyLine-Policemen-Publix-Flow

Description - This flow handles routing of customer calls to a dedicated Cisco number. This flow is mapped to be dedicated Amazon Connect DID to receive the customer calls and further get transferred out to Cisco DID config

1.9. HVOS Module – Report an Outage and Outage flows

1.9.1. Flow– 2XSFNV-EI864-Outage-ReportTicket-Flow

Description - This flow allows customers to report outage-related issues such as voltage concerns, or full outages. It also handles collecting contact information, additional details, and submitting a formal ticket via Lambda

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/669499106bdc66a06f103bb6

Lex Bots Used –

- Outage-Confirmation-LX - Used to confirm ticket submissions, collect additional remarks, or escalate to agent
- Outage-IntentDetection-LX - Used in disambiguation and detect intent
- Outage-CollectContactNumber-LX – Used to collect contact information for the customer for report outage steps

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration. This is also used to submit the ticket to ADMS, submits outage tickets via OutageTicketSubmission, fetches ticket status for ETRs via smart outage inquiry API
- Outage-LambdaHandler-LF – to fetch and process customer inputs, validations and counters
- Outage-SendCustomerInteractions-LF - Records customer interaction metadata to UCDP

API or Database Invocations –

- ADMS API to submit the outage ticket to ADMS portal
- Smart Outage Inquiry to retrieve ETR related information post ticket submission

List of attributes –

User defined

- UD_LoadControlToReport, UD_PartialFullFlag, UD_CheckReport, UD_IncomingIntent, UD_ReportTicketStormFlag, UD_ReportTicketFlag

Downstream – SOI_* for smart outage inquiry attributes

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviours. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes
- Failure to set attributes defaults to exit disconnect

1.9.2. Flow– 2XSFNV-EI864-Outage-MainEntry-Flow

Description - This is the primary entry point for outage-related interactions. It handles disqualifications, localized prompt loading, HVCA logic, ticket status check, outage acknowledgment, and dynamic routing to report, troubleshoot, or exit paths

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/669499106bdc66a06f103bb6

Lex Bots Used –

- Outage-Confirmation-LX - Used to confirm ticket submissions, collect additional remarks, or escalate to agent
- Outage-IntentDetection-LX - Used in disambiguation and detect intent

Lambda Functions invoked –

- 2XSFNV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration. This is also used to submit the ticket to ADMS, submits outage tickets via OutageTicketSubmission, fetches ticket status for ETRs via smart outage inquiry API
- Outage-LambdaHandler-LF – to fetch and process customer inputs, validations and counters
- Outage-SendCustomerInteractions-LF - Records customer interaction metadata to UCDP

API or Database Invocations –

- Smart Outage Inquiry - to retrieve ticket related information before ticket submission

List of attributes –

User defined

- UD_LoadControlToReport, UD_PartialFullFlag, UD_CheckReport

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*,

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviours. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.9.3. Flow– 2XSFNV-EI864-Outage-Troubleshooting-Flow

Description - This flow determines if the customer's power issue can be resolved without an agent by running automated checks, confirming premise power status, offering circuit breaker resets, and guiding through appliance troubleshooting before deciding whether to escalate

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66918f34f7859d495f7b39c5

Lex Bots Used –

- Outage-Confirmation-LX - Used to confirm ticket submissions, collect additional remarks, or escalate to agent
- Outage-IntentDetection-LX - Used in disambiguation and detect intent

Lambda Functions invoked –

- 2XSFNV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration. This is also used to submit the ticket to ADMS, submits outage tickets via OutageTicketSubmission, fetches ticket status for ETRs via smart outage inquiry API
- Outage-LambdaHandler-LF – to fetch and process customer inputs, validations and counters
- Outage-SendCustomerInteractions-LF - Records customer interaction metadata to UCDP

API or Database Invocations –

Smart Outage Inquiry to retrieve ticket related information before ticket submission

List of attributes –

User defined

- UD_PartialFullFlag, UD_CheckReport, UD_ApplianceCheckflag, UD_OfferTroubleshooting

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*,

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviours. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.10. HVOS Module – Get Outage Status and Outage flows

1.10.1. Flow– 2XSFNV-EI864-Outage-ETREntry-Flow

Description - his flow is designed to gather or confirm Estimated Time of Restoration (ETR) information based on the customer's ticket, status, and authentication. It routes the user to the right follow-up interaction (confirmation, collection, iNotify)

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66ad0feb70dfd5f4b6c3320b

Lex Bots Used –

- Outage-Confirmation-LX - Used to confirm ticket submissions, collect additional remarks, or escalate to agent
- Outage-IntentDetection-LX - Used in disambiguation and detect intent

Lambda Functions invoked –

- 2XSFNV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration. This is also used to submit the ticket to ADMS, submits outage tickets via OutageTicketSubmission, fetches ticket status for ETRs via smart outage inquiry API
- Outage-LambdaHandler-LF – to fetch and process customer inputs, validations and counters
- Outage-SendCustomerInteractions-LF - Records customer interaction metadata to UCDP

API or Database Invocations –

Smart Outage Inquiry to retrieve ticket related information before ticket submission

List of attributes –

Used for Logic:

- AC_HVCAMode, SOI_universalStormToggle
- SOI_ticketFoundFlag, SOI_ticketPrevCallFoundFlag
- SOI_itrEtrType (A/N), SOI_ticketType, SOI_OutageTicketStatus
- UD_ETREntryFlag, UD_ETRTicketTypeCheckFlag

Set During Flow:

- UD_ETREntryTransferNoStorm
- UD_ETRTimeSubmitBackADMSTicket = Y
- UD_ETRTicketTypeCheckFlag = Y
- User_Utterance (captured from Lex for analytics)

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviours. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.10.2. Flow– 2XSFNV-EI864-Outage-ETRAdditionalCauseCodes-Flow

Description - Based on the customer's ticket cause code, this flow provides personalized messaging about outage causes (e.g., storm damage, animal contact, equipment failure). It supplements the ETR communication process with informative, relevant messages

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66ad6467ba2e6706b6635571

Lex Bots Used –

- Outage-Confirmation-LX - Used to confirm ticket submissions, collect additional remarks, or escalate to agent
- Outage-IntentDetection-LX - Used in disambiguation and detect intent

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

API or Database Invocations – None

List of attributes –

Used for Logic:

SOI_ticketCauseCode

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*

Error Handling –

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.10.3. Flow– 2XSFNV-EI864-Outage-ETRResume-Flow

Description - Resumes the Estimated Time of Restoration (ETR) interaction, determines if customer is eligible for iNotify updates, confirms their preferences, and either transfers them to ETRTime or disambiguates their needs

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66ad0feb70dfd5f4b6c3320b

Lex Bots Used –

Outage-Confirmation-LX - Used to confirm ticket submissions, collect additional remarks, or escalate to agent

Lambda Functions invoked –

- 2XSFNV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration. This is also used to submit the ticket to ADMS, submits outage tickets via OutageTicketSubmission, fetches ticket status for ETRs via smart outage inquiry API

API or Database Invocations – None

List of attributes –

Used for Logic:

- AC_HVCAMode, SOI_universalStormToggle
- SOI_ticketFoundFlag, SOI_ticketPrevCallFoundFlag
- SOI_itrEtrType (A/N), SOI_ticketType, SOI_OutageTicketStatus
- UD_ETREntryFlag, UD_ETRTicketTypeCheckFlag

Set During Flow:

- UD_ETREntryTransferNoStorm
- UD_ETRTimeSubmitBackADMSTicket = Y
- UD_ETRTicketTypeCheckFlag = Y
- UD_ETRTimeTransferEndMsgElseHVCA = Y

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*

1.10.4. Flow– 2XSFNV-EI864-Outage-ETRTicketTypeCheck-Flow

Description - This flow analyzes outage ticket types (SNC, TX, FRG, etc.) to determine the most accurate Estimated Time of Restoration (ETR) messaging path. It also sets reporting markers and conditionally invokes other flows such as ETR Logic or Flood messaging modules

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66ad14ab6602f99e6782b8e4

Lex Bots Used –

None

Lambda Functions invoked –

- 2XSFNV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

API or Database Invocations –

Smart Outage Inquiry to retrieve ticket related information before ticket submission

List of attributes –

Used for Logic:

- AC_HVCAMode, SOI_universalStormToggle
- SOI_ticketFoundFlag, SOI_ticketPrevCallFoundFlag
- SOI_itrEtrType (A/N), SOI_ticketType, SOI_OutageTicketStatus
- SOI_ticketType (e.g., TX, FRG, Green tkt, SNC, etc.)
- SOI_universalStormToggle

- SOI_ticketCauseCode = 195
- SOI_eventName = “”
- SOI_ticketCustomerEffected
- AC_WeatherEvent
- UD_ETRTicketTypeCheckFlag

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*

Error Handling –

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block
- Compare Conditions (Routing Logic)
- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.10.5. Flow– 2XSFNV-EI864-Outage-ETRTime-Flow

Description - Communicates Estimated Time of Restoration (ETR) based on ticket and weather data. It dynamically routes customers to the correct message or flow depending on ETR type, HVCA mode, storm status, and customer flags

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66b0f5817212dd6a4809d91b

Lex Bots Used –

None

Lambda Functions invoked –

- 2XSFNV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary

- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Submits back ETR ticket data to backend systems

API or Database Invocations –

ADMS to submit ETR ticket related information to ADMS portal

List of attributes –

Used for Routing:

- SOI_itrEtrType (A/N)
- SOI_universalStormToggle
- AC_HVCAMode, AC_WeatherEvent
- UD_ReportTicketFlag, UD_ETRTimeSubmitBackADMSTicket

Set:

- UD_ETREntryTransferNoStorm,
- UD_ETRTimeTransferEndMsgElseHVCA
- UD_WantsCallBack = false

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*

Error Handling –

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect
- API Submission Failures - Trigger prompt fallback: Outage_ETR_Time_API_Failure_prompt then transfer to Goodbye
- Fallback Routing-Includes disconnect or graceful flow exits (e.g., transfer to Goodbye)

1.10.6. Flow– 2XSFNV-EI864-Outage-ETRTypeLogic-Flow

Description - Determines the appropriate Estimated Time of Restoration (ETR) experience based on a variety of ticket attributes including type, status, next step, and storm toggle flags. Routes to one of several follow-up flows or modules based on logic

Channels - Chat and Voice. Detected using $.Channel with dedicated branches for each.

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66b106667212dd6a4809d92b

Lex Bots Used –

None

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

API or Database Invocations –

None

List of attributes –

Used for Logic:

- SOI_ticketFoundFlag, SOI_ticketRestoredDate, SOI_ticketReferredTime
- SOI_ticketNextStep, SOI_OutageTicketStatus
- SOI_itrEtrType (I = Interactive, E = Express)
- SOI_universalStormToggle

Downstream – SOI_* for smart outage inquiry attributes, Admin Console Attributes AC_*

Error Handling –

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect
- Default Routing - If no ticket found, storm toggle or other key value is undefined → fallback messages and clean disconnect to Goodbye flow

1.10.7 Flow-- 2XSFNV-EI864-Outage-ActiverecentTicket-Flow – Updated content to be added

1.11. HVOS Module – Load Control

1.11.1 Flow– 2XSFNV-EI864-Outage-LoadControl-Flow

Description - This flow informs customers about Load Control (LC) program activity, offers them options to learn more, report issues, or exit the call. It provides interactive messaging and routing based on dynamic session attributes and customer confirmation

Channels – Chat and Voice. Determined via the $.Channel attribute, Lex prompts and transitions are adapted accordingly

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/667c461c1904be4b6727768b

Lex Bots Used –

- Outage-Confirmation-LX - Used for Message confirmation (on-call, pause explanation, follow-up questions). Confirmation of reporting interest or further help

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary

- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration

API or Database Invocations –

- Smart outage inquiry via Outage-DynamicAPIHandler-LF
- HVCA Authentication API – Via Outage-HVCAAuthentication-LF (already called and attributes fetched at HVCA authentication step)

List of attributes – Auth_* from the HVCA authentication API, SOI_* from Smart outage inquiry.

Error Handling -

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.12. HVOS Module – Outage Modules (Common)

1.12.1. Module– 2XSFNV-EI864-Outage-AreaSpecificMessage-Mod

Description - Dynamically plays localized, area-specific messages for customers based on area code and language, fetched from an external Admin Console Lambda. Supports both voice and chat interactions.

Channels – Chat and Voice. Determined via the $.Channel attribute, Lex prompts and transitions are adapted accordingly

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/cms/workflow/667c461c1904be4b6727768b

Lex Bots Used - None

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Global-fetch-admin-console-key-values-LF – for loading admin console related flags (ASMs, rep options, load control, etc.) from admin console tables

API or Database Invocations –

Admin Console for ASMs

List of attributes –

- localeId, Channel, Auth_areaCode
- AC_areaSpecificMessage from Global-Fetch-Admin-Console-ASM-LF

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits

- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.12.2 Module– 2XSFNV-EI864-Outage-Outage-CollectContactNumber -Mod

Description - Handles messaging logic for customers who are disconnected or pending disconnection due to non-payment. Determines customer state using collection flags, delivers appropriate messaging, and gracefully disconnects.

Channels – Chat and Voice. Determined via the $.Channel attribute, Lex prompts and transitions are adapted accordingly

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66adccec51405b89fac4cedb

Lex Bots Used - None

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes
- Outage-DynamicAPIHandler-LF - Integrated dynamically via flow attributes; used for bot support, user lookup, and service orchestration

API or Database Invocations – None

List of attributes –

- localeId, Channel

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input

- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.12.3 Module– 2XSFNV-EI864-Outage-NoMatch-NoInput-Mod

Description - Handles no-match and no-input cases when Lex fails to understand or receive input from a customer. It uses various fallback strategies including retry attempts, storm-related handling, disambiguation, or transfers to other flows (e.g., agent)

Channels – Chat and Voice. Determined via the $.Channel attribute, Lex prompts and transitions are adapted accordingly

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/668fd4bdd6dc201b8cc0d202

Lex Bots Used –

- Outage-Confirmation-LX - Used for capturing simple yes/no from customer (follow-up questions). Confirmation of switching channel to SMS for further help
- Outage-IntentDetection-LX – Used to capture customer intent capture for outage intents applicable in HVOS scenario

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

API or Database Invocations – None

List of attributes –

- localeId, Channel, Auth_areaCode

Used in Logic:

- UD_IsAuthenticated (Note - UD_IsAuthenticated = No (if not already set during fallback to auth)
- AC_HVCAMode, AC_UniversalStormToggle
- UD_FurtherHelpFlag
- Lex slot: slot_YesNo
- Session attribute: NoInput

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with conditions like Lex.SessionAttributes.NoInput = True. Compares against storm toggles, auth flags (UD_IsAuthenticated), retry counters
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.12.4 Module– 2XSFNV-EI864-Outage-Outage-FloodMessage –Mod – Updated content to be added

1.12.5 Module-- 2XSFNV-EI864-Outage-LastKnownPowerStatus –Mod – Updated content to be added

1.13. HVOS Module – Agent Transfer, Exit IVR, Goodbye

1.13.1. Flow– 2XSFNV-EI864-Outage-AgentTransfer-Flow

Description - Customer calls are seamlessly transferred to the Exit IVR Flow or the Report Ticket Flow based on the specific flags or attributes identified during the call. This targeted approach ensures that each customer is directed to the most suitable resolution path, optimizing support efficiency and enhancing customer satisfaction

Channels - Chat / IVR

Language - English (en_US)/Spanish(es_US)

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66a6fa63fb2a3eb0c9833087

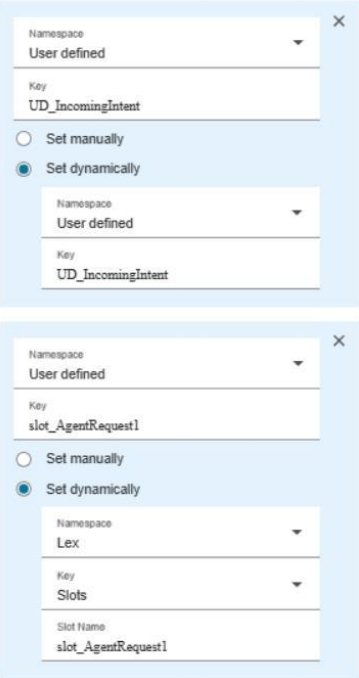
Lex Bots Used - NA

Lambda Functions invoked

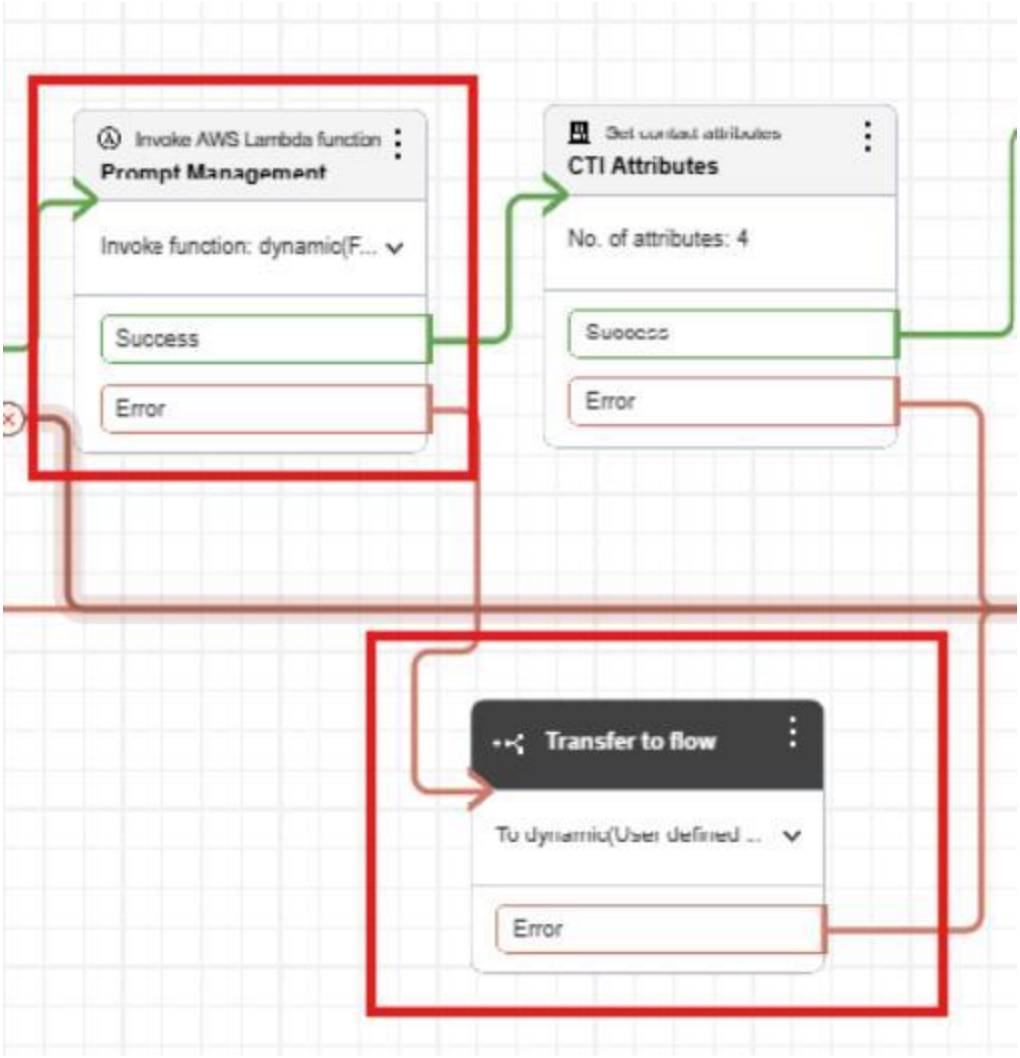
- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

API or Database Invocations – NA

List of attributes -



Error Handling - The 'Error Handling - Prompt Management Lambda' function is a critical component in maintaining system reliability by managing errors that occur within the prompt management process. In the event of a Lambda function failure, this function ensures continuity by seamlessly transferring the customer call to a live agent. This approach minimizes disruptions in customer interactions, ensuring that any issues are promptly addressed by a human representative, thereby enhancing overall customer satisfaction and service quality.



Important points to note:-Across all flows, English (en_US) and Spanish (es_US) are established as the common language options to cater to a diverse customer base. Additionally, the prompt management Lambda function serves as a universal component across these flows, ensuring consistent processing and management of prompts regardless of the language or specific flow.

1.13.2. Flow– 2XSFNV-EI864-Outage-ExitIVR-Flow

Description - This flow routes customers exiting IVR experiences (based on intent or status) to appropriate agent queues, based on attributes like region, language, and authentication state. It centralizes logic for clean exits from service modules like Load Control, Downwire, Emergency, etc..

Channels – Chat and Voice. Determined via the $.Channel attribute, Lex prompts and transitions are adapted accordingly

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - None

Lex Bots Used – None

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary

API or Database Invocations – None

List of attributes –

Key Attributes Used:

- AC_HVCAMode – To check HVCA Mode value
- UD_IncomingIntent → Defines why customer exited IVR. Example values are as follows
- intent_LoadControl
- intent_Emergency, intent_EmergencyDownwire
- intent_PoliceFire
- intent_OutageReportTicket
- intent_Agent
- intent_Ebill, etc.
- UD_IsAuthenticated → Used to branch to authenticated queues
- Auth_accountRegion → Used to differentiate FPL / FPLNW customers
- localeId → Determines language path

Error Handling –

- Final Exit Fallbacks

  Includes English and Spanish-specific NoMatch routes for:

  - HVOS vs non-HVOS
  - NW vs Legacy
  - With or without ANI
- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input

- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.13.3. Flow– 2XSFNV-EI864-Outage-GoodBye-Flow

Description - A short finalization flow used to gracefully end a customer interaction after a successful customer experience in the Contact centre post self service

Channels – Chat and Voice. Determined via the $.Channel attribute, Lex prompts and transitions are adapted accordingly

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/66a7ab81fb2a3eb0c98330bf

Lex Bots Used – None

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

API or Database Invocations – None

List of attributes –

- UD_MainToWelcomeFlag = N this is used to control re-routing bak to welcome flow if required

- localeId, Channel flow from downstream flows like welcome flow

Error Handling –

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.13.4. Flow– 2XSFNV-EI864-Outage-NoMatch-NoInput-Flow

Description - Handles no-match and no-input cases when Lex fails to understand or receive input from a customer. It uses various fallback strategies including retry attempts, storm-related handling, disambiguation, or transfers to other flows (e.g., agent)

Channels – Chat and Voice. Determined via the $.Channel attribute, Lex prompts and transitions are adapted accordingly

Language - English and Spanish. The flow uses dynamic language configuration via localeId and dynamically passed Lex attributes. Language is assumed to be preset via previous flow or detected upstream

Voiceflow - https://creator.voiceflow.com/project/6667252ef1a5985517ba4167/canvas/668fd4bdd6dc201b8cc0d202

Lex Bots Used –

- Outage-Confirmation-LX - Used for capturing simple yes/no from customer (follow-up questions). Confirmation of switching channel to SMS for further help
- Outage-IntentDetection-LX – Used to capture customer intent capture for outage intents applicable in HVOS scenario

Lambda Functions invoked –

- 2XSFTV-EI864-CFN-Resource-Search-LF – For Lookup resource names (voice/chat) from resource table in dynamo DB. The key-value pairs fetched are part of contact attributes dictionary
- Outage-PromptManagement-DBQuery-LF- to fetch prompt key-values pairs are insert them into contact attributes

API or Database Invocations – None

List of attributes –

- localeId, Channel, Auth_areaCode

Used in Logic:

- UD_IsAuthenticated (Note - UD_IsAuthenticated = No (if not already set during fallback to auth)
- AC_HVCAMode, AC_UniversalStormToggle
- UD_FurtherHelpFlag
- Lex slot: slot_YesNo
- Session attribute: NoInput

Error Handling –

- No Match / No Input and retries have been enabled at lex and lambda for retry behaviors. All flow has been enabled with conditions like Lex.SessionAttributes.NoInput = True. Compares against storm toggles, auth flags (UD_IsAuthenticated), retry counters
- If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue
- No of retries enables here is once at each attempt, as per design

Lambda Blocks

- Each InvokeLambdaFunction has a fallback path for NoMatchingError → routes to a DisconnectParticipant block

Compare Conditions (Routing Logic)

- NoMatchingCondition paths default to disconnects or graceful exits
- UpdateContactAttributes / FlowAttributes - Failure to set attributes defaults to exit disconnect

1.13.5. Flow– 2XSFNV-EI864-Outage-DirectTransferCaseCisco –Flow – Updated content to be added

1.13.6. Flow-- 2XSFNV-EI864-Outage-DirectTransferCaseGlitch –Flow – Updated content to be added

1.14. Flow– HVOS Foundational – Lex Bots

General Configurations

- Environments – Dev, Test, QA, Prod

- Aliases
    - English - <Environment>_English (e.g. Dev_English)
    - Spanish - <Environment>_Spanish (e.g. Dev_Spanish)
- Languages
    - English – en_US (Ruth) attached to English alias
    - Spanish – es_US (Lupe) attached to Spanish alias
- List of Bots

Outage-Confirmation-LX

1.14.1 Lex Bot– 2XSFNV-EI864-Outage-Confirmation-LX

Description – The purpose of this bot is to capture the confirmation type utterances from the customer at intent detection step for simple questions which are not asked at the slot level in the Amazon Connect flows. E.g. “Is your property completely without power?” in 2XSFNV-EI864-Outage-MainEntry-Flow in Amazon Connect. The customer response such as Yes/No etc. is invoked at intent level.

Channels - Chat / IVR

Language and Alias –

- English (en_US) – Dev_English
- Spanish (es_US) – Dev_Spanish

Lambda Functions invoked - No Lambda hooks present

Error Handling – Any No Match or No Input is sent to FallbackIntent. Each slot is also enabled with No match No input prompts at slot elicitation step

Intents and Slots

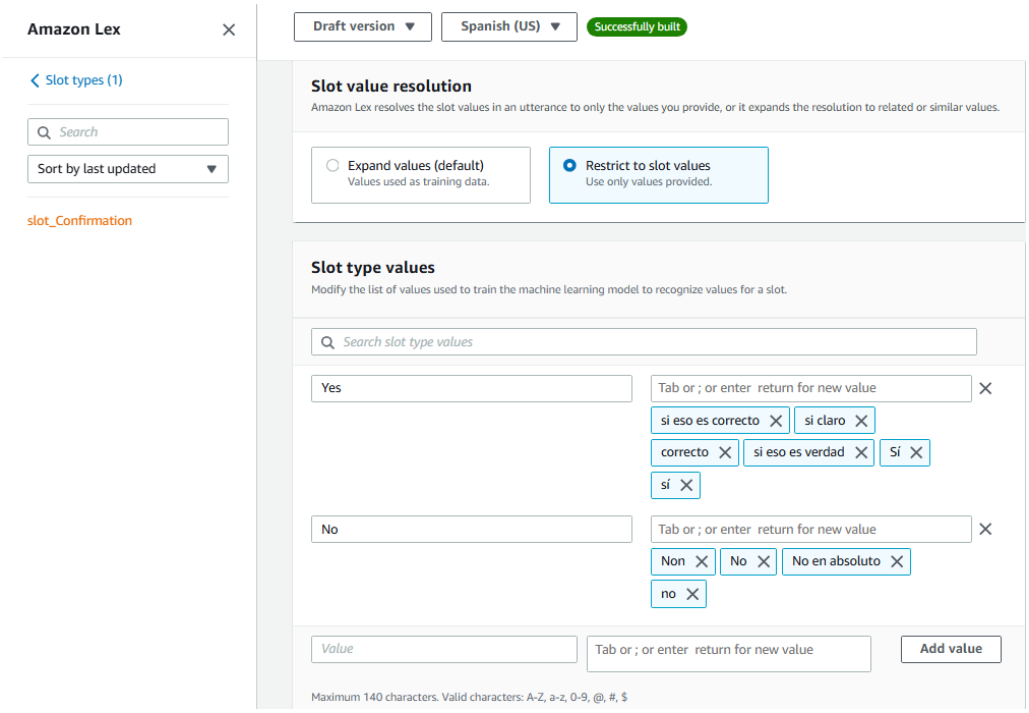
- Intents
    - intent_Confirmation
    - Sample Utterances – {slot_Confirmation} example “yes”, “No” etc.
    - FallbackIntent
    - Sample Utterances - No sample utterances defined — this is expected for fallback handling
- Slots

slot_Confirmation

- Slot Type –

AMAZON.Confirmation for English

For Spanish - Custom slot



1.14.2 Lex Bot– 2XSFNV-EI864-Outage-UserDetails-LX

Description – The purpose of this bot is to capture the information required for HVCA Authentication Module. E.g. “Please provide your account number” in 2XSFNV-EI864-Outage-HVCAAuthentication-Flow in Amazon Connect.

Channels - Chat / IVR

Language and Alias –

- English (en_US) – Dev_English
- Spanish (es_US) – Dev_Spanish

Lambda Functions invoked –

- 2XSFNV-EI864-Outage-UserDetails-LF – This lambda function is attached to do validation functions like invalid digits in phone number, account number, number of attempts to invalid phone/account number etc.
- Error Handling – Any No Match or No Input is sent to FallbackIntent. Each slot is also enabled with No match No input prompts at slot elicitation step

Intents and Slots

- Intents
  - intent_CaptureAddress – To Capture customer’s address
  - Sample Utterances – my address is {slot_CaptureAddress}.

- intent_CaptureCustomerPhoneNumber – To Capture customer's phone number and also capture if customer wants to provide phone number when the account number versus phone number preference is asked
- Sample Utterances - phone number is {slot_CaptureCustomerPhoneNumber}
- intent_CaptureAccountNumber- To Capture customer's account number and also capture if customer wants to provide account number when the account number versus phone number preference is asked
- Sample Utterances - my account number is {slot_CaptureAccountNumber}
- FallbackIntent
- Sample Utterances - No sample utterances defined — this is expected for fallback handling

- Slots
  - slot_CaptureAddress
  - Slot Type – custom Slot
  - slot_CaptureCustomerPhoneNumber
  - Slot Type - AMAZON.PhoneNumber
  - slot_CaptureAccountNumber
  - Slot Type - AMAZON.Number

1.14.3 Lex Bot– 2XSFNV-EI864-Outage-IntentDetection-LX

Description – The purpose of this bot is to identify the main intent to be triggered in Outage (e.g. Report outage, wire down get outage status, police fire etc.). It is triggered at multiple places such has welcome flow, HVCA authentication (No ANI).

Channels - Chat / IVR

Language and Alias –

- English (en_US) – Dev_English
- Spanish (es_US) – Dev_Spanish

Lambda Functions invoked –

- 2XSFNV-EI864-Outage-LambdaHandler-LF – The purpose of this lambda is to assist this bot with validations and call the prompt management for prompts.

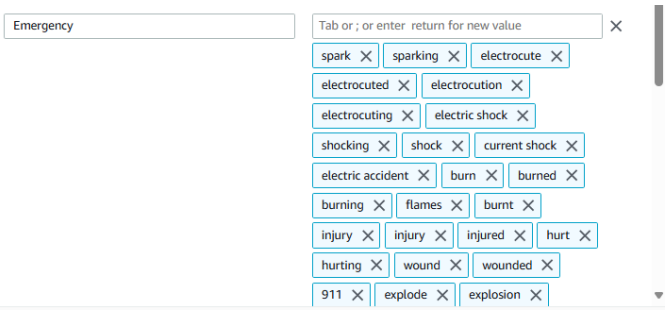
Error Handling – Any No Match or No Input is sent to FallbackIntent. Each slot is also enabled with No match No input prompts at slot elicitation step

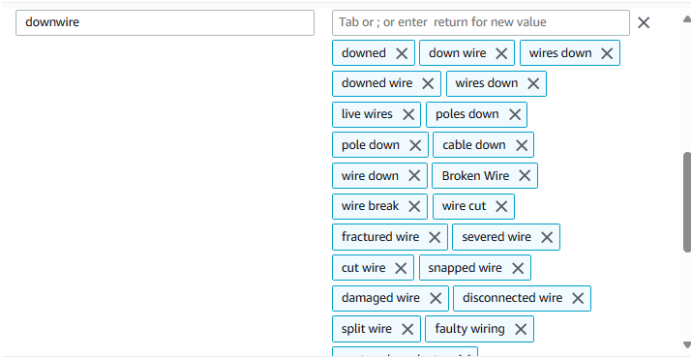
Intents and Slots

- Intents

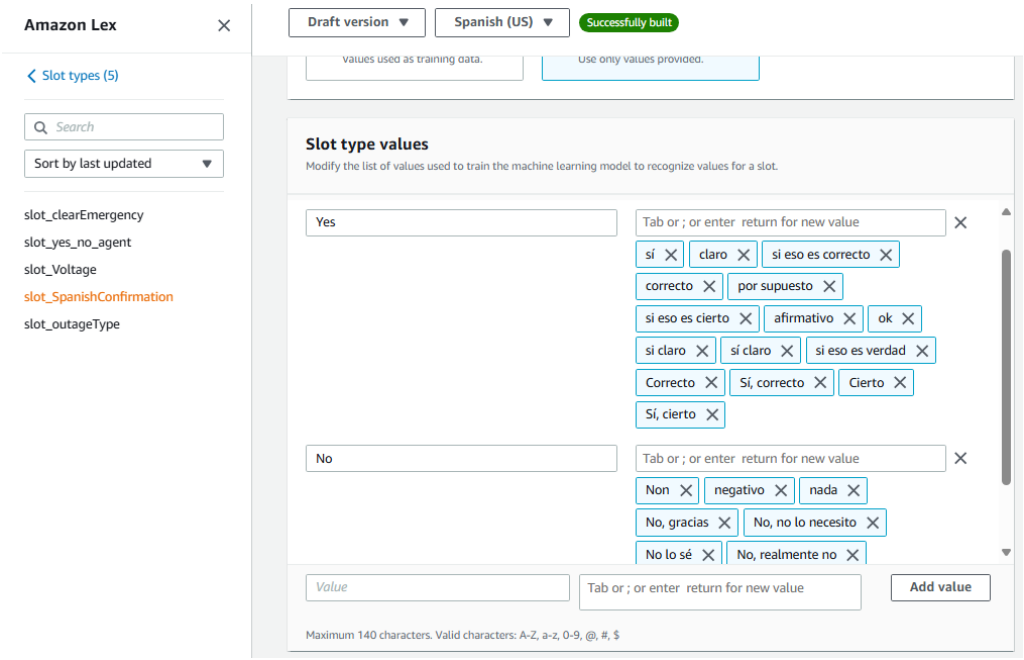
- intent_Outage – To capture intents related to report outage and get outage status
- Sample Utterances – Power outage, I have no power, I need to report an outage, what is the status of outage, status of my outage .
- intent_YesNo – to capture the intent when the first utterance is “Yes or No” and certain flags are sent from connect to internally conditional switch them to other intents based on flags sent
- Sample Utterances – {slot_YesNo}
- intent_LoadControl – to capture load control or on call related utterances
- Sample utterances - The AC in my house is not receiving electricity
- intent_Agent – to capture utterances related to customer requesting an agent
- Sample Utterances - I need to speak with a representative
- intent_PoliceFire – to capture police fire PINs entered using DTMF
- Sample Utterances – 711, 777
- intent_Emergency – to capture emergency related utterances including wiredown. Wiredown will be capture as a slot in slot_clearEmergency = wiredown.
- Sample utterances - {slot_ClearEmergency} on my street (Note – here {slot_ClearEmergency} can be equal to “wiredown”)
- FallbackIntent Sample Utterances - No sample utterances defined — this is expected for fallback handling

- Slots
  - slot_ClearEmergency
  - Slot Type – custom Slot
  - Possible values – Wiredown, Emergency (attached screenshot)





- Slot_YesNo
- Slot Type
- For English – AMAZON.Confirmation
- For Spanish – Custom Slot called slot_SpanishConfirmation



1.14.4 Lex Bot– 2XSFNV-EI864-Outage-CollectContactNumber-LX

Description – The purpose of this bot is to capture the contact number of the customer for the in report ticket or ETR entry flows. It is also enabled with validation of the entered number by the customer

Channels - Chat / IVR

Language and Alias –

- English (en_US) – Dev_English
- Spanish (es_US) – Dev_Spanish

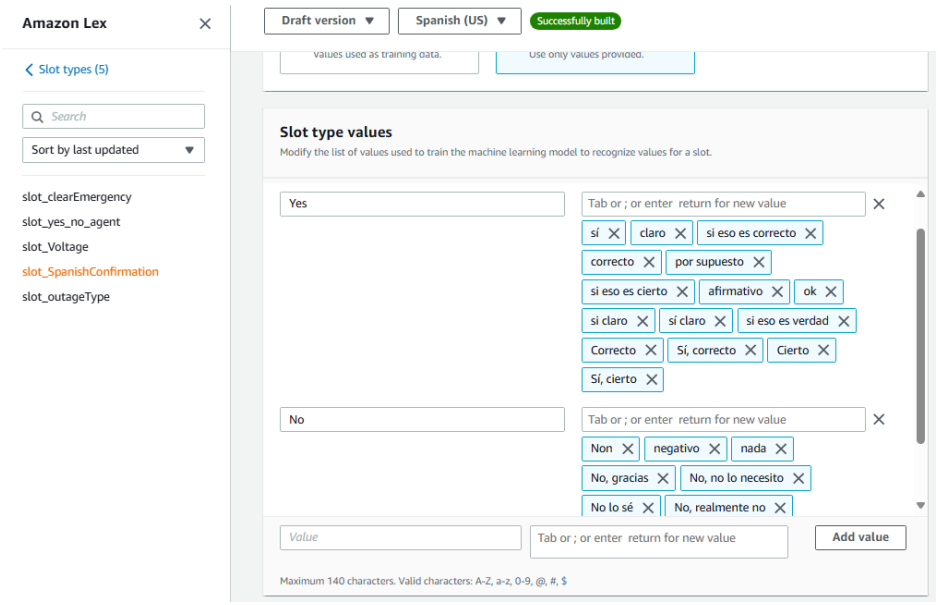
Lambda Functions invoked –

- 2XSFNV-EI864-Outage-LambdaHandler-LF – The purpose of this lambda is to assist this bot with validations and call the prompt management for prompts

Error Handling – Any No Match or No Input is sent to FallbackIntent. Each slot is also enabled with No match No input prompts at slot elicitation step

Intents and Slots

- Intents
  - intent_CollectContactNumber
  - Sample Utterances – The number is {slot_ContactNumber}
  - FallbackIntent
  - Sample Utterances - No sample utterances defined — this is expected for fallback handling
- Slots
  - slot_ContactNumber – To capture customer's entered phone number
  - Slot Type – AMAZON.PhoneNumber
  - Slot_confirmPhoneNumber
  - Slot Type – AMAZON.Confirmation for English and custom slot for Spanish (slot_SpanishConfirmation)
  - Slot_ValidNumberCheck – Ask the customer if the customer has a valid phone number to provide
  - Slot Type – AMAZON.Confirmation for English and custom slot for Spanish (slot_SpanishConfirmation)



1.15. HVOS Module – Lambdas

1.15.1. Lambda – 2XSFNV-EI864-Outage-DynamicAPIHandler-LF

Description:

- This AWS Lambda function integrates with Mulesoft APIs to handle outage inquiries and ticket submissions.
- It processes customer input from Amazon Connect, formats API responses, and supports fallback using mock data.
- The function ensures smooth execution with structured logging, error handling, and response cleanup.

Key Components

Imports:

- json – Used to encode and decode JSON data.
- os – Provides access to environment variables.
- logger_utils – Custom module for setting up logging and SDK log suppression.
- api_client – Custom module to initialize the API client for Mulesoft.
- api_utils – Custom utilities for API calls, secret management, JSON cleanup, and error handling.
- datetime – Provides tools for working with dates and times.
- mock_responses – Custom module providing mock data for outage inquiry and ticket submission.

Functions:

- outage_inquiry – Handles the outage inquiry API call, processes the response, and applies formatting.
- ticket_submission – Handles the outage ticket submission API call with input normalization and error fallback.
- lambda_handler – Main Lambda entry point that routes requests to the appropriate API function based on flow name.

Utility Functions:

- format_date_fields – Formats specific datetime fields in the API response into readable formats.
- fill_default_fields – Fills empty or missing fields in the response with default placeholder values.

Lambda Configurations & Environment Variables:

- Memory: 1024 MB
- Ephemeral Storage: 512 MB

- Lambda Timeout: 7 secs

Variable Name	Values	Description
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument	Used for enabling the X-Ray Trace for Lambda
LOG_LEVEL_VALUE	DEBUG	Used to set the log level
REGION_NAME	us-east-1	AWS region for resource access
REQUEST_TIMEOUT	3	Timeout for API requests (seconds)
SECRET_NAME	2xsfnv-ei864-mulesoft-token	Secret name for Mulesoft API token
SECRET_SESSION_SERVICE	secretsmanager	AWS Secrets Manager service
SSM_NAME	ei864/api/mulesoft/contact-center/host/stage	Parameter Store key for host URL
USE_MOCK_DATA	True	Toggles use of mock data on failure
http_proxy	http://webproxyeva.fpl.com:8080	Proxy for outbound HTTP requests
https_proxy	https://webproxyeva.fpl.com:8080	Proxy for outbound HTTPS requests
no_proxy	*.fpl.com,*.neeaws.local	Bypass proxy for internal domains

Custom Modules Used

Custom Module	Role
logger_utils	Sets up structured logging, suppresses SDK logs, adds contact ID context.
api_client	Initializes and manages the Mulesoft API client connection.
api_utils	Handles API endpoint calls, token retrieval, JSON formatting, and custom errors.

Functional Logic

- Initializes logger and attaches the contact ID for traceability in logs.
- Retrieves Mulesoft host URL from AWS SSM and fetches a secure token from Secrets Manager.
- Extracts the flow name (UD_FlowName) from Amazon Connect input to determine the requested operation.
- Based on the flow name, calls either the outage inquiry or ticket submission API via Mulesoft.
- Formats and enriches the API response (e.g., date fields, weather conditions, missing values).

- If the API response indicates an error, raises a custom exception (MulesoftAPIError).
- If mock mode is enabled (USE_MOCK_DATA=True) and an error occurs, returns predefined mock data.
- On success, cleans and flattens the response JSON structure for easy consumption by downstream systems.
- If an unhandled exception occurs, logs the error and returns a formatted error response.

Sample Event (Input)


```
{
  "Details": {
    "Parameters": {
      "UD_FlowName": "OutageInquiry",
      "channel": "IVR",
      "zzSubCode": "1600",
      "premiseNumber": "",
      "accountNumber": "2113267716",
      "localeId":"es_US"
    },
    "ContactData": {
      "InstanceARN": "arn:aws:connect:us-east-1:059221074852:instance
      "ContactId": "4ca79319-fdee-4cf5-ac1d-53af8fa3d0d3"
    }
  }
}
```


Sample Output

```
{
  "source": "AWS Connect",
  "channel": "IVR",
  "correlationId": "sdfsdf2134",
  "eventName": "outage/inquiry",
  "code": "200",
  "message": "SUCCESS",
  "outageCallFilterFlag": "Y",
  "universalStormToggle": "N",
  "weatherConditionType": "Hurricane",
  "weatherEvent": "Storm",
  "floodMessage": "Y",
  "ticketCustomerEffected": "1",
  "ticketMeterFlag": "Y",
  "ticketStatus": "arrived",
  "ticketCompletedDate": "NaN",
  "ticketType": "SNC",
  "ticketRestoredDate": "02/13/2024 22:07:22",
  "ticketCreateDate": "02/13/2024 22:07:22",
  "ticketFoundFlag": "N",
  "ticketNumber": "123"
```


Error Handling

- All main logic in lambda_handler is wrapped in a try-except block to catch unexpected errors.
- Logs detailed error messages using the custom logger for easier debugging.
- If an API returns an error, a custom MulesoftAPIError is raised to signal critical failure.
- When mock mode is enabled (USE_MOCK_DATA=True), mock responses are returned on exceptions.
- Differentiates between Mulesoft-specific errors and general exceptions for targeted handling.
- Returns a structured 500 error response when the issue is not mock-recoverable.

Logging and Debugging

- Appropriate levels of logging DEBUG / ERROR / INFO are incorporated in the code in wherever necessary to facilitate in debugging.
- Custom Module logger_utils.py has been defined and used in the lambda to enable the ContactId to be visible in the logs to make it easier to tie the lambda execution logs back to the actual AWS Connect invocation.

1.15.2. Lambda– 2XSFNV-EI864-Global-PromptManagementDBQuery-LF

Description:

- This Lambda function retrieves customer service prompts from DynamoDB based on input source (Amazon Lex or Connect).
- It formats the prompts by replacing placeholders, trims extra spaces, and returns them in a structured response.
- Supports language/channel selection, active status filtering for supporting admin console functionality, and logs context-specific debugging details.

Key Components

Imports

- os – Accesses environment variables like table names.
- re – Supports placeholder replacement using regular expressions.
- sys – Provides system-specific parameters (unused in this script).
- logger_utils – Custom module for structured logging and SDK log suppression.
- boto3 – AWS SDK used to access DynamoDB and Connect.
- boto3.dynamodb.conditions.Key – Used for querying DynamoDB indexes using key conditions.

Functions

- retrieve_prompts_by_channel_and_intent – Fetches prompt entries from DynamoDB based on workflow or intent.
- format_lex_result – Formats prompt data for Lex with support for placeholders and variations.
- format_connect_result – Formats prompt data for Connect with placeholder replacement support.
- lambda_handler – Main Lambda entry point that detects source type, fetches prompts, formats them, and returns the result.

Utility Functions

- replace_placeholder – Replaces template placeholders with actual values from session/contact attributes.
- trim_trailing_spaces – Recursively trims trailing spaces from keys and values in nested data.
- clean_prompts – Disables playback for prompts marked inactive by clearing prompt content.

Lambda Configurations & Environment Variables:

- Memory: 1024 MB
- Ephemeral Storage: 512 MB

- Lambda Timeout: 7 secs

Variable Name	Values	Description
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument	Used for enabling the X-Ray Trace for Lambda
LOG_LEVEL_VALUE	DEBUG	Used to set the log level
TABLE_NAME	2XSFNV-EI864-Admin-Console-Gen-Prompts-Table	Used to specify name of the DynamoDB Table

Custom Modules Used

Custom Module	Role
logger_utils.py	Sets up logging, suppresses SDK debug logs, adds contact ID filter

Functional Logic

- Initializes the logger and attaches contact ID for contextual logging.
- Connects to DynamoDB using a table name from environment variables.
- Determines if the event source is Lex or Connect based on event structure.
- Queries DynamoDB to fetch prompts using a GSI based on workflow or temporary intent.
- Filters out inactive prompts to prevent them from being played in IVR or chat.
- Formats prompts by selecting the right language and channel (e.g., English_Chat, Spanish_Voice).
- Replaces placeholders in prompt messages using session or contact data.
- Trims trailing spaces from all formatted prompts for cleaner output.
- Returns the final formatted result as the Lambda response.

Sample Event (Input)

```
{
  "Details": {
    "Parameters": {
      "Channel": "VOICE",
      "PQ_Timing": "10 minutes",
      "ItrEtrDateTime": "October 2nd, 7:30 PM",
      "SOI_ticketCustomerEffected":"10",
      "Auth_lastName":"Doey",
      "localeId": "en_US",
      "Workflow": "Outage_HVCA_Auth_Main",
      "Block": "Report_Outage",
      "Domain": "Outage",
      "UD_ReplacePlaceholderFlag":"true"
    },
    "ContactData": {
      "InstanceARN": "arn:aws:connect:us-east-1:059221074852:instance
      "ContactId": "4ca79319-fdee-4cf5-ac1d-53af8fa3d0d3"
    }
  }
}
```


Sample Output


```
{
  "statusCode": 200,
  "body": {
    "Outage_Outage_HVCA_Auth_Main_Last_name_read_aloud_acct": "Is the last name on the account Doey?",
    "Outage_Outage_HVCA_Auth_Main_Confirm_last_name": "Is the last name on the account Doey?",
    "Outage_Outage_HVCA_Auth_Main_Unable_to_assist_HVCA": "Due to a high volume of calls, we're currently focused on assisting with outage-rela
    not able to handle other inquiries at this time. If you're experiencing an outage and haven't already reported it, please press 1. For an
    press 2.",
    "Outage_Outage_HVCA_Auth_Main_Read_service_address_phone": "<speak> Does your service address start with <say-as interpret-as=\"digits\">{A
    say-as>?</speak>",
    "Outage_Outage_HVCA_Auth_Main_Goodbye_ani_match": "Sorry, we are unable to assist you right now. Please call again later.  As a reminder,
    threatening emergency, please call 911 immediately.",
    "Outage_Outage_HVCA_Auth_Main_No_Match_Auth_HVCA_2": "Please select one of the following options:\nPress 1 to report an outage.\nPress 2 to
    update.\nPress 3 to report a downed wire.",
    "Outage_Outage_HVCA_Auth_Main_First_eight_business_name_letters": "Are the first eight letters of your business name {Auth_businessNameStar
    "Outage_Outage_HVCA_Auth_Main_Confirm_business_name": "Is the business name on the account {Auth_businessName}?",
```


Error Handling

- Wraps the entire lambda_handler logic in a try-except block to catch unexpected errors.
- Logs any exceptions encountered during execution for easier debugging and traceability.
- Returns no fallback response or mock data, just logs the error (can be extended for better fault tolerance).
- Ensures prompt fetching and formatting failures do not crash the Lambda silently.

Logging and Debugging

- Appropriate levels of logging DEBUG / ERROR / INFO are incorporated in the code in wherever necessary to facilitate in debugging.
- Custom Module logger_utils.py has been defined and used in the lambda to enable the ContactId to be visible in the logs to make it easier to tie the lambda execution logs back to the actual AWS Connect invocation.

1.15.3. Lambda – 2XSFNV-EI864-Outage-HVCAAuthentication-LF

Description:

- This Lambda validates customers for HVCA interactions using account number, phone number, ANI, or address.
- It retrieves customer information via API or mock data, applies fuzzy matching for address validation, evaluates property details, and returns structured authentication results for downstream Connect flows.
- Results are cleaned, formatted, and returned for use in contact flows, with optional placeholder updates.

Key Components

Imports:

- json – Used for parsing and serializing JSON data.
- os – Accesses environment variables such as mock mode.
- re – Supports regex operations like pattern matching in addresses.
- logger_utils – Custom module for setting up structured logging.
- datetime – Handles date and time parsing and formatting.
- fuzzywuzzy.fuzz – Computes fuzzy string matching scores for address comparison.
- boto3 – AWS SDK to interact with services like Amazon Connect.
- mock_responses – Supplies mock user data for fallback handling.
- api_client – Custom client to communicate with Mulesoft APIs.
- api_utils – Provides helper functions for API calls, token management, JSON processing, and error handling.

Functions:

- get_mock_user_info – Finds a matching mock user by phone or account number.
- words_to_numbers – Converts number words (e.g., "one") to digits for fuzzy matching.
- check_address_for_close_match – Uses fuzzy matching to score and match user addresses.
- check_num_of_props_and_validate – Analyzes user details and returns matching logic for validation.
- process_addresses – Concatenates and formats address components for each property.
- handle_customer_data – Core handler for retrieving and validating user data.
- handle_ani – Handler for ANI-based authentication.
- handle_match_phone_number – Handler for phone number authentication.
- handle_match_account_number – Handler for account number authentication.
- handle_match_address – Handler for address-based matching using fuzzy logic.
- format_total_lmis – Formats LMIS value as currency if available.
- process_common – Consolidates shared processing logic across auth types.
- build_multiple_prop_res – Constructs response when customer has multiple properties.
- build_single_prop_res – Constructs response when customer has a single property.
- lambda_handler – Main entry point that routes based on the auth_type parameter and returns formatted results.

Utility Functions:

- get_appliance_info – Extracts appliance-related info from user data.
- process_output_element – Extracts and formats demand info and appliance names from a record.
- process_appliances – Converts appliance codes into human-readable names.
- process_demand_info – Finds appliance time windows and demand status.
- replace_connect_placeholders – Replaces placeholders in Connect attributes with actual values.
- final_response – Wraps a response dictionary in standard Lambda format.
- unnest_and_clean – Cleans and flattens nested user detail structures.
- separate_letters_from_word – Separates characters in a name for phonetic matching.
- extract_area_code – Extracts the area code from a phone number using regex.
- get_customer_data – Retrieves customer info from Mulesoft or falls back to mock data.
- map_channel – Maps input channel names to required Mulesoft-compatible values.
- build_query_params – Creates query params based on phone or account number.
- fallback_customer_data – Returns mock data when API call fails and mock mode is enabled.
- extract_address_start – Extracts street/house number from address strings.
- extract_numeric_part – Extracts numeric portion at start of an address.
- concatenate_address – Combines street, city, county, and postal code into a full address string.

Lambda Configurations & Environment Variables:

- Memory: 1024 MB
- Ephemeral Storage: 512 MB
- Lambda Timeout: 7 secs

Variable Name	Values	Description
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument	Used for enabling the X-Ray Trace for Lambda
LOG_LEVEL_VALUE	DEBUG	Used to set the log level
REGION_NAME	us-east-1	AWS region for resource access
REQUEST_TIMEOUT	3	Timeout for API requests (seconds)
SECRET_NAME	2xsfnv-ei864-mulesoft-token	Secret name for Mulesoft API token
SECRET_SESSION_SERVICE	secretsmanager	AWS Secrets Manager service
SSM_NAME	ei864/api/mulesoft/contact-center/host/stage	Parameter Store key for host URL
USE_MOCK_DATA	True	Toggles use of mock data on failure
http_proxy	http://webproxyeva.fpl.com:8080	Proxy for outbound HTTP requests
https_proxy	https://webproxyeva.fpl.com:8080	Proxy for outbound HTTPS requests
no_proxy	*.fpl.com,*.neeaws.local	Bypass proxy for internal domains

Custom Modules Used

Custom Module	Role
logger_utils	Sets up structured logging, suppresses SDK logs, adds contact ID context.
api_client	Initializes and manages the Mulesoft API client connection.
api_utils	Handles API endpoint calls, token retrieval, JSON formatting, and custom errors.

Authentication Strategy & Supported Auth Modes

Validation Scenario	Matching Logic
Address	Uses fuzzy matching based on cleaned, numeric address High: ≥75

	Medium: 35–74 Low: <35
Account Number	Must be 10 digits, numeric only
Phone Number	Must be 10 digits, numeric
Business/Last Name	If no address match, use business name or last name for verification

Attributes to Connect

Placeholder Key	Source Value
Auth_addressStart	Start of address from match
Auth_lastName, _Start	From customer name
LC_amount	totalLmis credit value
LC_loadcontroldevices	On-call appliances
LC_startTime/_endTime	Demand usage windows

Functional Logic

- Initializes logger and extracts contact ID for context-based logging.
- Parses input parameters to determine authentication method and locale.
- Retrieves Mulesoft host and secret token for API authentication.
- Handles four types of authentication: ANI, phone number, account number, or address.
- Fetches customer details via API or returns mock data if USE_MOCK_DATA is enabled.
- Processes appliance info, validates customer based on number of properties.
- For address match, uses fuzzy logic to determine confidence level of match.
- Formats LMIS and address details, builds response payload based on match type.
- Optionally replaces placeholders in Connect attributes using response data.
- Returns cleaned, structured output compatible with Amazon Connect contact flows.

Sample Event (Input)

```
    "RelatedContactId": null,
    "SegmentAttributes": {
      "connect:Subtype": {
        "ValueInteger": null,
        "ValueList": null,
        "ValueMap": null,
        "ValueString": "connect:Telephony"
      }
    },
    "SystemEndpoint": {
      "Address": "+17795484542",
      "Type": "TELEPHONE_NUMBER"
    },
    "Tags": {
      "aws:connect:instanceId": "95acc2c2-8444-4ced-8eda-31134ef9dc
      "aws:connect:systemEndpoint": "+17795484542"
    }
  },
  "Parameters": {
    "SessionId": "673da00e-c663-496f-85f1-a23ae59795ed",
    "auth_type": "match_account_number",
    "UD_CaptureAccountNumber": "2108266947"
  }
 },
 "Name": "ContactFlowEvent"
}
```


Sample Output


```
{
  "statusCode": 200,
  "body": {
    "ANIMatch": "Yes",
    "properties": "Single",
    "checkAttribute": "Address",
    "areaCode": "850",
    "phoneNumber": "8503776619",
    "address": "14 NORWOOD DR, PENSACOLA, 32506",
    "addressStart": "14",
    "lastName": "GLASS",
    "lastNameStart": null,
    "businessName": null,
    "businessNameStart": null,
    "accountRegion": "FPLNW",
    "accountType": "Residential",
    "accountStatus": "Active",
    "accountNumber": "2108266947",
    "premiseNumber": "60579064",
    "lmisFlag": "",
    "confidenceScore": "NaN",
    "UD_CaptureCustomerPhoneNumber": null,
    "UD_CaptureAccountNumber": "2108266947",
```


Error Handling

- All logic inside lambda_handler is wrapped in a try-except block for general error capture.
- Each critical helper function has its own try-except with specific logging for traceability.
- Fallback to mock data occurs when API calls fail, and USE_MOCK_DATA is set to true.

- Handles API response errors (e.g., empty outputs) with logged exceptions and structured fallback responses.
- Time durations are logged for key processing blocks to aid performance debugging.
- Returns a safe, default response in case of unhandled errors to avoid contact flow failures.

Logging and Debugging

- Appropriate levels of logging DEBUG / ERROR / INFO are incorporated in the code in wherever necessary to facilitate in debugging.
- Custom Module logger_utils.py has been defined and used in the lambda to enable the ContactId to be visible in the logs to make it easier to tie the lambda execution logs back to the actual AWS Connect invocation.

1.15.4. Lambda – 2XSFNV-EI864-Outage-SendCustomerInteractions-LF

Description:

- This Lambda function sends customer interaction data from Amazon Connect to a Mulesoft API endpoint.
- It builds a structured payload using contact attributes, handles API response validation, and logs each step.
- If enabled, it supports fallback to mock data (in Dev env only) in case of API errors or connection failures.

Key Components

Imports:

- json – Used to serialize and deserialize API payloads and responses.
- os – Reads environment variables.
- datetime – Gets the current UTC timestamp for the interaction.
- uuid – Generates unique correlation and interaction IDs for tracking.
- logger_utils – Custom module for initializing structured logging and suppressing SDK logs.
- api_client – Custom module for initializing the Mulesoft API client.
- api_utils – Provides helper methods for API calls, token retrieval, and endpoint construction.

Functions:

- lambda_handler – Main Lambda entry that builds the interaction payload and sends it to the Mulesoft API.

Lambda Configurations & Environment Variables:

- Memory: 1024 MB
- Ephemeral Storage: 512 MB
- Lambda Timeout: 7 secs

Variable Name	Values	Description
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-instrument	Used for enabling the X-Ray Trace for Lambda
LOG_LEVEL_VALUE	DEBUG	Used to set the log level
REGION_NAME	us-east-1	AWS region for resource access
REQUEST_TIMEOUT	3	Timeout for API requests (seconds)
SECRET_NAME	2xsfnv-ei864-mulesoft-token	Secret name for Mulesoft API token
SECRET_SESSION_SERVICE	secretsmanager	AWS Secrets Manager service
SSM_NAME	ei864/api/mulesoft/contact-center/host/stage	Parameter Store key for host URL
USE_MOCK_DATA	True	Toggles use of mock data on failure
http_proxy	http://webproxyeva.fpl.com:8080	Proxy for outbound HTTP requests
https_proxy	https://webproxyeva.fpl.com:8080	Proxy for outbound HTTPS requests
no_proxy	*.fpl.com,*.neeaws.local	Bypass proxy for internal domains

Custom Modules Used

Custom Module	Role
logger_utils	Sets up structured logging, suppresses SDK logs, adds contact ID context.
api_client	Initializes and manages the Mulesoft API client connection.
api_utils	Handles API endpoint calls, token retrieval, JSON formatting, and custom errors.

Functional Logic

- Initializes structured logger and extracts ContactId for tracing.
- Retrieves Mulesoft host URL and secret token from AWS services.
- Builds a POST payload including channel, timestamp, IDs, and metadata from Amazon Connect parameters.
- Transforms the "notes" field into a list if provided as a comma-separated string.
- Sends the payload to the Mulesoft "interaction" endpoint via a secure API call.
- Parses and validates the API response; raises an error if status is not 200.
- If the response is a JSON string, attempts to parse it before further use.
- On error, logs the issue and optionally returns a mock response if enabled via USE_MOCK_DATA (only in Dev Env).

Sample Event (Input):


```
{
"Details":{
  "Parameters":{
  "source": "fpl.com",
  "channel": "VOICE",
  "correlationId": "59910589-a7ee-2534-9718-369aaa89a165",
  "eventName": "ticketSubmission",
  "contractAccount": "2100000054",
  "businessPartner": "2100000054",
  "interactionId": "9bd09d6b-3dc8-42c6-a0a9-eb4e14cf192d",
  "comments": "Test Comment from 30th Apr 2025 - 01:45PM",
  "notes": "Test_Note_1, Test_Note_2",
  "timestamp": "2023-04-05T12:34:00",
  "webId": "",
  "payload": {}
  },
  "ContactData": {
      "InstanceARN": "arn:aws:connect:us-east-1:059221074852:instance/9
      "ContactId": "4ca79319-fdee-4cf5-ac1d-53af8fa3d0d3"
    }
  }
}
```


Output Format:

```
{
  "statusCode": 200,
  "body": {
    "source": "fpl.com",
    "channel": "IVR",
    "correlationId": "b857e117-d90a-43ff-9595-0be7dd1fbec5",
    "eventName": "ticketSubmission",
    "error": {
      "code": "",
      "message": ""
    },
    "output": {
      "message": "Interaction Posted to SAP/UCDP",
      "timestamp": "2025-05-28 17:01:26"
    }
  }
}
```


Error Handling

- The lambda_handler function is enclosed in a try-except block for full error capture.
- Logs detailed exception messages for troubleshooting and traceability.
- If the API returns a non-200 status or invalid JSON, a runtime or value error is raised.
- Supports fallback behavior using mock data when USE_MOCK_DATA is set to true.
- Provides a structured 500 response with error details if mock fallback is disabled.

Logging and Debugging

- Appropriate levels of logging DEBUG / ERROR / INFO are incorporated in the code in wherever necessary to facilitate in debugging.
- Custom Module logger_utils.py has been defined and used in the lambda to enable the ContactId to be visible in the logs to make it easier to tie the lambda execution logs back to the actual AWS Connect invocation.

1.15.5. Lambda – 2XSFNV-EI864-Outage-InteractionHandler –LF – Updated content to be updated

1.16. Lambda Layers

We have a common lambda layer named - 2XSFNV-EI864-common-layer, which has all the necessary python libraries and custom module packaged in it.

Following are the python libraries installed in it:

- requests (v2.32.3) – Simplifies making HTTP requests, commonly used for REST APIs.
- phonenumbers (v8.13.52) – Parses, formats, and validates international phone numbers.
- fuzzywuzzy (v0.18.0) – Performs fuzzy string matching using Levenshtein Distance.
- backoff (v2.2.1)– Adds automatic retry behavior with exponential backoff for functions.
- openpyxl (v3.1.5) – Reads and writes Excel (.xlsx) files with support for formatting and formulas.

1.16.1. Lambda Layer Module – api_client.py

Description:

- This custom module provides a simple API client for sending GET and POST requests to a specified base URL.
- It abstracts HTTP request logic using the `requests` library and allows configurable headers, payloads, and timeouts.

Key Components

Imports:

- json – Used to handle JSON data (imported but not used directly here).
- requests – Python library to send HTTP/HTTPS requests in a user-friendly way.

Functions:

- __init__ – Initializes the APIClient with a base URL for all requests.
- post – Sends a POST request to the given endpoint with JSON payload and headers.
- get – Sends a GET request to the given endpoint with query parameters and headers.

Functional Logic:

- Initializes the client with a base URL to be used across all requests.
- Constructs the full URL by appending the given endpoint to the base URL.
- For `post`, sends a JSON body with headers and timeout using `requests.post()`.
- For `get`, sends query parameters with headers and timeout using `requests.get()`.
- Returns the raw `response` object from the `requests` library for further processing.

Error Handling:

- Since this is a very basic function to make REST api requests, any error occurring is propagated on to the caller function where it is getting handled.

1.16.2. Lambda Layer Module – api_utils.py

Description:

- This module provides utilities for securely calling Mulesoft APIs using retry logic,
- retrieving secrets and configuration from AWS, and cleaning or flattening JSON responses.
- It supports robust error handling, timing logs, and mock fallback via configurable environment variables.

Key Components

Imports:

- backoff – Adds automatic retry logic with exponential backoff for API requests.
- json – Serializes and deserializes data between JSON and Python dicts.
- os – Reads environment variables for configuration.
- uuid – Generates unique IDs for correlation and interaction tracking.
- boto3 – AWS SDK for interacting with services like SSM and Secrets Manager.
- requests – Sends HTTP(S) requests using simple syntax.
- boto3.session.Session – Creates a custom AWS session for regional and service-specific clients.
- botocore.exceptions.ClientError – Catches and handles errors from AWS service calls.
- datetime – Tracks and formats timestamps for monitoring and logs.
- logging – Configures custom logging with levels and output formatting.
- time – Provides timing utilities (not directly used here).

Functions:

- get_mulesoft_host – Retrieves the Mulesoft host URL from AWS SSM Parameter Store.
- get_secret – Fetches an OAuth access token from AWS Secrets Manager.
- make_api_request – Sends GET or POST requests to the API with retry logic and structured headers.
- call_api_endpoint – Wrapper that calls make_api_request with default parameter handling.

- clean_json – Cleans JSON by preserving keys and replacing `None` values without dropping them.
- unnest_json – Flattens deeply nested JSON objects into a single-level dict with simplified keys.

Functional Logic:

- Initializes logging with the level set by the LOG_LEVEL_VALUE environment variable.
- Connects to AWS SSM to retrieve configuration values like the Mulesoft host.
- Connects to Secrets Manager using a session and returns the `access_token`.
- Wraps all API requests in a retry mechanism (up to 2 tries) using `backoff`.
- Generates headers with auth token, correlation ID, and tracking metadata before making the request.
- Measures and logs the time taken for each external call (SSM, Secrets, and API).
- Provides helper functions to flatten nested JSON (`unnest_json`) and clean out `None` values (`clean_json`).
- Returns responses in structured format with `statusCode` and `body` keys.

Error Handling:

- All AWS service calls (SSM, Secrets Manager) include try-except blocks for catching errors.
- If parameters are not found in SSM, it returns a structured 404 response.
- Secrets retrieval logs and raises the exception for upstream handling.
- API requests include retry logic (max 2 tries) using `backoff` on common `requests` exceptions.
- Fallback logic logs failures, and the API call time is logged regardless of success or failure.

1.16.3. Lambda Layer Module – logger_utils.py

Description:

- This module provides logging utilities for AWS Lambda functions, including contact ID injection,
- log level configuration, and suppression of noisy SDK logs from libraries like boto3 and OpenTelemetry.
- It ensures clean, traceable logs by tagging each log line with a unique ContactId.

Key Components

Imports:

- logging – Provides Python's standard logging infrastructure for capturing and formatting logs.
- os – Accesses environment variables like LOG_LEVEL_VALUE to set log level.
- sys – Directs log output to standard output for Lambda's log stream visibility.

Functions:

- ContactIdFilter – Logging filter that injects a unique contact ID into each log record.
- get_logger – Initializes and returns a logger with a stream handler and ContactId formatting support.
- suppress_sdk_logs – Suppresses verbose logs from AWS SDK and OpenTelemetry libraries to reduce noise.

Functional Logic:

- Reads the desired log level from the environment variable LOG_LEVEL_VALUE.
- Sets up a stream handler that writes logs to standard output (CloudWatch logs).
- Formats logs to include timestamp, log level, and custom ContactId tag.
- Creates a global logger and configures it with the specified level and handler.
- Provides a `ContactIdFilter` class that attaches a contact_id to every log record.
- Includes a utility to suppress logs from common noisy SDKs like boto3, botocore, and OpenTelemetry.

Error Handling:

- No explicit try-except blocks, as this module performs safe setup operations.
- All logging-related configuration is deterministic and does not introduce runtime errors.
- Any missing environment variables default gracefully (e.g., log level defaults to INFO). Fallback logic logs failures, and the API call time is logged regardless of success or failure.

1.17. API specifications

1.17.1. HVCA Authentication API

Description – This API is used in HVCA Authentication. It takes the ANI or Phone number or account number to authenticate the customer

Response from API

```
{
    "source": "AWS Connect",
    "channel": "Contact Center",
    "correlationId": "esfsef42342fcscd",
    "eventName": "hvosAuthentication",
    "error": {
        "code": "",
        "message": ""
    },
    "output": [
        {
            "customerInfo": {
                "name": "LAURA COLEMAN",
                "firstName": "LAURA",
                "lastName": "COLEMAN",
                "customerNumber": "1000661152",
                "customerType": "Consumer"
            },
            "addressInfo": {
                "street": "6019 RUNNING DEER RD",
                "houseNumber": "",
                "city": "MILTON",
                "county": "",
                "postalCode": "32570"
            },
            "contactInfo": {
                "primaryPhoneNumber": "8506268520",
                "secondaryPhoneNumber": "8504909406"
            },
            "accountInfo": {
                "accountNumber": "2113927376",
                "accountStatus": "Active",
                "accountType": "Residential",
                "accountName": "Laura Coleman",
                "accountRegion": "FPLNW",
                "premiseNumber": "60697587",
                "rateCategory": "FERS_44",
                "lmisFlag": "",
                "mespFlag": ""
            },
            "oncallInfo": {
                "credits": {
                    "summer": 0,
                    "winter": 0,
                    "totalLmis": 0
                },
                "appliances": {
                    "AC": 0,
                    "HS": 0,
                    "WH": 0,
                    "PP": 0
                }
            },
            "demandInfo": []
        }
    ]
}
```


1.17.2. Smart Outage Inquiry API

Description – This API primarily brings the ticket related details for the given account number.

Response from API

```
{
    "source": "AWS Connect",
    "channel": "Chatbot",
    "correlationId": "sdfsdf2134",
    "eventName": "inquiry",
    "error": {
        "code": "",
        "message": ""
    },
    "output": {
        "outageCallFilterFlag": "N",
        "universalStormToggle": "N",
        "weatherConditionType": null,
        "weatherEvent": null,
        "floodMessage": "N",
        "ticketCustomerEffected": "1",
        "ticketMeterFlag": "N",
        "ticketStatus": "new",
        "ticketCompletedDate": "",
        "ticketType": "SNCU",
        "ticketRestoredDate": "",
        "ticketCreateDate": "05/28/2025 14:36:29",
        "ticketComplaintCategories": [
            "noCurrent"
        ],
        "ticketFirstCallComplaints": [
            "noCurrent"
        ],
        "ticketFoundFlag": "Y",
        "ticketNumber": null,
        "itrEtrType": "N",
        "estimatedRestorationTime": "",
        "ticketNumOfCalls": "1",
        "OutageTicketStatus": "New",
        "ticketFloodlistFlag": "N",
        "ticketCauseCode": null,
        "ticketReferredTime": null,
        "ticketNextStep": null,
        "meterStatus": "Unavailable",
        "meterStatusTimeStamp": "",
        "displayMeterStatusFlag": "N",
        "proceedMessage": "Active Ticket",
        "ticketPrevCallFoundFlag": "Y",
        "recentlyClosedTicket": "",
        "activeTicket": "",
        "additionalInfo": "Incident Status : Incident Found; Meter Status :Not available for GULF Meter accounts;",
        "remarks": null,
        "code": "200",
        "message": "SUCCESS",
        "customerDisqualifiedReportOutage": "No"
    }
}
```


1.17.3. ADMS Ticket Submission API

Description – This API submits a ticket to the ADMS for reporting an outage for the customer with details fetched during the customer experience navigation in the call flow

Response from API


```
{
    "source": "AWS Connect",
    "channel": "IVR",
    "correlationId": "124143242",
    "eventName": "ticketSubmission",
    "error": {
        "code": "",
        "message": ""
    },
    "output": {
        "message": "Successfully Sent to ADMS Queue",
        "code": "SUCCESS",
        "premiseNumber": "60697587",
        "zzSubCode": "1600",
        "status": "200"
    }
}
```


1.17.4. Interactions API

Description – This API submits customer interactions to SAP which inturn posts these interactions to the UCDP

Response from API


```
{
    "source": "fpl.com",
    "channel": "Web",
    "correlationId": "59910589-a7ee-409b-9718-369aaa89a113",
    "eventName": "checkActiveTicket",
    "error": {
        "code": "",
        "message": ""
    },
    "output": {
        "message": "Interaction Posted to SAP/UCDP",
        "timestamp": "2023-04-05T12:34:00"
    }
}
```

1.18. Prompt Management

The Prompt Management process is a generic process that has been created to enable us to store the prompts in the DynamoDB table, which can be used across all the epics. The prompts that are stored in the DynamoDB table are later fetched in the connect flows by invoking the 2XSFNV-EI864-PromptManagement-DBQuery-LF lambda based on the workflow.

1.18.1. HVOS Prompt Repo File

The prompts that are to be inserted are signed off and given by the business in the form of an Excel File.

A fixed structure of the input file is agreed upon to maintain consistency so that it can be processed by the python scriptFollowing is the structure of the file:

Channel	Language	Domain	Workflow	Block	English Prompts	English - Variation of Prompt	Spanish Prompts
Chat	English	Outage	Agent_Transfer	Transfer_to_agent	Please click the button below to call one of our agents.		Por favor, haga clic en el botón de abajo para llamar a uno de nuestros agentes.
Chat	English	Outage	Collect_Contact_Number	Check_for_valid_number	Do you have a phone number I can reach you at?		¿Tiene un número de teléfono al que pueda contactarlo?
Chat	English	Outage	Collect_Contact_Number	Confirm_Contact_Number	Can you please provide the number we should use to contact you if necessary?		¿Podría proporcionarnos el número que deberíamos usar para contactarlo si es necesario?
Chat	English	Outage	Disambiguate	Disambiguate	Could you please describe your issue briefly?		¿Podría describir brevemente su problema?
Chat	English	Outage	Disambiguate	Outage_down_message	Sorry, our outage reporting system is unavailable right now. Please wait while I transfer you to a customer service representative.		Lo siento, nuestro sistema de reporte de interrupciones no está disponible en este momento. Por favor, espere mientras lo transfiero a un representante de servicio al cliente.
Chat	English	Outage	ETR_Entry	Additional_instructions	Would you like to provide some additional instructions to access the FPL equipment at or near your location?		¿Le gustaría proporcionar algunas instrucciones adicionales para acceder al equipo de FPL en o cerca de su ubicación?
Chat	English	Outage	ETR_Entry	NO_ETR_Storm	We can't provide an estimated restoration time due to {SOI_eventType} {SOI_eventName}. Agents currently have no further information. Rest assured, we are working to restore your power safely and as quickly as possible. Thank you for your patience and understanding.		No podemos proporcionar un tiempo estimado de restauración debido a {SOI_eventType} {SOI_eventName}. Los agentes actualmente no tienen más información. Tenga la seguridad de que estamos trabajando para restaurar su electricidad de manera segura y lo más rápido posible. Gracias por su paciencia y comprensión.
Chat	English	Outage	ETR_Ticket_Type_Check	Necessary_work	We've temporarily interrupted power for maintenance to ensure better service in the future. We understand this is inconvenient, and we appreciate your understanding.		Hemos interrumpido temporalmente el suministro eléctrico para realizar mantenimiento y asegurar un mejor servicio en el futuro. Entendemos que esto es inconveniente y apreciamos su comprensión.
Chat	English	Outage	ETR_Time	Restoration_time_Disclaimer	Please remember that estimated times of restoration are subject to change based on the issue, which can be sooner or later.		Por favor, recuerde que los tiempos estimados de restauración están sujetos a cambios según el problema, lo cual puede ser más pronto o más tarde.

Link to the File: https://nee.sharepoint.com/teams/EXT_CIStoSAP-AWSConnect_Lex/_layouts/15/doc.aspx?sourcedoc={7376b97c-aaa8-45c6-989a-657efd81d36b}&action=edit

1.18.2. DB Insert Script

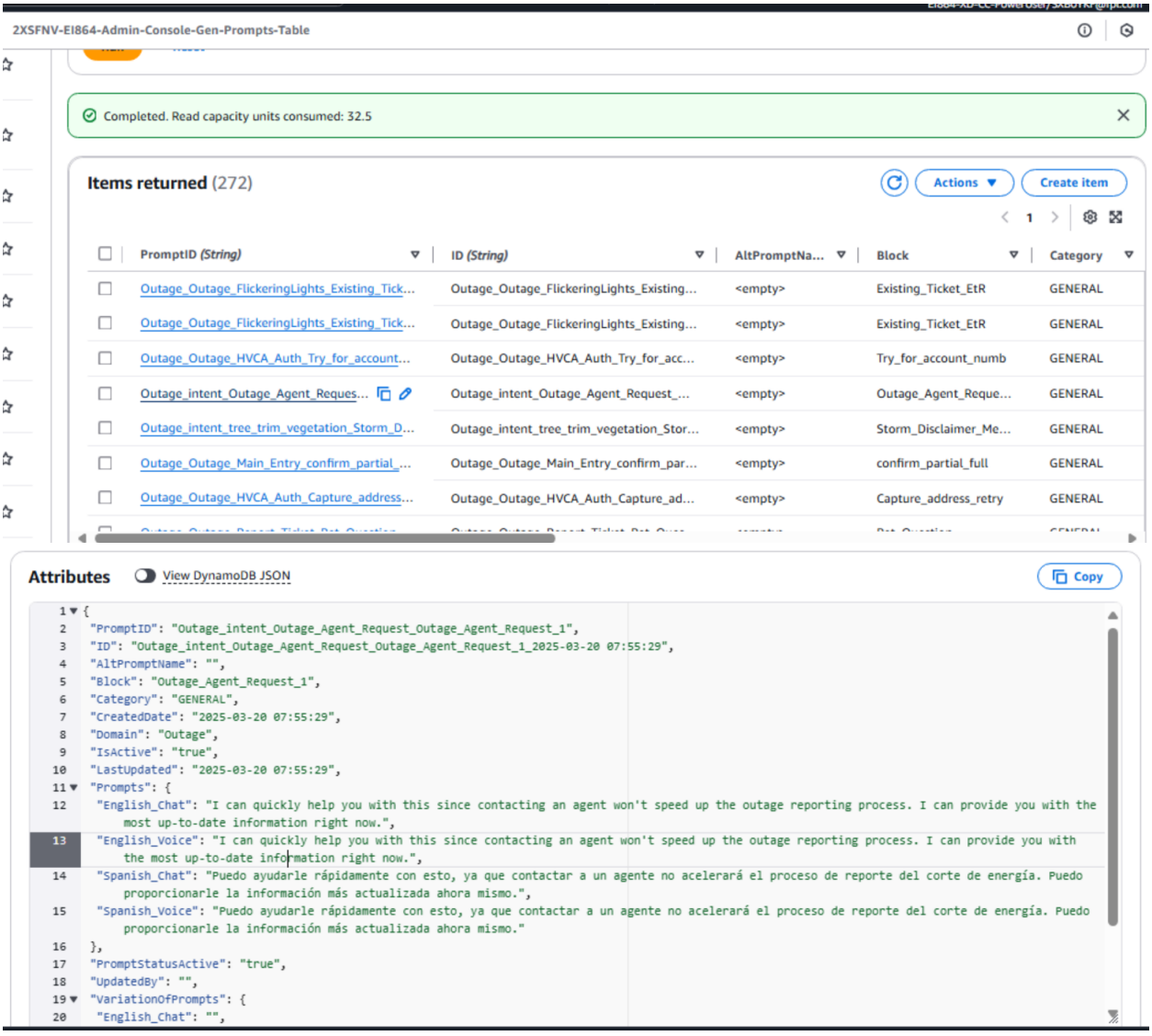
To process the above input excel file, we have a python script that would process this input file and push the prompts to the DynamoDB table.

- This script is in the ei864-XD-CC-Infrastructure github repo -> prompt-data-initial directory.
- This script will read the input excel file using pandas, apply some data cleaning methods (like trimming of leading/trailing spaces, removal of special characters etc.) and convert each row into a JSON object. Once the rows are converted into the JSON format, they are written onto the DynamoDB table.
- Link to the script: https://github.com/NextEraEnergy/ei864-XD-CC-Infrastructure/tree/development/prompts-table/prompt-data-initial
- Once the code is deployed or any changes are made to the input excel file (which is stored in the same directory in the repo) and committed, the CICD process will run the prompts workflow

which will trigger this script and deploy the prompts in the corresponding environment of DynamoDB (dev/test etc.)

- The DynamoDB table that is used to store the prompts is named as 2XSFNV-EI864-Admin-Console-Gen-Prompts-Table.

Following is the structure of the table:



- The audit columns included in addition to the columns that we are already using in the input file are CreatedDate, LastUpdated, UpdatedBy, IsActive, ActiveStatus etc.
- In the DBInsert script in the Infrastructure Repo, these additional columns will be added without needing to make any changes to the input csv files.

- This table will also hold the Flow Level Messages (FLM) which are inserted by the admin console and are used in the flows.

Following are the key improvements that were added to the script over time:

Processing only the recently committed files:

- The changes were made to the workflow script such that the input files that were recently committed are passed as the system argument when the workflow is triggered.

- In the above example the script is triggered with the command :
  python3 create-or-update-prompts-data.py 2XSFNV-EI864 '["outage_prompts_2.csv"]'
- Here outage_prompts_2.csv is recently committed and is passed in a string format - '["outage_prompts_2.csv"]'

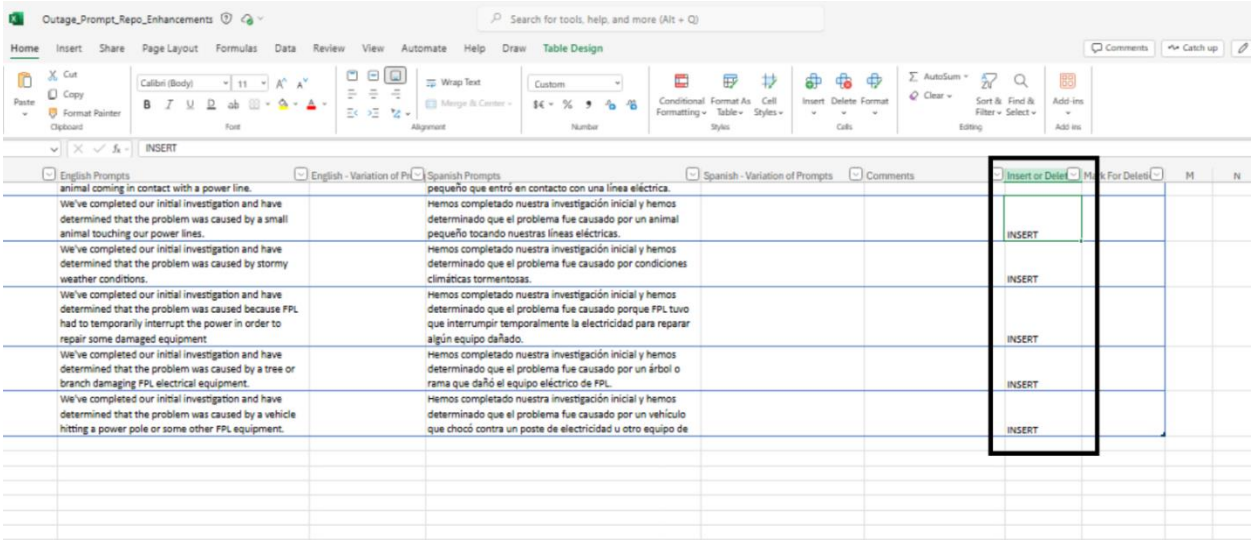
```
if __name__ == "__main__":
    table_name = sys.argv[1] + "-Prompts-Table"
    recent_file_commits = sys.argv[2]

    print(f" initial data into table {table_name}")
    print(f"Changed Files: {recent_file_commits}")
```

```
recent_file_commits = json.loads(recent_file_commits)

for file in recent_file_commits:
    try:
        print( f"***** processing this file [{file}] now ***** ")
```

- In the script we are reading this argument into recent_file_commits variable and then using json.loads() we are converting the list that is in a string format into actual list object.
- In the for loop, only the csv files present in this list will be processed.

Processing only the recently added records in the csv:

- For only processing the new records that have been added to the csv instead of processing all the records, there is an extra column that is added in the csv.
- This extra column would be a ‘Insert Delete Flag’ at which a set of prompts are added to the csv.

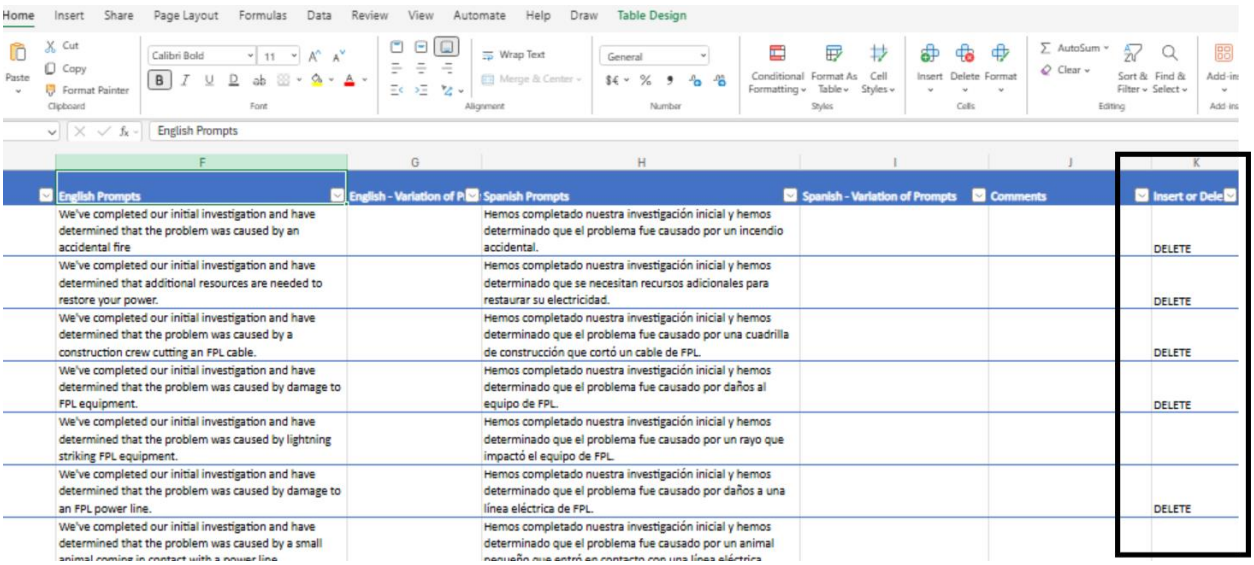


- For the prompts that are newly added we can add the tag ‘INSERT’ so that they can be identified and processed by the code.
- The prompts that are already updated in earlier push will have this column blank.
- In the code, wherever we have mentioned ‘INSERT’, those records will be filtered and pushed to the DDB.

```
# only select the recent addition to the file based on timestamp column
df = df[df['Insert Delete Flag'] == "INSERT"]
```

<u>Deletion of specified old records:</u>

- To delete the old records, a new column is added to the csv to keep track of which records are to be deleted from the DDB



English Prompts	English - Variation of P	Spanish Prompts	Spanish - Variation of Prompts	Comments	Insert or Dele
We've completed our initial investigation and have determined that the problem was caused by an accidental fire		Hemos completado nuestra investigación inicial y hemos determinado que el problema fue causado por un incendio accidental.			DELETE
We've completed our initial investigation and have determined that additional resources are needed to restore your power.		Hemos completado nuestra investigación inicial y hemos determinado que se necesitan recursos adicionales para restaurar su electricidad.			DELETE
We've completed our initial investigation and have determined that the problem was caused by a construction crew cutting an FPL cable.		Hemos completado nuestra investigación inicial y hemos determinado que el problema fue causado por una cuadrilla de construcción que cortó un cable de FPL.			DELETE
We've completed our initial investigation and have determined that the problem was caused by damage to FPL equipment.		Hemos completado nuestra investigación inicial y hemos determinado que el problema fue causado por daños al equipo de FPL.			DELETE
We've completed our initial investigation and have determined that the problem was caused by lightning striking FPL equipment.		Hemos completado nuestra investigación inicial y hemos determinado que el problema fue causado por un rayo que impactó el equipo de FPL.			
We've completed our initial investigation and have determined that the problem was caused by damage to an FPL power line.		Hemos completado nuestra investigación inicial y hemos determinado que el problema fue causado por daños a una línea eléctrica de FPL.			DELETE
We've completed our initial investigation and have determined that the problem was caused by a small		Hemos completado nuestra investigación inicial y hemos determinado que el problema fue causado por un animal			

- This new column is 'Insert Delete Flag' which has 'DELETE' wherever we need to delete the corresponding record.

```
# get the prompt ids for the records that are marked for deletion
records_to_delete = df[df['Insert Delete Flag'] == "DELETE"].copy()
prompt_ids_to_delete = set(records_to_delete['Domain'] + "_" + records_to_delete['Workflow'] + "_" + rec
```

- If a particular prompt is marked as 'DELETE' then the prompt ids associated with these prompts are noted and stored in a variable.

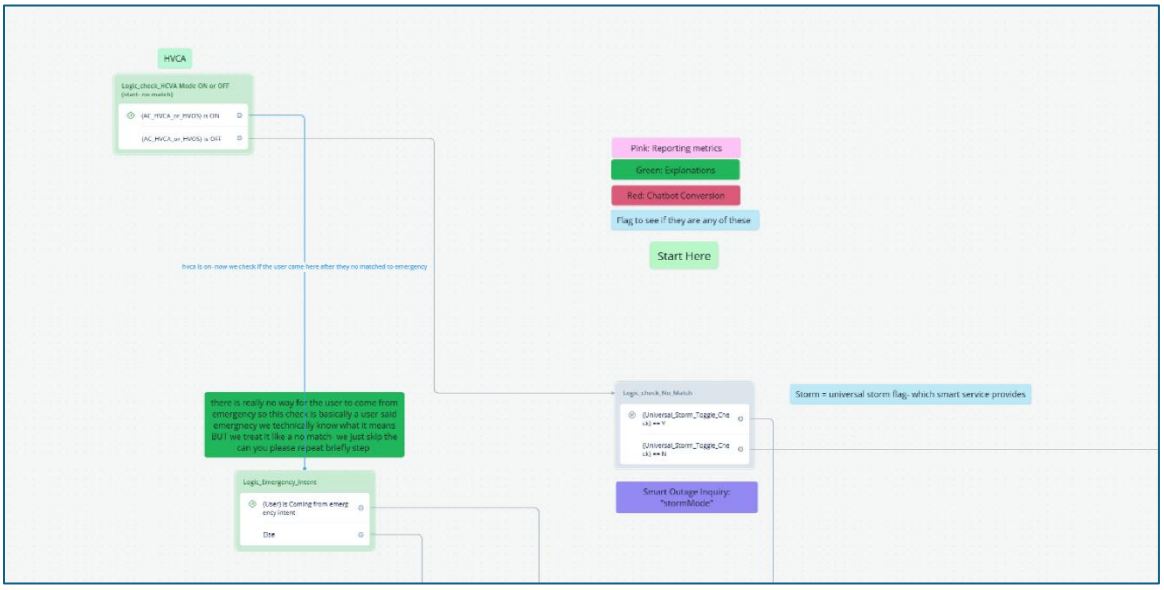
```
print(f"### Prompt ids to delete: {prompt_ids_to_delete}")

if prompt_ids_to_delete:
    bulk_delete_prompt_ids(table, prompt_ids_to_delete)
```

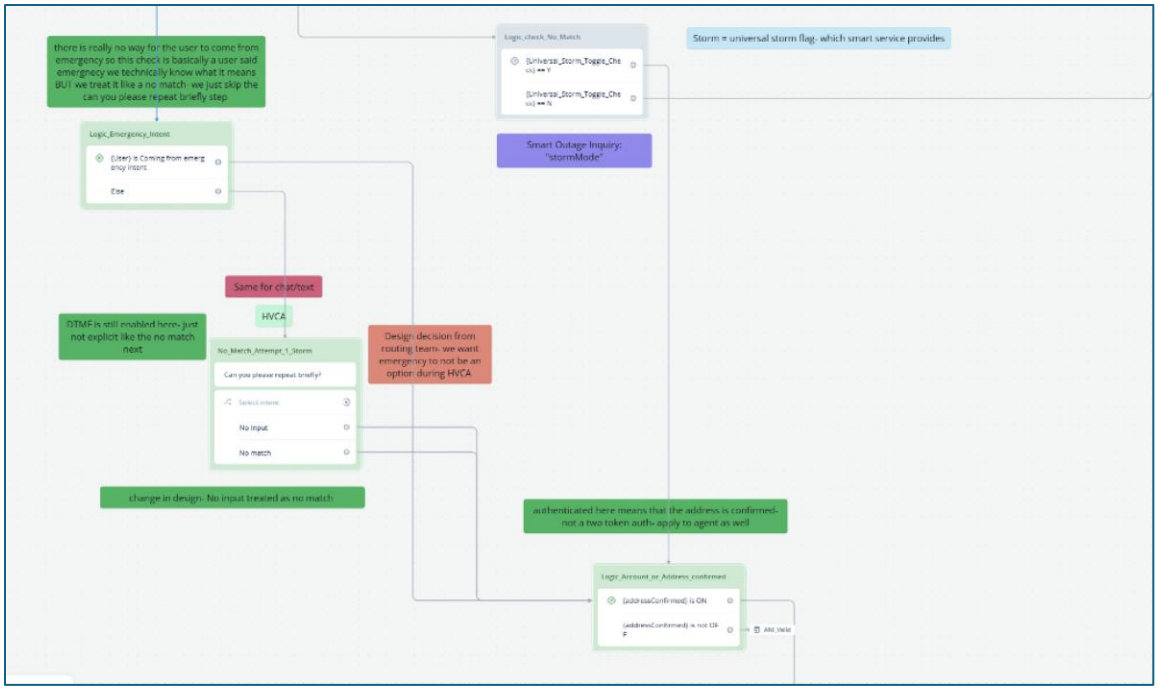
- Then this set of prompt ids to be deleted is passed to the function which will take the prompt id one by one and simply do a delete item operation.

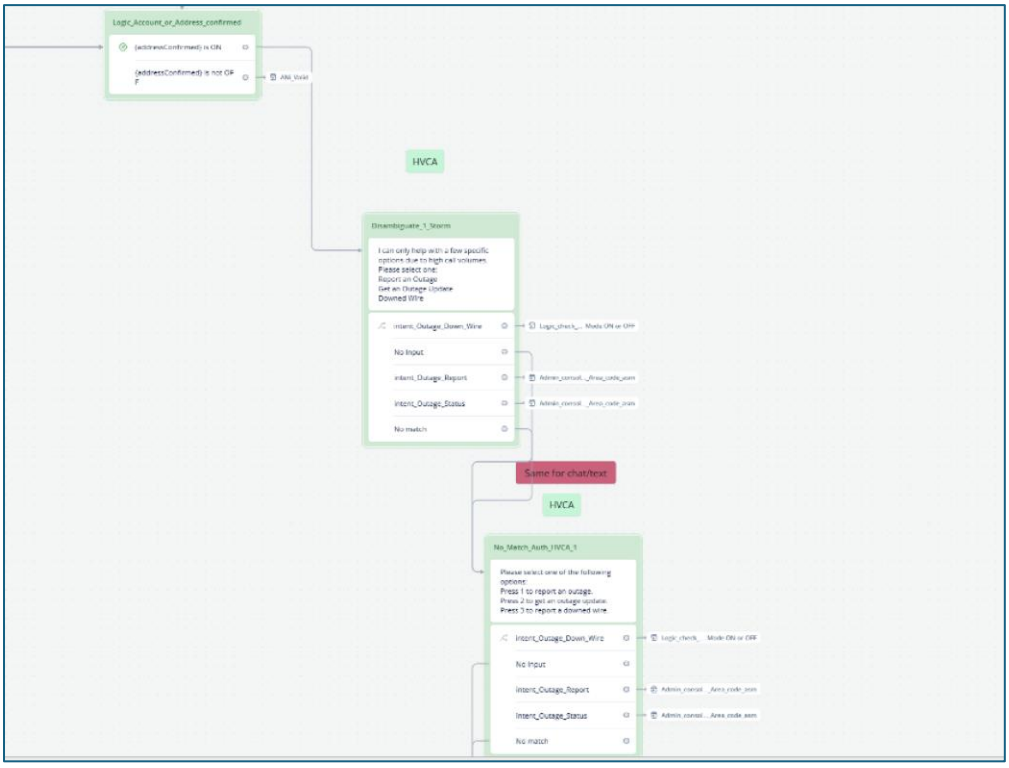
1.19. HVOS Recovery Behavior: Exception Handling, Fallbacks, Retries, Timeouts

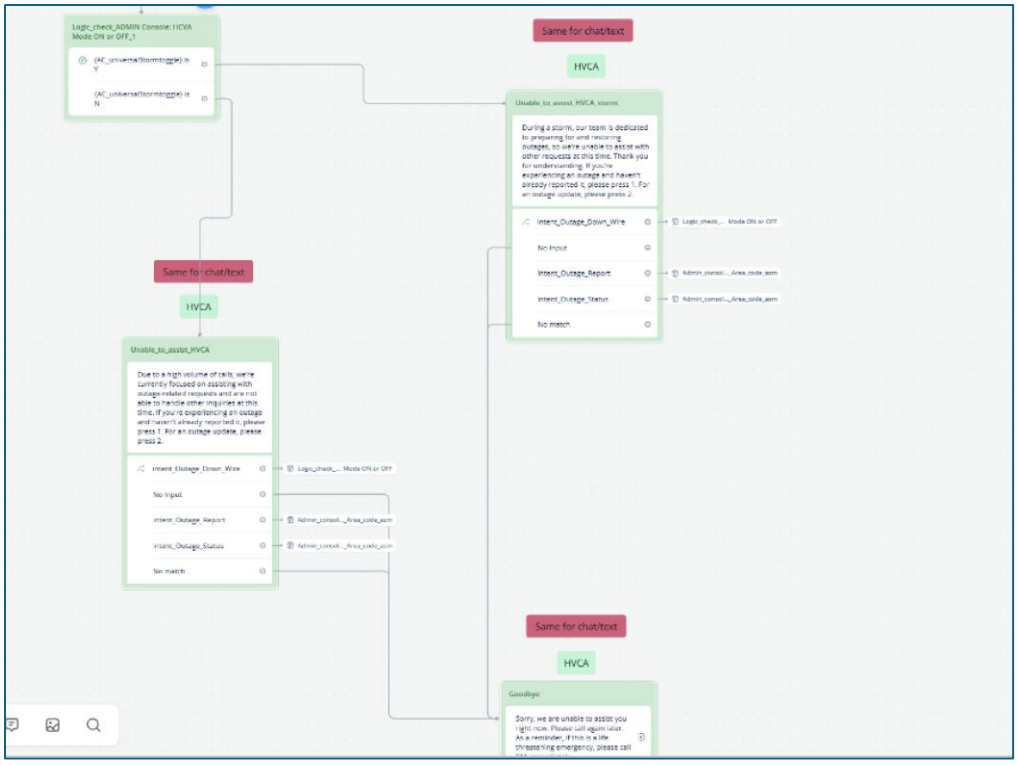
General Handling

- Phone numbers for HVOS are specific for HVOS scenarios.
- For R1, we have common HVCA authentication
- For authentication in R2, design will cover the global modules and recovery behaviors to suit multiple domains in R2.
- No Match No Input
- Intent_Outage_No_Match voice flow (link to voiceflow)

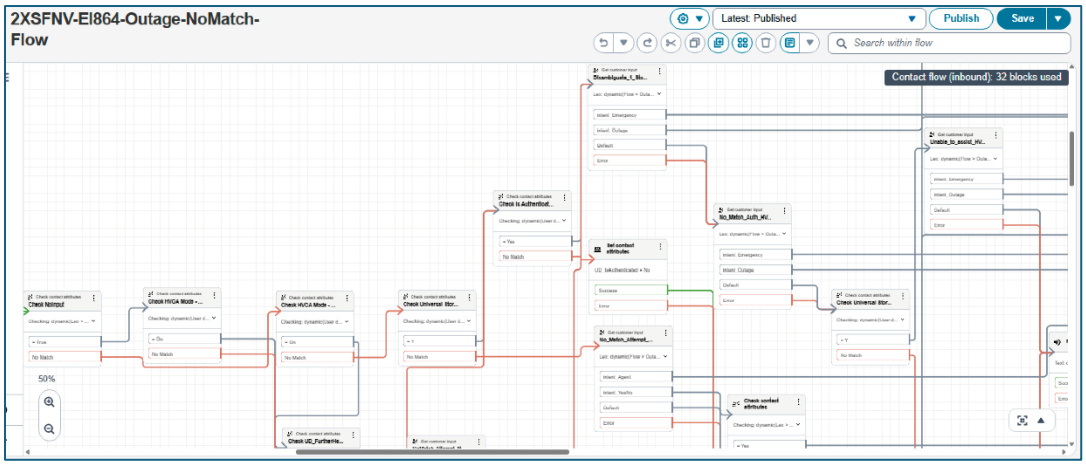



there is really no way for the user to come from emergency so this check is basically a user said emergency we technically know what it means BUT we treat it like a no match- we just skip the can you please repeat briefly step
Logic_Emergency_Intent
(User) is Coming from emergency intent
Else
Logic_check_No_Match
{Universal_Storm_Toggle_Check} == Y
{Universal_Storm_Toggle_Check} == N
Storm = universal storm flag- which smart service provides
Smart Outage Inquiry: "stormMode"
Same for chat/text
HVCA
DTMF is still enabled here- just not explicit like the no match next
No_Match_Attempt_1_Storm
Can you please repeat briefly?
Select intent
No Input
No match
Design decision from routing team- we want emergency to not be an option during HVCA
change in design- No input treated as no match
authenticated here means that the address is confirmed- not a two token auth- apply to agent as well
Logic_Account_or_Address_confirmed
{addressConfirmed} is ON
{addressConfirmed} is not OFF
ANI_Valid


Logic_Account_or_Address_confirmed
{addressConfirmed} is ON
{addressConfirmed} is not OFF
ANI_Valid
HVCA
Disambiguate_1_Storm
I can only help with a few specific options due to high call volumes.
Please select one:
Report an Outage
Get an Outage Update
Downed Wire
Intent_Outage_Down_Wire
No Input
Intent_Outage_Report
Intent_Outage_Status
No match
Same for chat/text
HVCA
No_Match_Auth_1HVCA_1
Please select one of the following options:
Press 1 to report an outage.
Press 2 to get an outage update.
Press 3 to report a downed wire.
Intent_Outage_Down_Wire
No Input
Intent_Outage_Report
Intent_Outage_Status
No match


Same for chat/text
HVCA
Same for chat/text
HVCA
Same for chat/text
HVCA

1.19.1. 2XSFNV-EI864-Outage-NoMatch-Flow



No Inputs for Report Outage/Status of Outage/Wiredown – No ANI Match

- Customer journey – No ANI match
  - Customer dials the TFN/DID
  - An authentication check is carried out for customer
  - Customer is presented with 3 options to choose from (Report an Outage, Status of an Outage and wiredown)
- Customer gives no input
  - Customer is again presented with 3 options to choose from (Press 1 for Report an outage, press 2 for Status of an Outage, Press 3 for Wiredown)
- Customer gives no input
  - Customer is presented with prompts experience where and is again asked to select between 2 options (Report and Outage and Status of an outage)
- Customer gives no input
  - Post this call is gracefully disconnected by giving a prompt

No Match for Report Outage/Status of Outage/Wiredown – No ANI match

- Customer journey – No ANI match
  - Customer dials the TFN/DID
  - An authentication check is carried out for customer
  - Customer is presented with 3 options to choose from (Report an Outage, Status of an Outage and wiredown)
- Customer gives wrong input

    - Customer is again presented with 3 options to choose from (Press 1 for Report an outage, press 2 for Status of an Outage, Press 3 for Wiredown)
- Customer gives wrong input
    - Customer is presented with prompts experience where and is again asked to select between 2 options (Report and Outage and Status of an outage)
- Customer gives wrong input
    - Post this call is gracefully disconnected by giving a prompt

<u>No Input for Report Outage/Status of Outage/Wiredown – ANI match</u>

- Customer journey – No Input ANI Match
    - Customer dials the TFN/DID
    - An authentication check is carried out for customer
    - Customer is asked to confirm the address
- Customer gives no input
    - Customer is presented with another chance by asking “Lets try again can you please tell me the address you are contacting about?” - 2 times
- Customer gives no input
    - Customer is asked again ”Are you still there?”
    - Customer gives “No” or “No Input”
    - Then customer is traversed to ”Outage-Goodbye-Flow"

<u>No Match for Report Outage/Status of Outage/Wiredown – ANI match</u>

- Customer journey – No match ANI Match
    - Customer dials the TFN/DID
    - An authentication check is carried out for customer
    - Customer is asked to confirm the address
- Customer gives no match
    - Customer is presented with another chance to select “account number or service address” for authentication – 1 time.
- Customer gives no match again
    - Customer is asked with “Could you please repeat briefly?” - 2 times
- Customer gives no match
- Then we gracefully disconnect the call with a statement “We are experiencing a high volume of calls and are unable to help with your request at this moment. For instant updates or report an outage you can visit FPL.com/outage or use our mobile app. Thanks and feel free to hang up whenever you are ready”.

No Match/No input for Yes/No response for Outage-Intent-Detection

(Questions asked :- Are you calling to report an outage? and Are you reaching out about an outage update?)

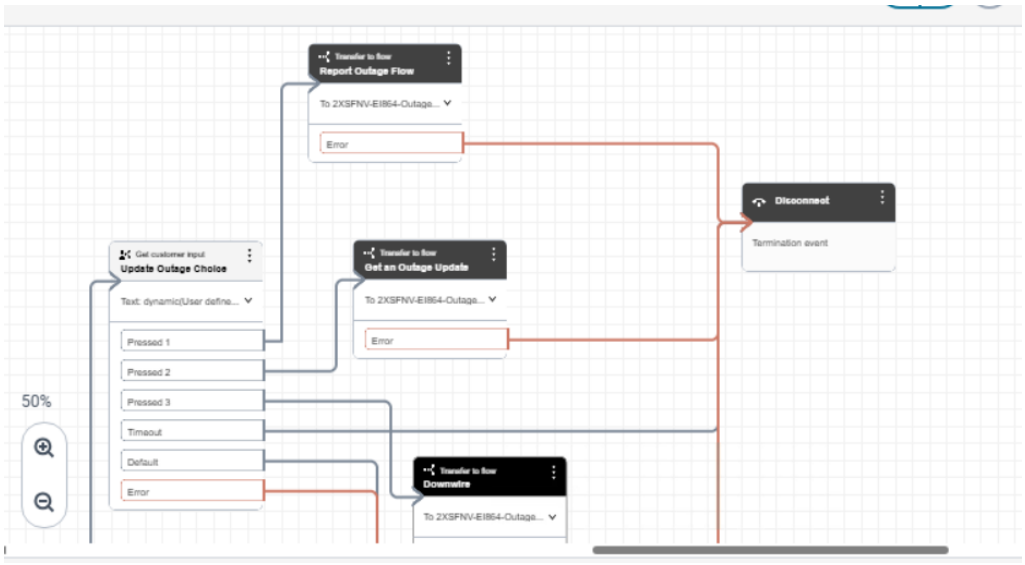
- Customer is presented with 3 options to choose from (Report an Outage, Status of an Outage and wiredown)
- Customer gives wrong input/no input
    - Customer is again presented with 3 options to choose from (Press 1 for Report an outage, press 2 for Status of an Outage, Press 3 for Wiredown)
- Customer gives wrong input/no input
    - Customer is presented with prompts experience where and is again asked to select between 2 options (Report and Outage and Status of an outage)
- Customer gives wrong input/no input
    - Post this call is gracefully disconnected by giving a prompt.

No Match/No input for Yes/No response for rest of the flows

- Customer provides No Match/No input responses for Yes/No questions
    - Customer is given another chance with DTMF options with the prompts stating “lets try again press 1 for yes press 2 for No” - 2 times
    - After second retry of customer gives no input.
    - Customer is presented with a prompt “Are you still there?”
- Customer gives “No” or “No Input”
    - Then customer is traversed to ”Outage-Goodbye-Flow"
    - If customer gives no match
    - Its checked whether customer is authenticated or not.
    - If customer is authenticated we gracefully disconnected the call.
- If customer is not authenticated we check for “Admin Console" No Match toggle. If “OFF” call is disconnected gracefully. If “ON” customer is asked with region details ”NW or Legacy” with prompt “Are you calling about an account in the Florida pan handle including Pensacola and surrounding areas?”
- If “yes” region is considered as NW else Legacy and call is transferred to the agent accordingly.

DTMF

- We can use the DTMF input option from Amazon connect. But the conversational AI functionality won't be available.

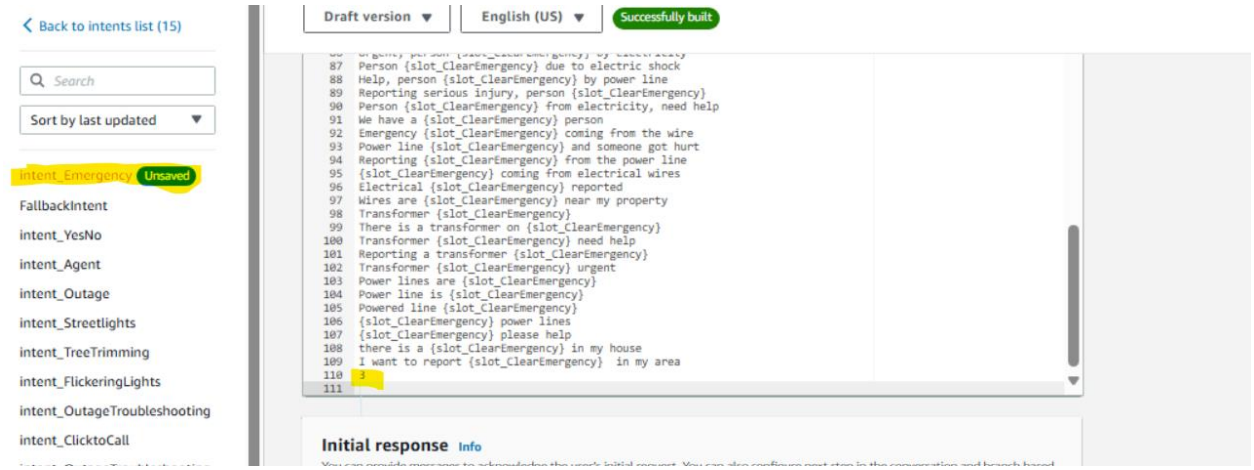


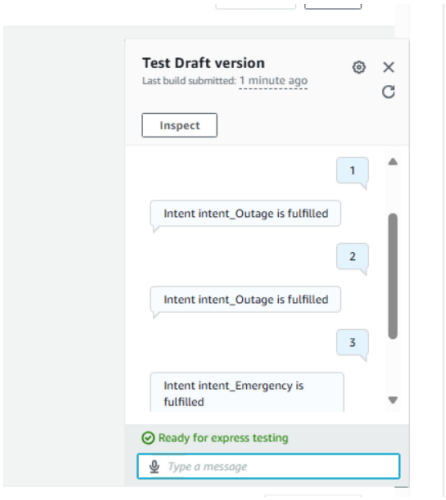
- We can use the AWS lambda to handle the DTMF approach

We can train intents as follows:

Limitations:

Utterances 1 and 2 are trained on intent_Outage, while utterance 3 is trained for intent_Emergency. Ideally, utterances 4 through 9 should go on fallback, but due to Lex NLU capabilities, they map to intent_Outage as a standard behavior. To address this, we need to use AWS Lambda or set a condition in the initial response to fallback if inputTranscript is 4 to 9.



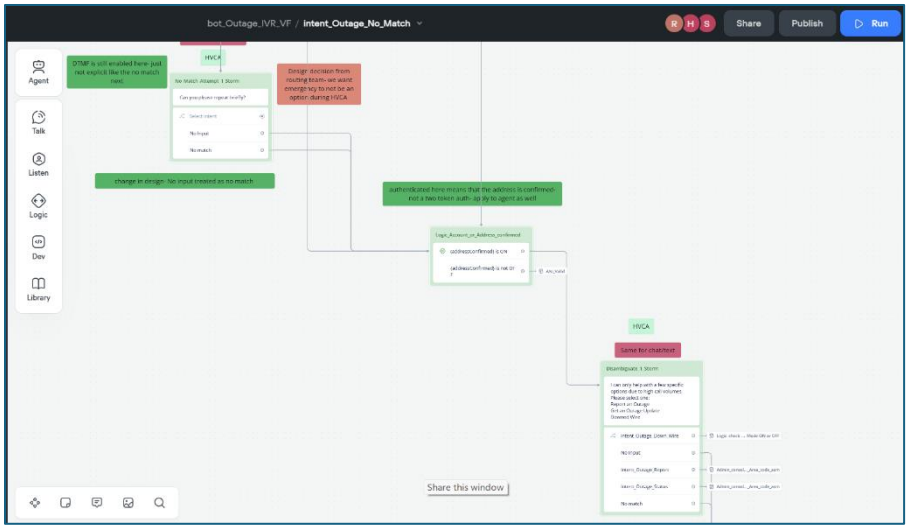


Connect Flows

All flow has been enabled with No match flow (2XSFNV-EI864-Outage-NoMatch-Flow) where for any No match or No input it prompts the customer to provide input

If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue

No of retries enables here is once at each attempt, as per design

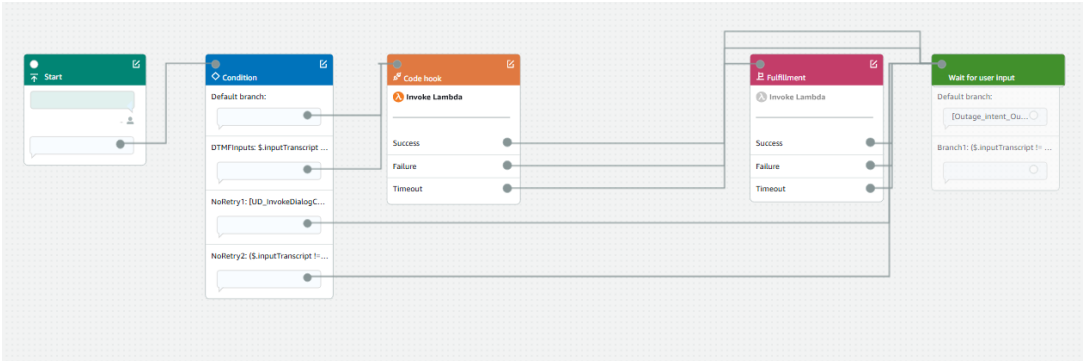


Lex bots

All lex bots has been enabled with Fallback Intent scenario where for any No match or No input it prompts the customer to provide input “Could you please repeat briefly?”

If again the customer does not provide input, or it does not match then it will go to the recovery behavior as per the design. It will start giving the three HVOS options for the customer to choose from to continue

No of retries enables here is once at each attempt, as per design



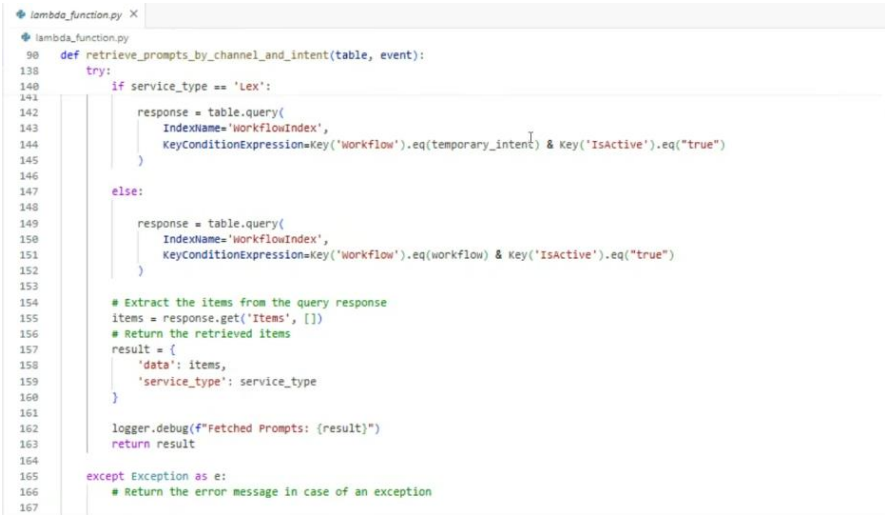
Lambda Functions

Prompt Management

SSML inside DB

DB Query (Global-PromptManagement-DBQuery-LF) –

The Lambda function includes try-except blocks to catch and log exceptions during table query calls and data processing. Errors are logged with logger.error for further investigation



```python
lambda_function.py
def retrieve_prompts_by_channel_and_intent(table, event):
    try:
        if service_type == 'Lex':

            response = table.query(
                IndexName='WorkflowIndex',
                KeyConditionExpression=Key('Workflow').eq(temporary_intent) & Key('IsActive').eq("true")
            )

        else:

            response = table.query(
                IndexName='WorkflowIndex',
                KeyConditionExpression=Key('Workflow').eq(workflow) & Key('IsActive').eq("true")
            )

        # Extract the items from the query response
        items = response.get('Items', [])
        # Return the retrieved items
        result = {
            'data': items,
            'service_type': service_type
        }

        logger.debug(f"Fetched Prompts: {result}")
        return result

    except Exception as e:
        # Return the error message in case of an exception
```

```python
lambda_function.py
def retrieve_prompts_by_channel_and_intent(table, event):
    try:
        else:

            response = table.query(
                IndexName='WorkflowIndex',
                KeyConditionExpression=Key('Workflow').eq(workflow) & Key('IsActive').eq("true")
            )

        # Extract the items from the query response
        items = response.get('Items', [])
        # Return the retrieved items
        result = {
            'data': items,
            'service_type': service_type
        }

        logger.debug(f"Fetched Prompts: {result}")
        return result

    except Exception as e:
        # Return the error message in case of an exception

        logger.error(f"Exception in fetching prompts: {str(e)}")

        return str(e)


def format_lex_result(data, event):
```

Logger.debug has been added as logger functions for logging step by step process to aid in debugging

In higher environments, the value of logging logic is set to “info” so that debugging logs are not populated in CloudWatch to manage latency

DB Insert (Global-PromptManagement-DBInsert-LF) –

Dictionary for standardizing the encoding across different input files

```
if __name__ == "__main__":
    table_name = sys.argv[1] + "-Admin-Console-Gen-Prompts-Table"
    recent_file_commits = sys.argv[2]

    print(f" initial data into table {table_name}")
    print(f"Changed Files: {recent_file_commits}")
    files_encodings = {
        'budgetbilling_prompts_data.csv':'iso-8859-1',
        'ebill_prompts_data.csv':'iso-8859-1',
    }

    recent_file_commits = json.loads(recent_file_commits)
```

Instead of processing all files, only recently committed file to the repository will be taken by the workflow to update the prompts.

Prompts used in interactions are managed as key-value pairs within a centralized Prompt DB, enabling dynamic retrieval based on flow context

Dynamic APIs

Backoff Mechanism - There is a backoff exception handling which is added to the code in api_utils module for the APIs. Suppose the API calls lead to an exception, it will do 2 times retry. Post third try it will throw an exception. The timeout value has been set to 4 seconds for waiting for an API response

Snippet from Code

```
_maintenance_2.csv | api_utils.py | anum_prompts2.csv | budgetbilling_prompts_data.csv | creditcard_prompts.csv | ebill_prompts_data.csv
da-layer > python > api_utils.py
def get_secret():

@backoff.on_exception(
        backoff.expo,
        (requests.exceptions.HTTPError, requests.exceptions.ConnectionError, requests.exceptions.Timeout,
         requests.exceptions.RequestException),
        max_tries=2,
        on_backoff=lambda details: logger.warning(f"Retrying in {details['wait']} seconds.. Due to exception: {details['exception'].__class__.__name__}
        on_giveup=lambda details: logger.error("Giving up after retries")
)
```

Suppose HVCA Auth API does not return any value or empty response, the recovery behavior is set as NO ANI match route and try to authenticate customer again.

Logger.debug has been added as logger functions for logging step by step process to aid in debugging

In higher environments, the value of logging logic is set to “info” so that debugging logs are not populated in CloudWatch to manage latency

1.20. Logging Mechanism – CloudWatch Logs

The AWS cloudwatch logging is enabled for each asset built in HVOS and can be accessed via cloudwatch logs or log groups for monitoring and logging purposes.

Connect Flows Log groups

Dev Env - aws/connect/2xsfnv-ei864-cai-co

Test Env - aws/connect/2xsftv-ei864-cai-co

Lex bots Log Groups

Dev Env - aws/2xsfnv-ei864-lex-bots

Test Env - aws/2xsftv-ei864-lex-bots

Each lambda has a separate dedicated log group created at the lambda creation

1.21. Unit Testing

Confluence link:-

Lex Bot	Unit testing status
Outage-Confirmation-Bot	Completed
Outage-CollectContactNumber-Bot	Completed
Outage-UserDetails-Bot	Completed
Outage-IntentDetection-Bot	Completed

Lambda	Unit testing status
Outage-PromptManagementDBQuery-LF	Completed
Outage-HVCAAuthentication-LF	Completed
Outage-DynamciAPIHandler-LF	Completed

6-Mar-25

Unit Test Cases – Lex (Confirmation Bot: 100% coverage)

Approach

Unit Testing approach for Lex Bots:

- The team has leveraged the Amazon Lex test workbench for unit testing (Intent and Slots) to simulate the conversation and validating responses
- Test data in the test workbench includes test scripts with different input variations to cover multiple scenarios
- Observe the Lex bot's responses and Validate Responses
- Analyze Test Workbench logs for incorrect responses
- Retest after modifications

Unit test evidence

Lex Workbench Input

2XSFNV-EI864-Outage-Confirmation-LX-spanish	2XSFNV-EI864-Outage-Confirmation-LX-spanish	Text	Ready for testing	8	10 days ago
2XSFNV-EI864-Outage-Confirmation-LX-english	2XSFNV-EI864-Outage-Confirmation-LX - confirmation english	Text	Ready for testing	8	11 days ago

Lex Workbench Results

Line #	Conversation #	Source	Input	Expected Output Intent	Expected Output Slot
1	N/A	User	si eso es correcto	intent_Confirmation	slot_Confirmation = Yes
2	N/A	User	si claro	intent_Confirmation	slot_Confirmation = Yes
2	N/A	User	correcto	intent_Confirmation	slot_Confirmation = Yes
3	N/A	User	si eso es verdad	intent_Confirmation	slot_Confirmation = Yes
4	N/A	User	Si	intent_Confirmation	slot_Confirmation = Yes
5	N/A	User	No	intent_Confirmation	slot_Confirmation = No
6	N/A	User	Non	intent_Confirmation	slot_Confirmation = No
7	N/A	User	No en absoluto	intent_Confirmation	slot_Confirmation = No

Spanish

Line #	Conversation #	Source	Input	Expected Output Intent	Expected Output Slot
1	N/A	User	yes	intent_Confirmation	slot_Confirmation = Yes
2	N/A	User	yes, please	intent_Confirmation	slot_Confirmation = Yes
2	N/A	User	yes, proceed	intent_Confirmation	slot_Confirmation = Yes
3	N/A	User	no	intent_Confirmation	slot_Confirmation = No
4	N/A	User	No	intent_Confirmation	slot_Confirmation = No
5	N/A	User	No, please	intent_Confirmation	slot_Confirmation = No
6	N/A	User	i don't want	intent_Confirmation	slot_Confirmation = No
7	N/A	User	i don't need	intent_Confirmation	slot_Confirmation = No

English

60

6-Mar-25

Unit Test Cases – Lex (Collect Contact Number Bot: 100% coverage)

Approach

Unit Testing approach for Lex Bots:

- The team has leveraged the Amazon Lex test workbench for unit testing (Intent and Slots) to simulate the conversation and validating responses
- Test data in the test workbench includes test scripts with different input variations to cover multiple scenarios
- Observe the Lex bot's responses and Validate Responses
- Analyze Test Workbench logs for incorrect responses
- Retest after modifications

Unit test evidence

Lex Workbench Input

Completed on: 14 hours ago Test type: Text	Status: Complete Test set: 2XSFNV-EI864-Outage-CollectContactNumber-LX-Spanish	Tested bot: 2XSFNV-EI864-Outage-CollectContactNumber-LX Tested alias: Dev_Spanish	Endpoint: NonStreaming Language: Spanish (US)
Completed on: 14 hours ago Test type: Text	Status: Complete Test set: 2XSFNV-EI864-Outage-CollectContactNumber-LX-English	Tested bot: 2XSFNV-EI864-Outage-CollectContactNumber-LX Tested alias: Dev_English	Endpoint: NonStreaming Language: English (US)

Lex Workbench Results

CollectContactNumber-Lex-Spanish

Spanish

CollectContactNumber-Lex-English

English

61

6-Mar-25

Unit Test Cases – Lex (User Details Bot: 100% coverage)

Approach

Unit Testing approach for Lex Bots:

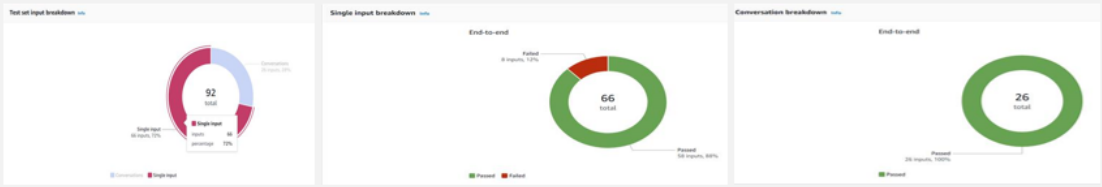
- The team has leveraged the Amazon Lex test workbench for unit testing (Intent and Slots) to simulate the conversation and validating responses
- Test data in the test workbench includes test scripts with different input variations to cover multiple scenarios
- Observe the Lex bot's responses and Validate Responses
- Analyze Test Workbench logs for incorrect responses
- Retest after modifications

Unit test evidence

Lex Workbench Input

Completed on: 14 hours ago	Status: Complete	Tested bot: 2XSFNV-EI864-Outage-CollectContactNumber-LX	Endpoint: NonStreaming
Test type: Text	Test set: 2XSFNV-EI864-Outage-CollectContactNumber-LX-Spanish	Tested alias: Dev_Spanish	Language: Spanish (US)
Completed on: 10 minutes ago	Status: Complete	Tested bot: 2XSFNV-EI864-Outage-UserDetails-LX	Endpoint: NonStreaming
Test type: Text	Test set: 2XSFNV-EI864-Outage-UserDetails-LX-English	Tested alias: Dev_English	Language: English (US)

Lex Workbench Results



UserDetails-Lex-Spanish

Spanish

English

UserDetails-Lex-English

62

3-Mar-25

Unit Test Cases – Lex (Outage Intent Detection Bot:100% coverage)

Approach

Unit Testing approach for Lex Bots:

- The team has leveraged the Amazon Lex test workbench for unit testing (Intent and Slots) to simulate the conversation and validating responses
- Test data in the test workbench includes test scripts with different input variations to cover multiple scenarios
- Observe the Lex bot's responses and Validate Responses
- Analyze Test Workbench logs for incorrect responses
- Retest after modifications

Unit test evidence

Lex Workbench Input

Completed on 9 minutes ago	Status Complete	Tested bot 2XSFNV-EI864-Outage-IntentDetection-LX	Endpoint NonStreaming
Test type Text	Test set 2XSFNV-EI864-Outage-IntentDetection-LX-	Tested alias Dev_Spanish	Language Spanish (US)
Completed on 2 minutes ago	Status Complete	Tested bot 2XSFNV-EI864-Outage-IntentDetection-LX	Endpoint NonStreaming
Test type Text	Test set 2XSFNV-EI864-Outage-UserDetails-LX-English	Tested alias Dev_English	Language English (US)

Lex Workbench Results

Intent Dection - Spanish

Spanish

English

Intent Detection - English

63

6-Mar-25

Unit Test Cases – Lambda (Prompt Mgmnt: 18 Test/94% Coverage)

Approach

Unit Testing approach for Lambda:

- Team wrote test cases using py test which uses unit test, mock and moto python library
- Team wrote functions each of the test cases and each function will validate whether it's a positive or negative scenario (Assertions in the form of Expected versus Actual)

Example –

```
def test_valid_phone_number(self):
    self.assertEqual(get_phone_number_validity("+11234567890"), "1234567890")
    self.assertEqual(get_phone_number_validity("911234567890"), "1234567890")
```

- Based on how much code is being covered in the lambda function, Test results will be generated

```
---------- coverage: platform win32, python 3.11.9
Name                    Stmts   Miss  Cover
-------------------------------------------
lambda_function.py        148     16    89%
test_lambda_function.py    47      1    98%
test_py.py                106      1    99%
-------------------------------------------
TOTAL                     301     18    94%

============================ 16 passed in 0.69
Finished running tests!
```

Modules inside the code package are tested separately

Overall Test summary

Unit test evidence

File Name	Test Class	Test Cases	Description
test_lambda_function.py	TestLambdaFunction	test_get_prompts_in_contact_attributes	Retrieves prompts from contact attributes
		test_update_prompts_in_contact_attributes	Updates prompts in contact attributes
		test_retrieve_prompts_by_channel_and_intent	Fetches prompts based on channel and intent
		test_format_lex_result	Formats the result from Lex bot
		test_format_connect_result	Formats the result for Amazon Connect
test_py.py	TestLambdaFunction	test_get_prompts_in_contact_attributes	Retrieves prompts from contact attributes
		test_get_prompts_in_contact_attributes_error	Handles errors when fetching prompts from contact attributes
		test_update_prompts_in_contact_attributes	Updates prompts in contact attributes
		test_update_prompts_in_contact_attributes_error	Handles errors when updating contact attributes
		test_retrieve_prompts_by_channel_and_intent_connect	Retrieves prompts for channel and intent in Connect
		test_retrieve_prompts_error	Handles errors when retrieving prompts
		test_replace_placeholder	Replaces placeholders in prompts
		test_replace_placeholder_missing_key	Handles missing keys in placeholder replacement
		test_trim_trailing_spaces	Trims unnecessary spaces in text
		test_lambda_handler_connect_success	Tests successful execution of the Lambda handler in Connect
		test_lambda_handler_connect_error	Handles errors in the Lambda handler for Connect

64

This page presents a sample of unit test cases conducted to date

FPL.

Unit Test Cases – Lambda (HVCA Auth –114 test /83% Coverage)

3-Mar-25

Approach

Unit Testing approach for Lambda:

- Team wrote test cases using py test which uses unit test, mock and moto python library
- Team wrote functions each of the test cases and each function will validate whether it's a positive or negative scenario (Assertions in the form of Expected versus Actual)

Example –

```
def test_valid_phone_number(self):
    self.assertEqual(get_phone_number_validity("+11234567890"), "1234567890")
    self.assertEqual(get_phone_number_validity("911234567890"), "1234567890")
```

- Based on how much code is being covered in the lambda function, Test results will be generated

```
---------- coverage: platform win32, python 3.11.9-
Name                          Stmts   Miss  Cover
-------------------------------------------------
api_client.py                    33      2
api_utils.py                     85     39
lambda_function.py              205      8
mock_responses.py                 1      0   1
placeholder_handler.py           72     62
test_api_client.py               64      1  98%
test_api_utils.py                16      1
test_lambda_functions.py        209      1
-------------------------------------------------
TOTAL                           685    114  83%
```

Modules inside the code package are tested separately

Overall Test summary

Unit test evidence

File	Test Class	Test Cases
test_lambda_functions.py	TestCustomerDataFunctions	test_get_customer_data_with_phone
		test_get_customer_data_with_account
		test_get_customer_data_empty_output
		test_get_customer_data_no_phone_or_account
		test_get_customer_data_chat_channel
		test_get_customer_data_fallback_to_mock
		test_get_customer_data_mock_fallback_error
		test_get_mock_user_info_by_phone
		test_get_mock_user_info_by_account
		test_get_mock_user_info_no_match
		test_calculate_confidence_score_high
		test_calculate_confidence_score_medium
		test_calculate_confidence_score_low
		test_check_address_for_close_match_found
		test_check_address_for_close_match_not_found
		test_check_num_of_props_and_validate_single_address
		test_check_num_of_props_and_validate_single_last_name_start
		test_check_num_of_props_and_validate_single_last_name
		test_check_num_of_props_and_validate_multiple
		test_extract_area_code_valid
		test_extract_area_code_invalid
		test_extract_numeric_part_valid
		test_extract_numeric_part_invalid
		test_concatenate_address
		test_separate_letters_from_word
		test_lambda_handler_ani_success
		test_lambda_handler_match_phone_number
		test_lambda_handler_match_account_number
		test_lambda_handler_match_address
		test_lambda_handler_unrecognized_auth_type
		test_lambda_handler_no_customer_data
		test_lambda_handler_exception

65

This page presents a sample of unit test cases conducted to date.

3-Mar-25

Unit Test Cases – Lambda (Dynamic API Handler: 80 Test/81% Coverage)

Approach

Unit Testing approach for Lambda:

- Team wrote test cases using py test which uses unit test, mock and moto python library
- Team wrote functions each of the test cases and each function will validate whether it's a positive or negative scenario (Assertions in the form of Expected versus Actual)

Example –

```
def test_valid_phone_number(self):
self.assertEqual(get_phone_number_validity("+11234567890"), "1234567890")
self.assertEqual(get_phone_number_validity("911234567890"), "1234567890")
```

- Based on how much code is being covered in the lambda function, Test results will be generated

```
---------- coverage: platform win32, python 3.11.9-final-0 -
Name                    Stmts   Miss  Cover
-------------------------------------------
api_client.py              33      2    94%
api_utils.py               85     59    31%
lambda_function.py        142     16    89%
mock_responses.py           3      0   100%
test_api_client.py         64      1    98%
test_api_utils.py          16      1    94%
test_lambda_function.py    76      1    99%
-------------------------------------------
TOTAL                     419     80    81%
```

Modules inside the code package are tested separately

Overall Test summary

Unit test evidence

File	Test Class	Test Cases	Description
test_api_client.py	TestAPIClient	test_post_success	Unit test cases for API calls capturing success and failures of the code modules required to run the APIs (GET/POST/DELETE/Put)
		test_get_success	
		test_put_success	
		test_delete_success	
		test_get_failure	
		test_post_server_error	
test_api_utils.py	TestPhoneNumberValidity	test_valid_phone_number	Validation of phone numbers
		test_invalid_phone_number_too_short	Validation of phone numbers if the digits are less than 10
		test_invalid_phone_number_non_numeric	Validation of phone numbers if the digits are non-numeric
		test_invalid_phone_number_long_country_code	Validation of phone numbers if they have long country codes
		test_invalid_phone_number_missing_local_number	Validation of phone numbers if they have 10 digit number but its an invalid number
test_lambda_function.py	TestOutageAPI	test_customer_auth_with_phone_number	Unit test of module for Authentication with Phone number
		test_customer_auth_with_account_number	Unit test of module for Authentication with Account number
		test_outage_inquiry_success	Unit test to hit the Outage Inquiry API when successfully run
		test_outage_inquiry_null_dates	Unit Test for Outage Inquiry API call when the input parameter (dates) are NULL
		test_ticket_submission_success	Unit testing the Ticket submission API call when run successfully
		test_lambda_handler_customer_auth	Unit testing the Authentication API call when run successfully
		test_lambda_handler_invalid_flow	Unit test cases for failure to hit Auth API as inputs are invalid
		test_lambda_handler_multiple_functions	Unit test case when we multile API executed together (Outage inquiry+HVCA Auth)

This page presents a sample of unit test cases conducted to date.

24-Mar-25

Unit Test Cases – Lambda

Approach

Unit Testing approach for Lambda:

- Team wrote test cases using py test which uses unit test, mock and moto python library
- Team wrote functions each of the test cases and each function will validate whether it's a positive or negative scenario (Assertions in the form of Expected versus Actual)

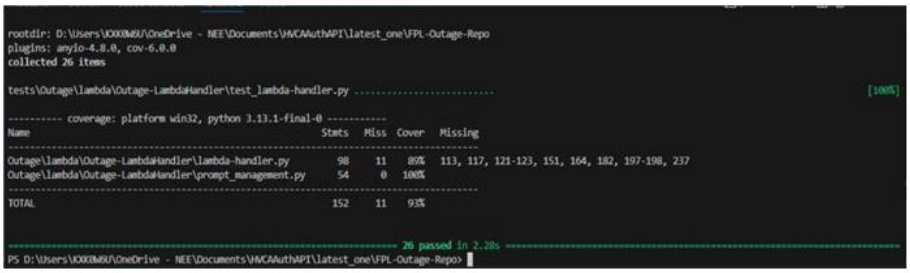
Example –

```
def test_valid_phone_number(self):
    self.assertEqual(get_phone_number_validity("+11234567890"), "1234567890")
    self.assertEqual(get_phone_number_validity("911234567890"), "1234567890")
```

- Based on how much code is being covered in the lambda function, Test results will be generated

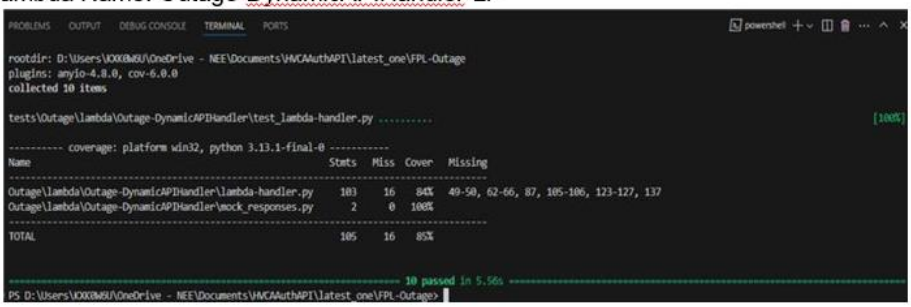
Unit test evidence

Lambda Name: Outage-LambdaHandler-LF



```
rootdir: D:\Users\KXK8W8U\OneDrive - NEE\Documents\HVCAAuthAPI\latest_one\FPL-Outage-Repo
plugins: anyio-4.8.0, cov-6.0.0
collected 26 items

tests\Outage\lambda\Outage-LambdaHandler\test_lambda-handler.py ..........................                    [100%]

---------- coverage: platform win32, python 3.13.1-final-0 -----------
Name                                                          Stmts   Miss  Cover   Missing
-------------------------------------------------------------------------------------------
Outage\lambda\Outage-LambdaHandler\lambda-handler.py             98     11    89%   113, 117, 121-123, 151, 164, 182, 197-198, 237
Outage\lambda\Outage-LambdaHandler\prompt_management.py           54      0   100%
-------------------------------------------------------------------------------------------
TOTAL                                                           152     11    93%


============================================================ 26 passed in 2.28s =============================================================
PS D:\Users\KXK8W8U\OneDrive - NEE\Documents\HVCAAuthAPI\latest_one\FPL-Outage-Repo>
```

Lambda Name: Outage-DynamicAPIHandler-LF



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                                    powershell

rootdir: D:\Users\KXK8W8U\OneDrive - NEE\Documents\HVCAAuthAPI\latest_one\FPL-Outage
plugins: anyio-4.8.0, cov-6.0.0
collected 10 items

tests\Outage\lambda\Outage-DynamicAPIHandler\test_lambda-handler.py ..........                                  [100%]

---------- coverage: platform win32, python 3.13.1-final-0 -----------
Name                                                          Stmts   Miss  Cover   Missing
-------------------------------------------------------------------------------------------
Outage\lambda\Outage-DynamicAPIHandler\lambda-handler.py         103     16    84%   49-50, 62-66, 87, 105-106, 123-127, 137
Outage\lambda\Outage-DynamicAPIHandler\mock_responses.py           2      0   100%
-------------------------------------------------------------------------------------------
TOTAL                                                           105     16    85%


============================================================ 10 passed in 5.56s =============================================================
PS D:\Users\KXK8W8U\OneDrive - NEE\Documents\HVCAAuthAPI\latest_one\FPL-Outage>
```

67

24-Mar-25

Unit Test Cases – Lambda

Approach

Unit Testing approach for Lambda:

- Team wrote test cases using py test which uses unit test, mock and moto python library
- Team wrote functions each of the test cases and each function will validate whether it's a positive or negative scenario (Assertions in the form of Expected versus Actual)

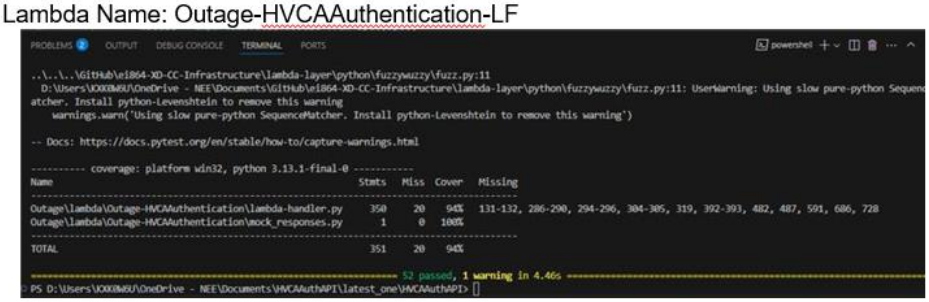
Example –

```
def test_valid_phone_number(self):
self.assertEqual(get_phone_number_validity("+11234567890"), "1234567890")
self.assertEqual(get_phone_number_validity("911234567890"), "1234567890")
```

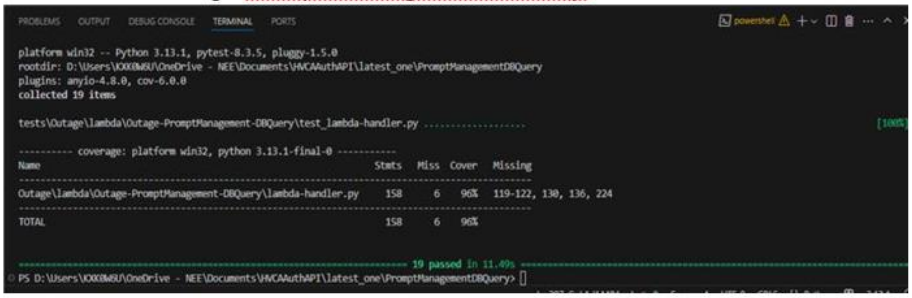
- Based on how much code is being covered in the lambda function, Test results will be generated

Unit test evidence

Lambda Name: Outage-HVCAAuthentication-LF



Lambda Name: Outage-PromptManamgementDBQuery-LF



2. Reporting – Kinesis Data Streams

2.1. Reporting Overview

The data generated from the Conversational channels for the AWS Connect and Lex Applications will also serve as a data source for the Reporting. The data provisioning strategy will facilitate access to Amazon data to enable existing reporting as well as new insights from the data generated.
Following are the accesses that will facilitate the reporting -

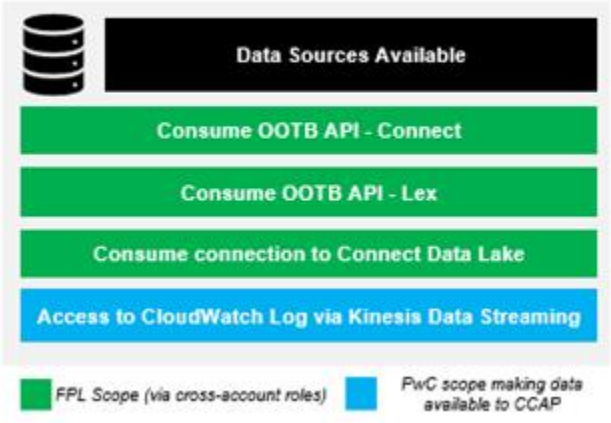
Access to Out of the box reports generated by AWS

- Amazon Lex - Measuring business performance with Analytics
- Amazon Connect - Real-time and historical metrics, dashboards, and reports in Amazon Connect
- Access to AWS Out of the box generated data via API from AWS
- Access to native Amazon Lex API – (Source – Link)
- Access to native Amazon Connect API (Source – Link)
- Access to Amazon Analytics Data Lake (Source – Link)

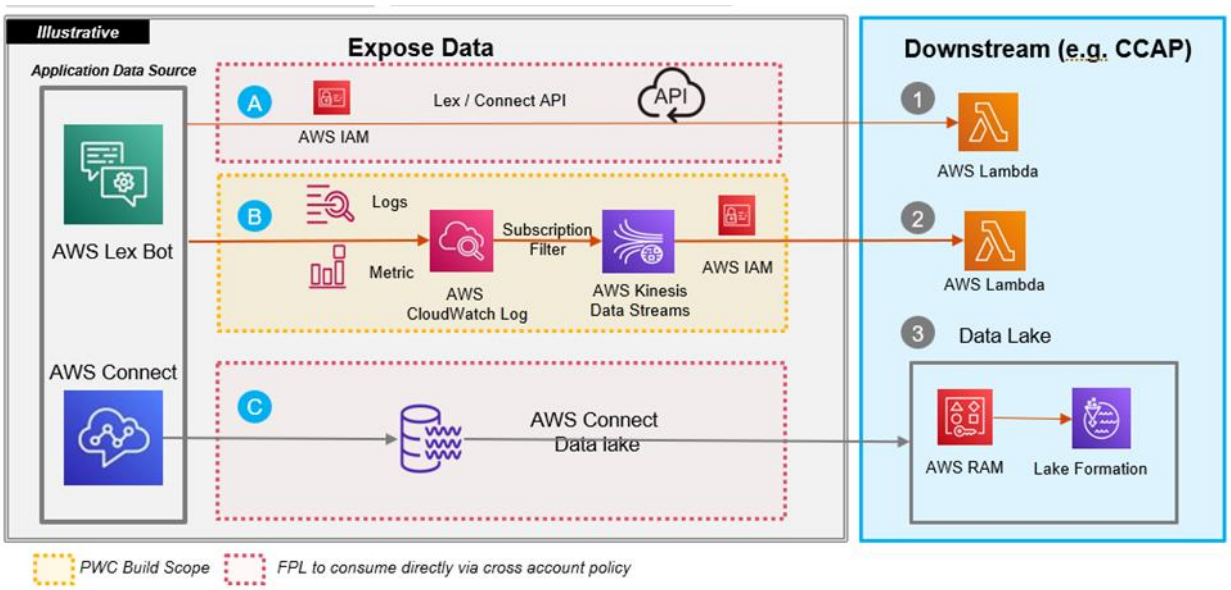
Access to Raw Data

Setup CloudWatch Logs for AWS Connect and Lex (CloudWatch logs will include the reporting markers which are custom added to the flows by the business for facilitating reporting and metrics above and beyond what is already available)

Expose the streaming data through AWS Kinesis for continuous reporting (Source – Link)



Reporting Architecture
The following architecture explains on the high-level on the data exposure strategy



Sourcing the Data

- Access to OOTB API
- Access to native Amazon Lex Metric OOTB API
- Access to native Amazon Connect Metric OOTB API
- Access to CloudWatch Log
- Setup CloudWatch Logs for AWS Connect and Lex
- Expose the streaming data through AWS Kinesis
- Access to Connect Data Lake
- In the AWS Connect Analytics tool, share the data lake to target account
- Target account will accept the request in AWS RAM and build the Lakeformation tables

How Downstream to access data

Access data from API:

- In the CCAS source data end, configure the IAM role permissions to the target AWS account and lambda service
- In the target AWS account, setup the Lambda to access the metric data API through Python Boto3 lib

Access Data from Kinesis Data Stream:

- In the CCAS source data end, configure the IAM role permissions to the target AWS account and lambda service
- In the target AWS account, setup the lambda with Kinesis data stream configured as the event trigger

Access Data from AWS Connect Data Lake:

- Share the data lake data types from the AWS Connect Analytics Tools
- In the target AWS account, accept the resource share request
- Setup the Lakeformation database and tables, query the data from Athena

Reporting Custom Reporting Markers -

- From Business for HVOS - MasterSheet_Reporting Markers_ConvoAI_HVOS_2_19.xlsx (sheet - HVOS (only AUTH))
- Development Team inputs on Marker Placements - Reporting Markers_Outage_Draft_10_25.xlsx (sheet - HVOS (only AUTH))

Reporting Kinesis Data Streams

AWS Dev:

- Cross account IAM Role: arn:aws:iam::059221074852:role/NEE-CO972-Reporting-Role
- CloudWatch Logs:
- Connect Logs: arn:aws:logs:us-east-1:059221074852:log-group:/aws/connect/2xsfnv-cai-co:*
- Logs subscription filter: 2xsfnv-ei864-co-sub-filter

Lex Logs:

- arn:aws:logs:us-east-1:059221074852:log-group:/aws/lex/2xsfnv-ei864-lex-bots:*
- Logs subscription filter: 2xsfnv-ei864-lex-sub-filter

Kinesis:

- Connect Logs Kinesis: arn:aws:kinesis:us-east-1:059221074852:stream/2xsfnv-ei864-co-logs-kds 2xsfnv-ei864-connect-ctr-kds

Lex Logs Kinesis: arn:aws:kinesis:us-east-1:059221074852:stream/2xsfnv-ei864-lex-logs-kds

AWS QA:

- Cross account IAM Role: arn:aws:iam::905418007800:role/NEE-CO972-Reporting-Role

- CloudWatch Logs: arn:aws:logs:us-east-1:905418007800:log-group:/aws/connect/2xsftv-cai-co:*
- Logs subscription filter: 2xsftv-ei864-co-sub-filter: arn:aws:logs:us-east-1:905418007800:log-group:/aws/lex/2xsftv-ei864-lex-bots:*
- Logs subscription filter: 2xsftv-ei864-lex-sub-filter
- Kinesis: arn:aws:kinesis:us-east-1:905418007800:stream/2xsftv-ei864-co-logs-kds arn:aws:kinesis:us-east-1:905418007800:stream/2xsftv-ei864-lex-logs-kds

3. Outbound, Telephony and Routing

3.1. Scope Of work

The scope of work covers the end-to-end call handling process across all call legs, starting from AT&T, transitioning through Amazon Connect, and finally transferring to Cisco using PSTN (Public Switched Telephone Network) for call routing. It includes configuring Amazon Connect to receive inbound calls from AT&T's telephony network and establishing PSTN connectivity with Cisco to enable seamless call bridging. The IVR in Amazon Connect will be designed and developed to guide callers through prompts and routing logic, ensuring efficient call handling before initiating the transfer to Cisco. Testing and optimization will ensure minimal latency, high call quality, and resilience against failures. The system will include monitoring and reporting tools to track call flows, measure success rates, and detect anomalies. Comprehensive documentation, along with training for support teams, will enable ongoing system management and troubleshooting. This holistic approach ensures a reliable and customer-centric call transfer process across all call legs using PSTN connectivity.

PWC will assess the technical feasibility and challenges of PSTN bridge transfer architecture. The objective is to design a solution that enables seamless transfer of calls from Amazon Connect, a cloud-based contact center service, to a Cisco-based telephony system, ensuring smooth and efficient communication.

The primary goal of this scope of work is to detail the technical design and implementation strategy for establishing connectivity for customers connecting from local area zones to AT&T service provider to Cisco environment and incase of HVOS scenario using PSTN bridge transfer pooling architecture between Amazon Connect and a Cisco environment. This includes ensuring that the PSTN bridge transfer functionality is reliable, maintains call quality, passes on the call to Cisco and integrates effectively with both systems.

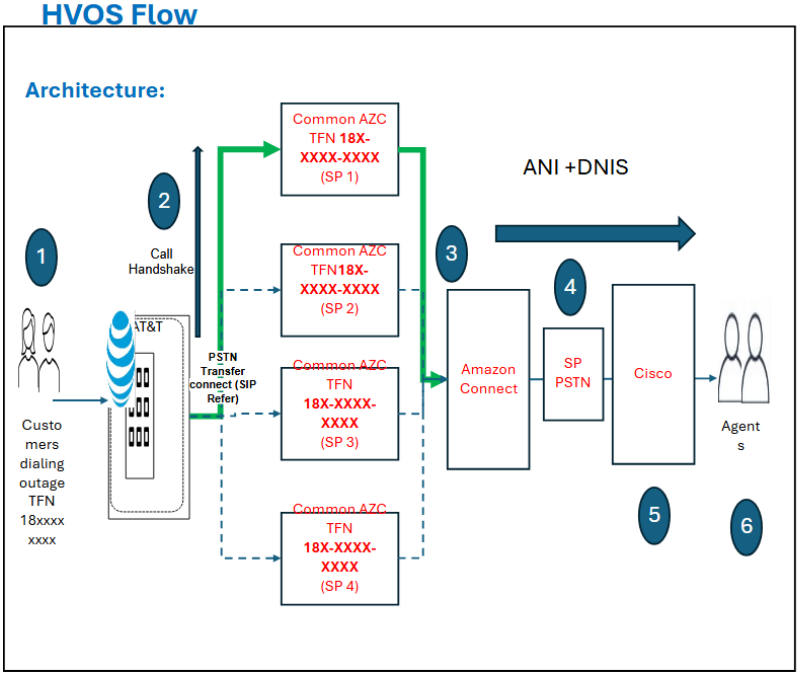
3.2. Requirements Overview

https://confluence.nexteraenergy.com/pages/viewpage.action?pageId=604344722

3.3. Production Call Flow

HVOS Flow

Architecture:

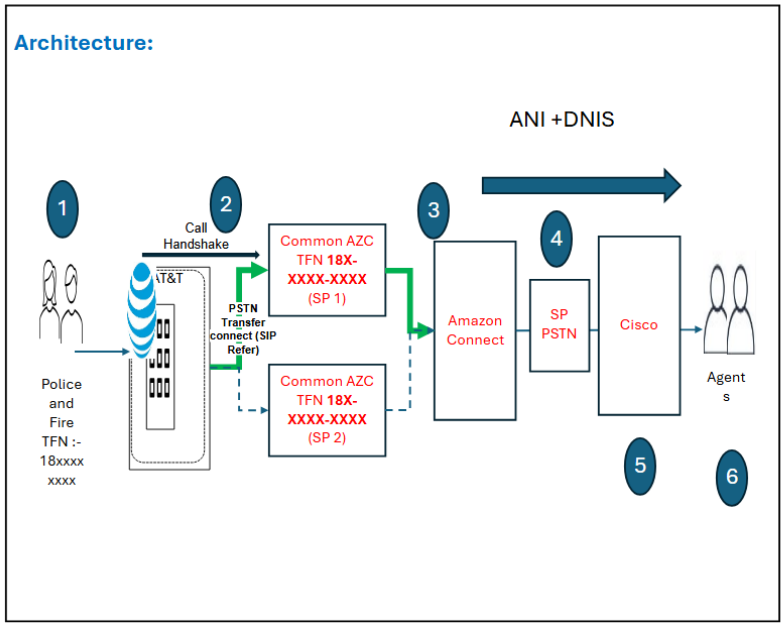


Call Flow:-

1. **Customer Dial-In**: Customers call a toll free number.
2. **AT&T Transfer Connect**: AT&T Transfer Connect, to facilitate the seamless transfer of a call from one AT&T Toll-Free Number (TFN) to another SP TFN. This will be achieved using ADT signaling in conjunction with Transfer Connect, enabling a proper handshake between the originating and destination TFNs.
3. **Amazon Connect Interaction**: AZC handles the calls, providing self-service options through a bot and third-party applications.
4. **IVR & Call Transfer**: After the IVR interaction, calls are transferred based on the customer request to Cisco agents for specific scenarios like Wiredown, Police/Fire pin, REP options, etc., over PSTN.
5. **Cisco UCCE Queuing**: In Cisco UCCE, the call enters a queue. The assigned agent is determined by their mapping to the queue.
6. **Agent CRM Screen Pop**: Based on the end customer's ANI, the CRM screen pops up on the agent's desktop

Police and Fire Emergency

Architecture:

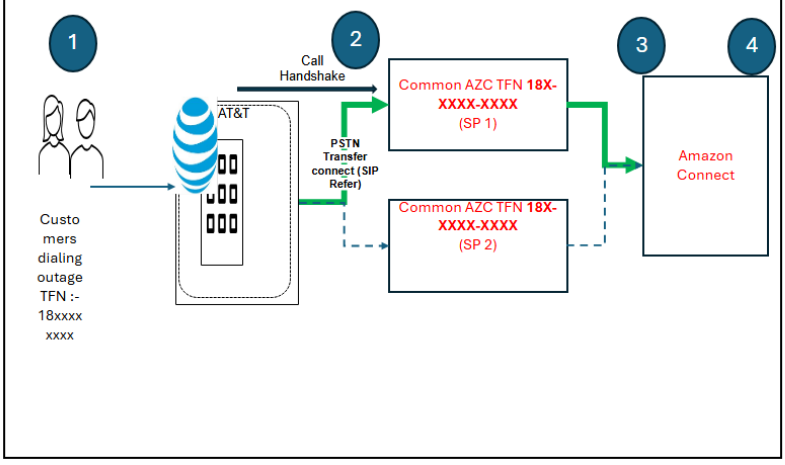


Call Flow:-

1. **Customer Dial-In**: Customers call a toll free number.
2. **AT&T Transfer Connect**: AT&T Transfer Connect, to facilitate the seamless transfer of a call from one AT&T Toll-Free Number (TFN) to another SP TFN. This will be achieved using ADT signaling in conjunction with Transfer Connect, enabling a proper handshake between the originating and destination TFNs.
3. **Amazon Connect Interaction**: AZC handles the calls..
4. **Call Transfer**: Calls are transferred to Cisco agents over PSTN.
5. **Cisco UCCE Queuing**: In Cisco UCCE, the call enters a queue. The assigned agent is determined by their mapping to the queue.
6. **Agent CRM Screen Pop**: Based on the end customer's ANI, the CRM screen pops up on the agent's desktop

Storm Status line

Architecture:



Call Flow:-

1. **Customer Dial-In**: Customers call a toll free number.
2. **AT&T Transfer Connect**: AT&T Transfer Connect, to facilitate the seamless transfer of a call from one AT&T Toll-Free Number (TFN) to another SP TFN. This will be achieved using ADT signaling in conjunction with Transfer Connect, enabling a proper handshake between the originating and destination TFNs.
3. **Amazon Connect Interaction**: AZC handles the calls..
4. **Play Announcement & Disconnect** : A storm status announcement is played, after which the call disconnects.

3.4. DID Allocation

3.4.1. DEV DID

SR No	Intent	Test TFN Number	Amazon Connect DID
1	Outage:-HVOS NW +Legacy Number	833-821-1312	+1 779-548-4573
2	Outage: HVOS NW Number	866-252-6047	+1 786-375-5188
3	Police /Fire Emergency	866-263-9184	+1 786-375-5344
4	Storm status line	888-440-6802	+1 786-375-5428
5	General care	833-919-0940	+1 786-369-1319
6	Account maintenance	To be received	+1 786-375-5524
7	Move out	To be received	+1 786-375-5133
8	Billing & Payment Programs: Auto Bill Pay / Budget Bill / Fixed Rate /eBill	To be received	+1 786-369-5844
9	General Retrieval	To be received	+1 786-369-1319

3.4.2. Test DID

Sr. No.	Test Scenario	AT&T TFN	AWS DID	AWS DID Address
1	Outage: HVOS NW +Legacy Number	833-919-0943	18284835326	US/Canda/PR/VI or Other? : US Home NPA (area code) : 828 Number : +18284835326 Street : NA City : Arden State : North Carolina Zip Code : 28704 County : Buncombe
2	General Care - Salesforce Intents (Acct Maint, Move Out...)	866-252-6047	18607068754	US/Canda/PR/VI or Other? : US Home NPA (area code) : 860 Number : +18607068754 Street : NA City : Hartford State : New York Zip Code : 06152 County : United States
3	Police /Fire Emergency	866-263-9184	18607068766	US/Canda/PR/VI or Other? : US Home NPA (area code) : 860 Number : +18607068766 Street : NA

				City : Hartford State : New York Zip Code : 06152 County : United States
4	Storm Status Line	888-440-6802	18432780755	US/Canda/PR/VI or Other? : US Home NPA (area code) : 843 Number : +18432780755 Street : NA City : Charleston State : South Carolina Zip Code : 29401 County : Charleston
5	General Care - HVOS ON	833-919-0940	18328569156	US/Canda/PR/VI or Other? : US Home NPA (area code) : 832 Number : +18328569156 Street : NA City : Houston State : Texas Zip Code : 77002 County : Harris

3.4.3. QA DID

Sr. No.	Scenario	Required	Comment
1	Wire Down	DID	HVOS DID+13322164101, +13473530823 , +18503534003 , +18503534002
2	Wire Down - Agent Call Transfer	DID, CISCO Routing DID	HVOS DID+13322164101, +13473530823 , +18503534003 , +18503534002Cisco: 3862436115
3	Report an Outage (Auth via ANI)	DID	HVOS DID: +13322164101, +13473530823 , +18503534003 , +18503534002
4	Report an Outage (Auth via BA#)	DID	HVOS DID+13322164101, +13473530823 , +18503534003 , +18503534002
5	Get Outage Update (Auth via ANI)	DID	HVOS DID+13322164101, +13473530823 , +18503534003 , +18503534002
6	Get Outage Update (Auth via BA#)	DID	HVOS DID+13322164101, +13473530823 , +18503534003 , +18503534002
7	Police/Fire	DID	Police and Fire Emergency DID+18503534004
8	Storm Line Status	TFN	Storm status line TFN. +18333673908

3.4.4. Production Numbers

Sr. No.	Test Scenario	AT&T TFN	AWS DID
1	Outage: HVOS NW +Legacy Number+ General Care	Production Numbers	+1 833-380-4105 +1 833-380-4116 +1 833-380-4125 +1 833-380-4064 +1 833-380-4059 +1 833-380-4129
2	Police /Fire Emergency	Production Numbers	+18583790295, +18588683714 (TFN to be allocated)

3	Storm Status Line	Production Number	+18583790310 +18588683813 (TFN to be allocated)

3.4.5. AZC flow DID number map

Amazon Connect DID's Dev Env

Number	Description	DID/TFN	Channel	Flow mapped
+1 779-548-4542	HVOS Number 1 - Dev Env - Do not reassign	DID	Voice	2XSFNV-EI864-Outage-Initial-Flow
+1 858-248-8230	CFN - Storm Status line - Do Not reassign	DID	Voice	2XSFNV-EI864-CFN-Outage-Initial-Flow

+1 786-375-5344	Police/Fire Emergency	DID	Voice	2XSFNV-EI864-Global-DirectTransferCasePoliceAndFireEmergency-Flow

3.4.6. Amazon Connect DID's Test Env

Number	Description	DID/TFN	Channel	Flow mapped
+1828483326	HVOS/HVCA	DID	Voice	2XSFTV-EI864-CFN-Outage-Initial-Flow
+18432780755	StormStatusLine	DID	Voice	2XSFTV-EI864-CFN-Outage-Initial-Flow
+18607068764	Police/Fire Emergency	DID	Voice	2XSFTV-EI864-CFN-Outage-EmergencyLine-Policemen-Publix-Flow

3.4.7. Amazon Connect DID's QA Env

Number	Description	DID/TFN	Channel	Flow mapped
+13473530823 +18503534003 +13322164101 +18503534002	HVOS Outage	DID	Voice	2XSFQV-EI864-CFN-Outage-Initial-Flow
+16464993183	StormStatusLine	DID	Voice	2XSFQV-EI864-CFN-Outage-StormStatusLine-Flow
+18503534004	Police/Fire Emergency	DID	Voice	2XSFQV-EI864-CFN-Outage-EmergencyLine-Policemen-Publix-Flow

3.5. Transfer Flow

The table below shows different agent transfer scenarios and their respective Cisco numbers for NW.

- In each scenario, Amazon Connect transfers the call to Cisco if there is a trunk available.
- If no trunk is available, callers will hear wait messages.
- For the Wire Down, No Match, and Rep Option scenarios, if a trunk does not become available within 60 seconds, Amazon Connect informs the caller that a transfer is not possible and ends the call.
- For the Police/Fire scenario, if a trunk does not become available within 120 seconds, Amazon Connect informs the caller that a transfer is not possible and ends the call.

3.6. DNIS:-

3.6.1. Dev Environment

Scenario – Language English	**Number**
Rep Option Eng Legacy	+12392134621
Rep Option Eng NW	+12392626427
Wiredown Eng Legacy	+12392772001
Wiredown Eng Spanish	+12393043060
Unable to ID Account Legacy ENG (No match scenario)	+12393045453
Unable to ID Account NW ENG (No match scenario)	+12393165210
Police and Fire Eng Legacy	+12393165211
Police and Fire Eng NW	+12393340517
Police and Fire Emergency	+18004016199

Language - Spanish	**Number**
Wire Down Legacy Spa	+12398950027
Wire Down NW Spa	+12398950028
Unable to ID Account Legacy Spa (No match scenario)	+12398950029

Unable to ID Account NW Spa (No match scenario)	+13054284034
Rep Option Legacy Spa	+13054284035
Rep Option NW Spa	+13054284036

3.6.2. Production DNIS

BU	Reason For Transfer	Cisco Termination Number (English)	Cisco Termination Number (Spanish)
FPL NW	Police/Fire	833-407-2007	-
	Wire Down	877-825-1813	833-407-2002
	No Match in Account Identification	844-893-9899 (E)	844-893-9900 (S)
	Rep Option	877-825-1820 (E)	877-881-6813 (S)
FPL Legacy	Police/Fire	800-375-2112	-
	Wire Down	877-691-3447	800-811-8040
	No Match in Account Identification	877-861-6901	833-313-2968
	Rep Option	800-811-6996 (E)	800-811-9412 (S)

3.7. Scenarios

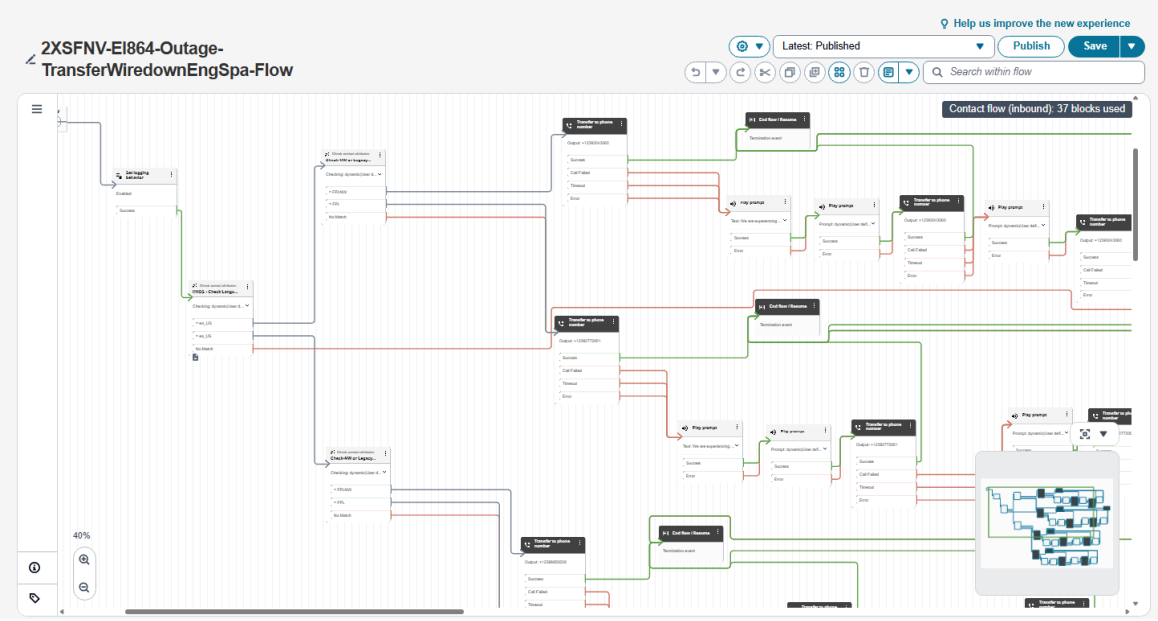
Scenarios	Description
Wire down	Call will get transferred to agent in case, customer opts for wiredown option at the initial menu
Rep Option	Call will get transferred to agent in case customer opts for a report an outage option and then customer opts to speak to an agent
NoMatch	Call will get transferred to agent in case customer is not identified in initial Authentication flow
Police and Fire Pin	Call will get transferred to agent in case customer presses 777 during the call
Police and Fire emergency	A dedicated DID is assigned for emergency line and calls will get transferred directly from this DID t agent in case customers call on this.

Scenarios	Transfer Flows	Exit Flow (Transfer to Agents)
Wire down	2XSFNV-EI864-Outage-TransferWiredownEngSpa-Flow	Outage-ExitIVR-Flow

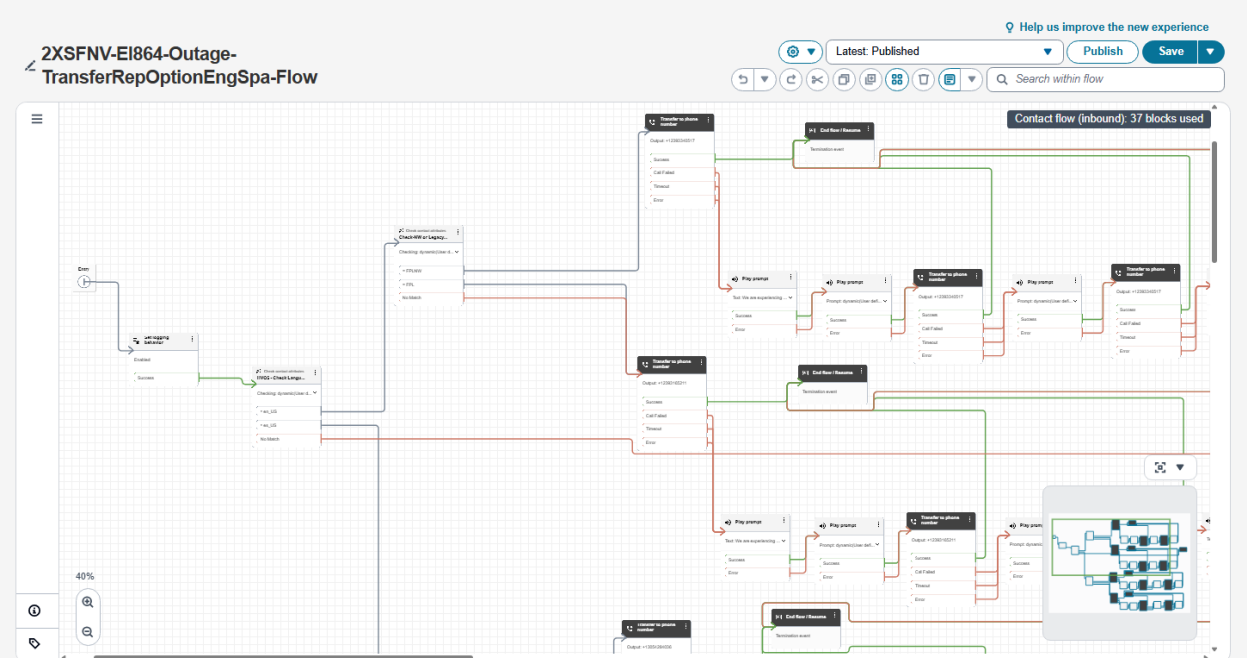
Rep Option	2XSFNV-EI864-Outage-TransferRepOptionEngSpa-Flow	Outage-ExitIVR-Flow
NoMatch	2XSFNV-EI864-Outage-TransferNomatchEngSpa-Flow	Outage-ExitIVR-Flow
Police and Fire	2XSFNV-EI864-Outage-TransferPoliceandFireEngSpa-Flow	Outage-ExitIVR-Flow
Police and Fire Emergency	NA	Global-DirectTransferCasePoliceAndFireEmergency-Flow

Transfer flows by scenario

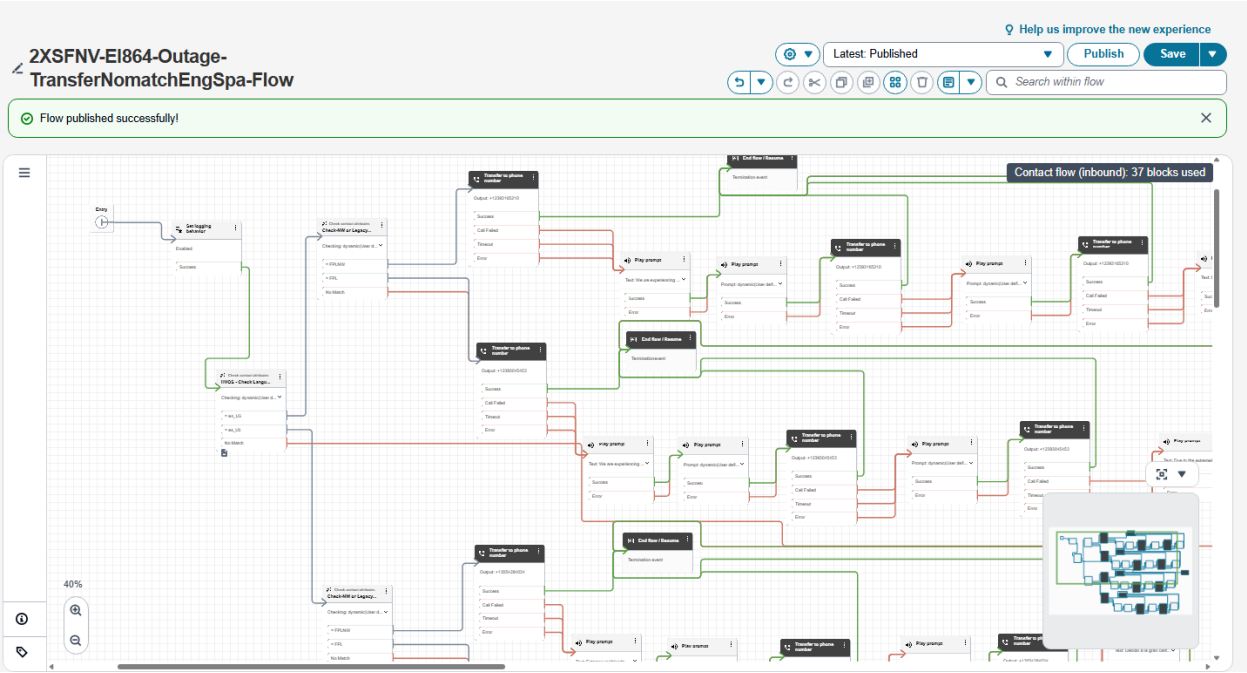
Wiredown Eng and Spanish - 2XSFNV-EI864-Outage-TransferWiredownEngSpa-Flow



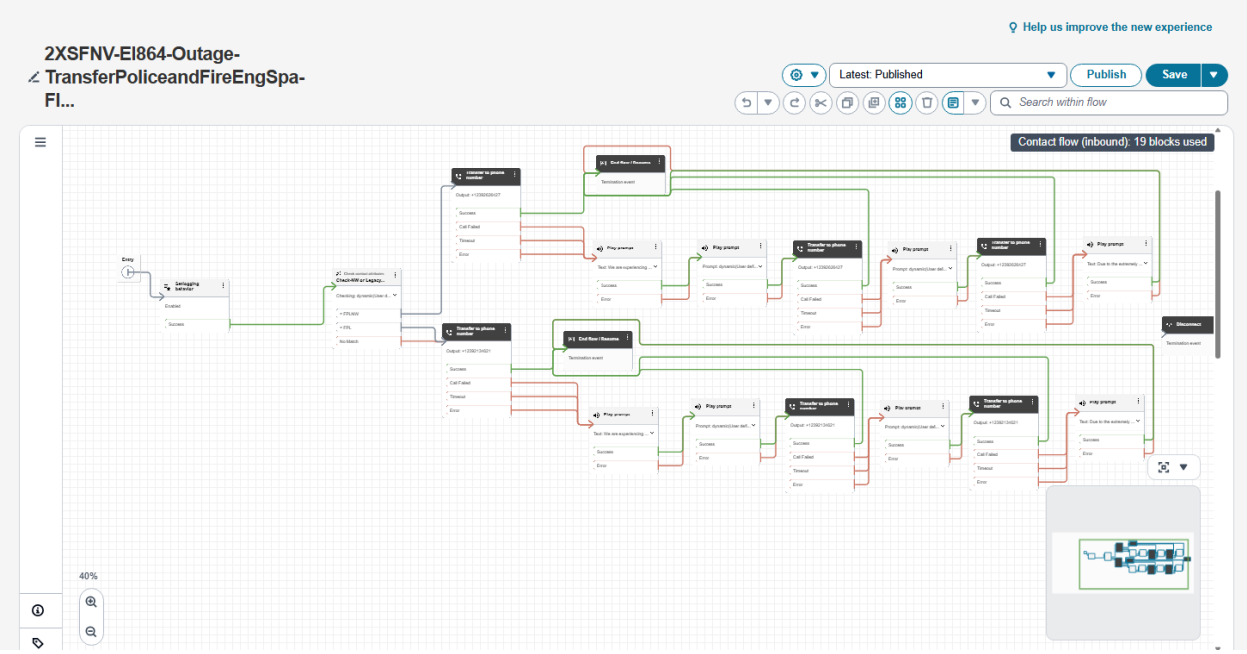
Repoption Eng and Spa - 2XSFNV-EI864-Outage-TransferRepOptionEngSpa-Flow



Nomatch Eng and Spa - 2XSFNV-EI864-Outage-TransferNomatchEngSpa-Flow



PoliceandFire Eng and Spa - 2XSFNV-EI864-Outage-TransferPoliceandFireEngSpa-Flow



All exit points for call transfers to Cisco are set up within the Outage-ExitIVR-Flow. Customers from 'FPLNW' and 'FPLLegacy' will be routed to their respective Cisco DID's based on their selected language, either English or Spanish.

3.8. Logging Mechanism – CloudWatch Logs

The AWS Cloudwatch logging is enabled for each asset built in HVOS and can be accessed via Cloudwatch logs or log groups for monitoring and logging purposes.

Connect Flows Log groups

Dev Env - aws/connect/2xsfnv-ei864-cai-co

Test Env - aws/connect/2xsftv-ei864-cai-co

QA Env – aws/connect/2xsfqv-ei864-cai-co

4. Admin Console

4.1. Introduction

Business teams update configuration data (e.g., HVCA Modes, or ASM etc.) in an Admin Console DynamoDB table via a structured Excel template. The file is uploaded to a designated S3 bucket, and an AWS Lambda function is triggered to process sheets of the uploaded file, process the records and update DynamoDB accordingly.

4.2. Functional Requirements

We are implementing a subset of functional requirements. So, highlighted are those points that are present in the Manual Process of Admin Console for HVOS Release.

Jira Id	Requirements Summary	Requirements Description	Acceptance Criteria
OTRAI-2166	View Active Messages	As an FPL Admin Console user, I want to view active messages on the welcome page, so that I can quickly see all active messages and where they are playing.	1. Given that I am on the welcome interface, then I should see a list of any active messages that are being played on a schedule. 2. Given that there are no active messages, when I view the welcome page, then I should see a message indicating that there are no active messages. 3. Given that I am an IVR Analyst, Developer, or Routing team member, I should have access to the Welcome Page. 4. Given that I am on the Welcome page, I should be able to export the list of active messages by triggering an email to be sent to me with the output of the list.
OTRAI-2188	Manage Prompts	As an FPL Admin Console user, I want to query and update prompts in the Prompt Deployment Application, so that I can manage the audio generated during conversations.	1. Given that I am on the Prompt Manager, I should be able to query from all available prompts based on channel, language, workflow, and block in order to view the prompt details. 2. Given that I am in the Prompt Manager, I should be able to update the text version of ASM, and FLM prompts, and able to make it Active or Inactive and save it (to be updated in the flows immediately).

			5. Given that I attempted to edit a prompt, and there should be data quality checks in place to raise errors for word/free text limitations. 7. Given that I am the Prompt Manager, I should be able to specify other parameters required for upfront messaging (e.g., Area Code, All Caller, Customer Type, Message Type, etc.) 8. Given that I am an IVR Analyst, Developer, or Routing team member, I should have access to the Prompt Manager function.
OTRAI-2158	Set Destinations of Applications	As an FPL Admin Console user, I want to be able to turn on/off applications in the Destination List, so that I can manage the availability and behavior of individual applications effectively.	1. Given that I am in the Application Destination List, then I should see a list of individual applications with options to set available, disconnect, and transfer options for each one. 2. Given that I am in the Application Destination List, when I modify the availability of an application, the action should affect only the selected application. 4. Given that I am an IVR Analyst, Developer, or Routing team member, I should have access to the Application Destination List. 5. Given that I am in the Application Destination List, I should be able to export the list by triggering an email to be sent to me with the output of the list.
OTRAI-2163	List & Add Key Values	As an FPL Admin Console user, I want to see and update key values, so I can manage various contact center configurations such as toggles, flags, and modes.	1. Given that I am on the Key Values page, then I should see a list of existing key values with their names, values, and descriptions. 2. Given that I am on the Key Values page, then I should be able to add a new key value with its corresponding value and description to the existing list. 3. Given that I am on the Key Values page, then I should be able to edit ~~and delete~~ existing key values. 4. Given that I am on the Key Values page, I should be able to see an

			organized and clear segmentations for various categories of key values (e.g. storm controls, HVCA toggles), with the option to easily create new categories. 6. Given that I attempted to add a configuration without populating a required field, then there should be an error alert indicating the required field, and there should be a data quality check validating the data types (boolean, numbers, strings, JSON objects, arrays) and only accept valid entries. 7. Given that I am an IVR Analyst, Developer, or Routing team member, I should have access to the Key Values. 8. Given that I am in the Key Values, I should be able to export the list by triggering an email to be sent to me with the output of the list.
OTRAI-2169	Auditing	As an FPL Admin Console user, I want to have access to message logs, so that I can keep an audit trail of Admin Console activity.	1. Given that I have access to the Admin Console, I should be able to see changes made to any Admin Console function via the DynamoDB audit logs. 2. Given that I am viewing the DynamoDB audit logs, I should be able to see the individual responsible for making changes. 3. Given that I am viewing the DynamoDB audit logs, I should be able to see Admin Console users and their roles (level of access). 4. Given that I am an IVR Analyst, Developer, or Routing team member, I should have access to the Auditing feature.

4.3. System Overview

The Business frequently needs to update values such as HVCA Mode settings (e.g., changing a status from Off to On). To streamline this workflow:

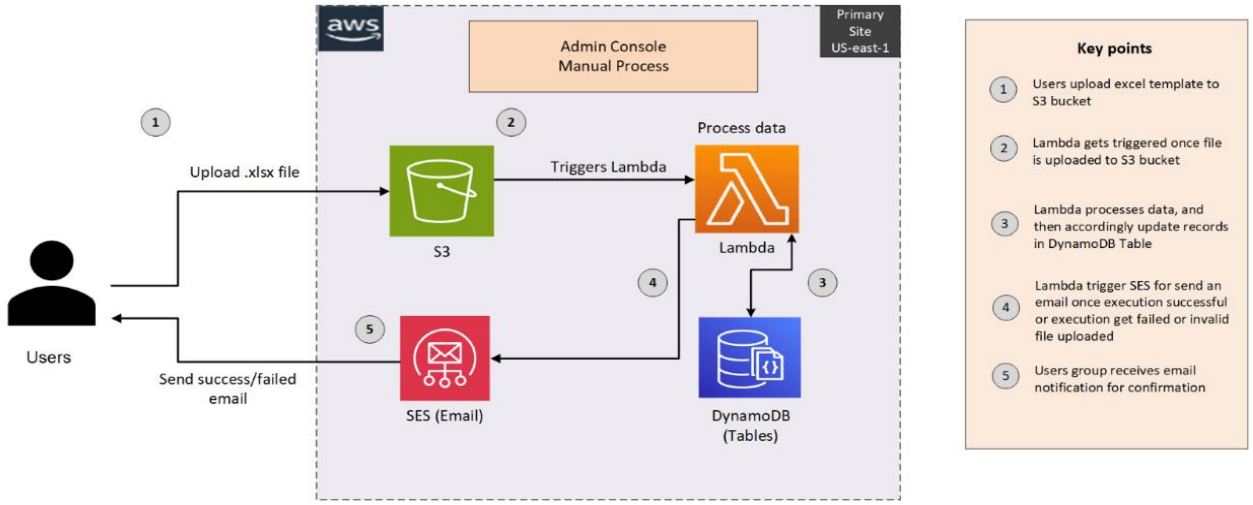
- The designated user from C&T team and IVR Team receives a standard pre-defined **Excel template** from Contact Center development team.

- The users will fill in the data as per the instructions provided (sheet-specific) and upload the file in S3 bucket.
- A Lambda function will trigger as soon as files get uploaded in S3 bucket.
- Lambda processes those data which has **Updated** column marked as **Yes** and performs update operations on the **Admin Console DynamoDB tables**.
- The Users group receives notifications of records updated via email.

4.4. Architecture

4.4.1 Component Diagram

The architecture consists of the components as depicted in below diagram:



Note: Scope of the Admin Console Bot as highlighted above

4.4.2 Data Flow

This section outlines the end-to-end operational steps for executing the Excel-driven update process, from the business team request to backend Lambda execution. Screenshots should be added at each relevant step.

Step-by-Step for Update Process:

Step 1: Designated users from C&T Team updates the pre-defined Excel Template

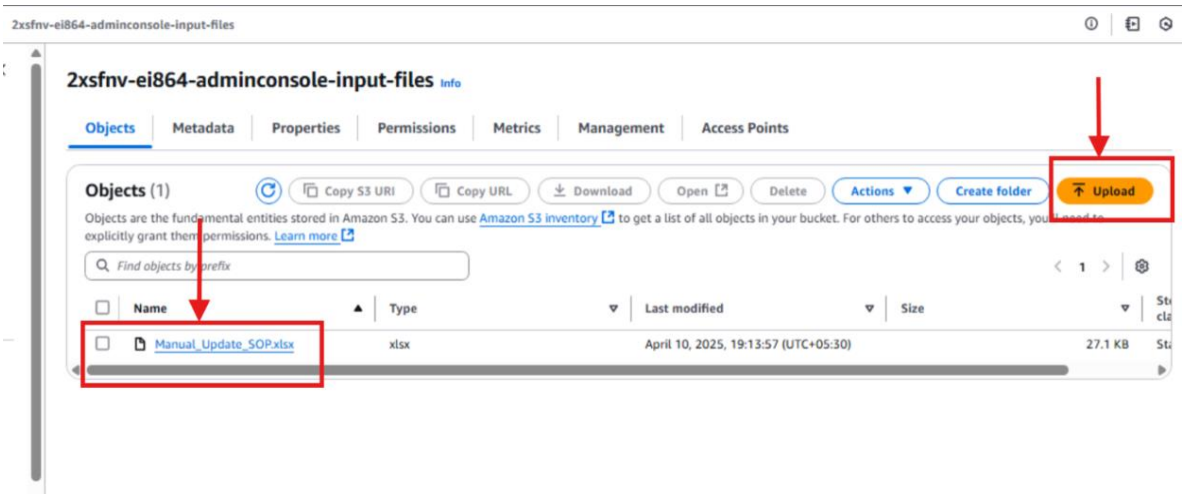
- Use the **pre-approved** **Excel Template** format shared by the development team.
- Ensure all required columns are populated correctly.

- Whenever the records need to change, only those records mark **Updated** column as **Yes** and for the remaining records leave **Updated** column as **No**.
- Validate data (no missing mandatory fields or invalid formats).

[Manual_Update_SOP.xlsx (click to see Excel Template)]

Step 2: The users upload Excel to S3

- Upload excel file to the designated S3 bucket via SFTP Client (WinSCP).
- Follow this Step-by-Step guide (URL) to upload file via SFTP Client.
- **Bucket Name:** 2xsfnv-ei864-adminconsole-input-files
- File Name must be **Manual_Update_SOP.xlsx**



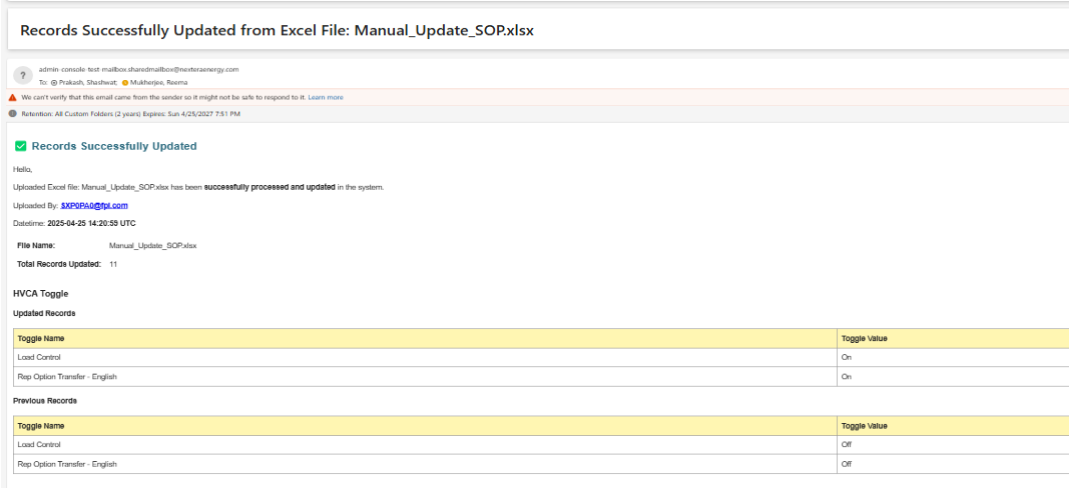
Step 3: Lambda Function triggers automatically

- The associated Lambda function gets triggered once a file is uploaded in S3 Bucket. And processes only those data which has **Updated** column marked as **Yes** in every sheet.
- Lambda Name: ***2XSFNV-EI864-AdminConsole-DBUpdate***

Response: User will get an email once all records have been successfully updated in DynamoDB.

Step 4: Email triggered

- An email is triggered within a minute when records are successfully processed to the group of users which consists of the details like **total number of records**, Timestamp, who uploaded the file and the **list of records** that has been updated.



Records Successfully Updated from Excel File: Manual_Update_SOP.xlsx

admin console test mailbox.sharedmailbox@nexteraenergy.com
To: Prakash, Shashwat; Mukherjee, Reema
We can't verify that this email came from the sender so it might not be safe to respond to it. Learn more
Retention: All Custom Folders (2 years) Expires: Sun 4/25/2027 7:51 PM

Records Successfully Updated

Hello,

Uploaded Excel file: Manual_Update_SOP.xlsx has been **successfully processed and updated** in the system.

Uploaded By: SXP0PA0@fpl.com

Datetime: 2025-04-25 14:20:59 UTC

File Name: Manual_Update_SOP.xlsx

Total Records Updated: 11

HVCA Toggle

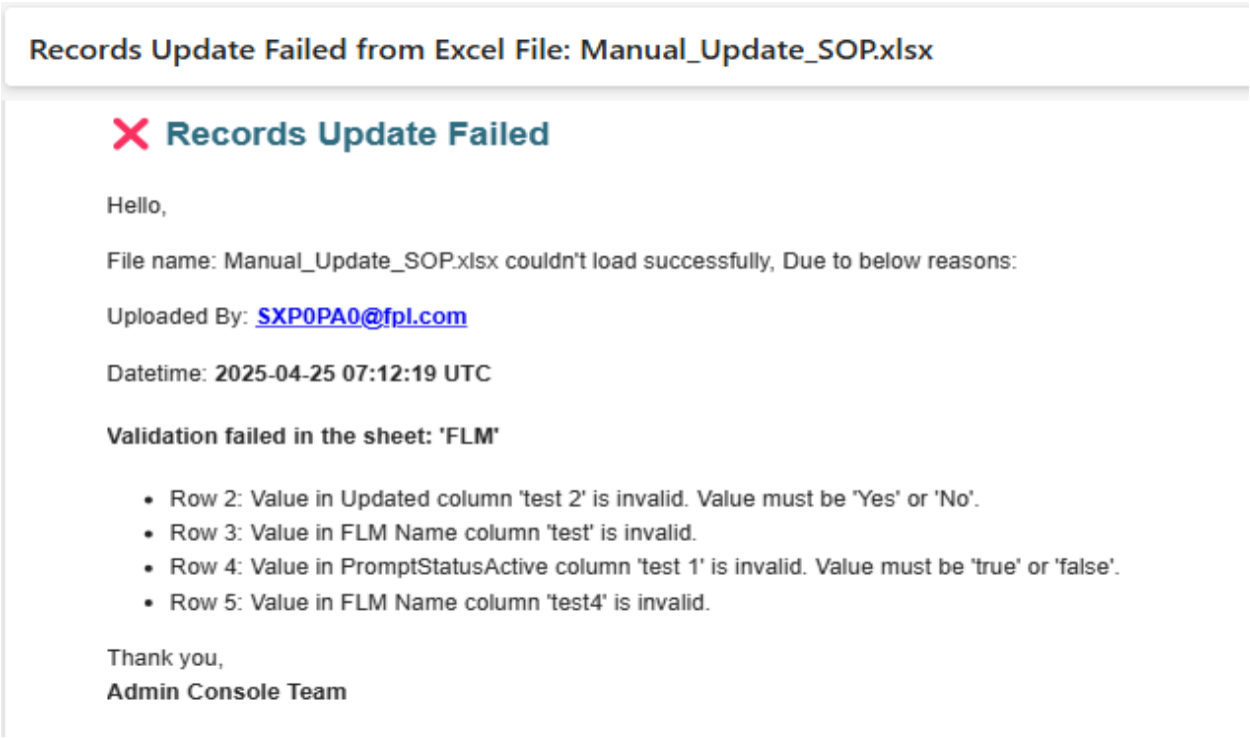
Updated Records

Toggle Name	Toggle Value
Load Control	On
Rep Option Transfer - English	On

Previous Records

Toggle Name	Toggle Value
Load Control	Off
Rep Option Transfer - English	Off

- An Email is also triggered to group users when **data validation failed**.



Records Update Failed from Excel File: Manual_Update_SOP.xlsx

❌ **Records Update Failed**

Hello,

File name: Manual_Update_SOP.xlsx couldn't load successfully, Due to below reasons:

Uploaded By: SXP0PA0@fpl.com

Datetime: **2025-04-25 07:12:19 UTC**

Validation failed in the sheet: 'FLM'

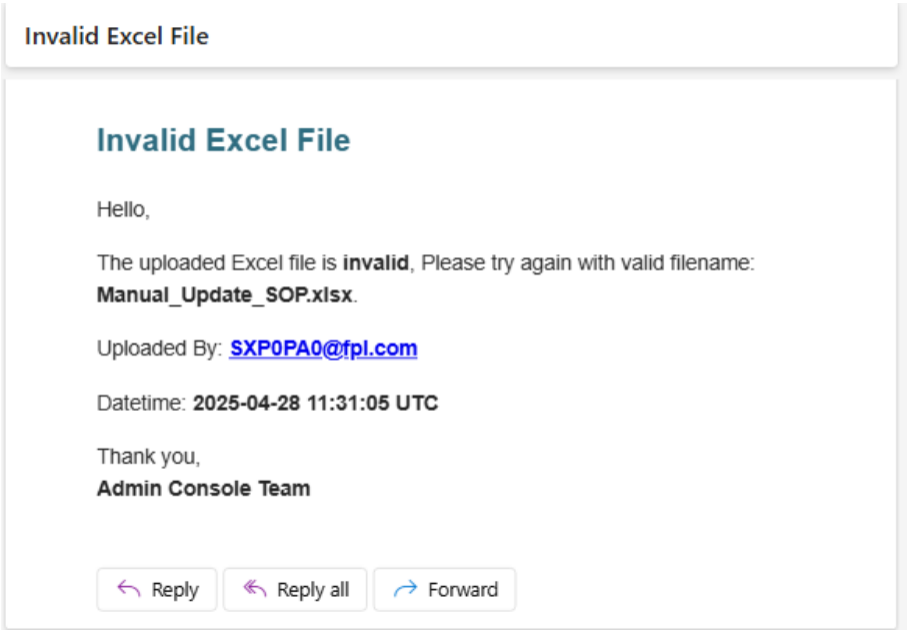
- Row 2: Value in Updated column 'test 2' is invalid. Value must be 'Yes' or 'No'.
- Row 3: Value in FLM Name column 'test' is invalid.
- Row 4: Value in PromptStatusActive column 'test 1' is invalid. Value must be 'true' or 'false'.
- Row 5: Value in FLM Name column 'test4' is invalid.

Thank you,
Admin Console Team

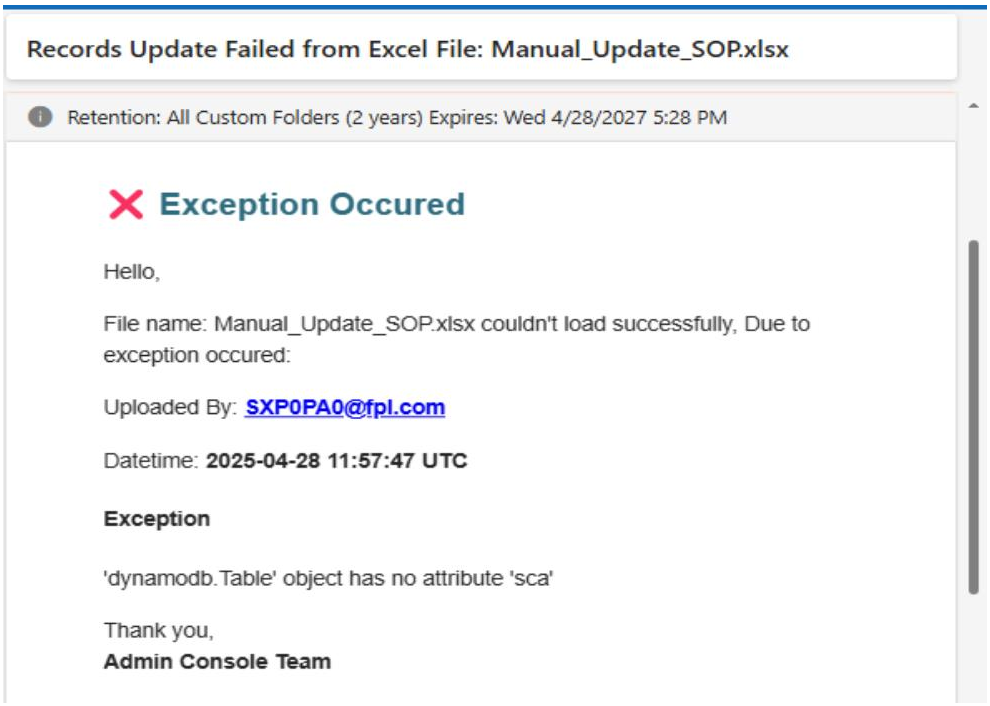
- An email is also triggered to group of users when there are **no records found to update**.

- An email is also triggered to group of users within a minute when a file uploaded with invalid name, or any kind of failure happens.

Email for Invalid File



Email for any exception that occurred.



- **Note:** Email will land in the External folder instead of the inbox folder.
- Need group of users' email id for each environment from the business team.

4.5. Audit Design

For Audit Trails in Admin Console, as if now we are storing the history of records in the same Table

- Current details of records with **IsActive = true**, which shows the recent data
- When updating a record in the table, update I**sActive = false** for current record, and then add new record with updated values and for that **IsActive = true** which is present in the same table.

Note: *As discussed, going forward we plan to migrate the historical records to some other resource, but we have not yet decided.*

4.6. Backend Component

Web Admin console backend comprises of following components:

Services	Resources Name	Description
SES	Sender Email: admin-console-test-mailbox.sharedmailbox@nexteraenergy.com	SES is used to trigger an email from Sender email to a DL email with a CSV file attachment.

	Reciever Email: DL-AdminConsole-ConfigUpdate@nexteraenergy.com	
DynamoDB	2XSFNV-EI864-Admin-Console-Application-Destination-List-Table	Table to manage configs for Application Destination List
	2XSFNV-EI864-Admin-Console-Asm-Prompts-Table	Table to manage configs for Area Specific Messages applicable for both HVCA and non-HVCA flows
	2XSFNV-EI864-Admin-Console-Key-Values-Table	Table to manage configs for Key-Values applicable for both HVCA and non-HVCA flows
	2XSFNV-EI864-Admin-Console-Gen-Prompts-Table	Table to manage configs for FLM records
Lambda	2XSFNV-EI864-Global-fetch-admin-console-key-values-LF	Lambda function to fetch key-value data on-demand from 2XSFNV-EI864-Admin-Console-Key-Values-Table table in the conversation flows
	2XSFNV-EI864-Global-Fetch-Admin-Console-ASM-LF	Lambda function to fetch area specific messages on-demand from 2XSFNV-EI864-Admin-Console-Asm-Prompts-Table table in the conversation flows
	2XSFNV-EI864-AdminConsole-DBUpdate	Lambda function to update the records present in different DynamoDB Table and send confirmation email to DL
	2XSFNV-EI864-AdminConsole-ActiveRecords-LF	Lambda function fetches the records from different Table and creates an excel then send on email
	2XSFNV-EI864-AdminConsole-DBInsert-LF	Lambda function to add one time data load into DynamoDB which is needed for release 1.
S3	2xsfnv-ei864-adminconsole-input-files	S3 bucket is used to store excel file and txt file which trigger the respective Lambda functions.
SFTP User Access	Raise request for SFTP access	Grants you access to a virtual SFTP endpoint that is internally mapped to an **Amazon S3 bucket** path
Putty	Raise request for installation of Putty	Used to generate the required **SSH-2 RSA private key** which is needed for authenticating via SFTP.

WinSCP	Raise request for installation of WinSCP	Secure file transfer client that uses the SSH key (from PuTTY) to authenticate to the SFTP endpoint and upload the file.

***Note**: Refer SFTP to S3 Upload file to get all ServiceNow request details, and step by step guide to upload file.*

4.6.1 Amazon Lambda - 2XSFNV-EI864-AdminConsole-DBUpdate

Functions

Functions are created to update records in DynamoDB Table via uploading excel file in S3 Bucket via SFTP.

Update Process Lambda Function Behavior

Steps

1. Fetch file from S3 using **boto3**.
2. Load Excel file using **openpyxl library**.
3. Processes only those data which has **Updated** column marked as **Yes** in every sheets.
4. Iterate over rows and update records in DynamoDB Tables.
5. Trigger SES to send email notification.
6. Log operations for audit.

Excel File Format that needs to upload on S3

- **Format**: .xlsx (Excel format)
- **Name**: *Manual_Update_SOP.xlxs (click to see Excel Template)*
- Multiple Sheets Allowed
- **Sheet Names**: *(Sheet name not allowed to change)*
    - Instructions *(Consist of all instructions related to other sheets)*
    - HVCA Toggle
    - FLM
    - ASM
    - Key Value
    - Application Destination

Environment Variables

List of environment variable used:

- APPLICATION_DESTINATION_TABLE

- ASM_TABLE
- GEN_TABLE
- KEY_VALUE_TABLE
- LOG_LEVEL_VALUE
- RECIPIENT_EMAIL
- S3_BUCKET
- S3_KEY
- SENDER_EMAIL

Exceptions Handled

Scenario	**Handling**
Missing sheet	Raise an error, log and exit.
Invalid sheet names	Raise an error, log and exit.
Permission denied (S3/DynamoDB access)	Catch and log "Access Denied to resource [S3/DynamoDB]".
Unexpected Exception	Generic fallback to catch unknown exceptions, log error and stack trace, return failure response.

4.6.2 Amazon Lambda - 2XSFNV-EI864-AdminConsole-ActiveRecords

Functions

This Lambda function is created to fetch the records in present in different DynamoDB Table via uploading txt file in S3 Bucket via SFTP.

Get Active Records Lambda Function Behavior

Steps

1. Once get_records.txt file is uploaded in root folder, Lambda gets triggered.
2. Lambda fetch records from different Tables.
3. Processes the list and creates an excel file with different sheets.
4. Trigger SES to send email notification to the group of users.
5. Log operations for audit.

Environment Variables

List of environment variable used:

- APPLICATION_DESTINATION_TABLE
- ASM_TABLE
- GEN_TABLE
- KEY_VALUE_TABLE
- LOG_LEVEL_VALUE
- RECIPIENT_EMAIL
- FILE_PATH
- SENDER_EMAIL

4.6.3 Amazon Dynamo DB

Area Specific Message Table: 2XSFNV-EI864-Admin-Console-Asm-Prompts-Table

Table Structure

Attribute Name	Description	Key Indicator	Data Type
Prompt Id	Unique Id given to the Prompt which is a combination of flow location name and Flow location Id	Partition Key	String
Id	Unique identifier of the item created by concatenation of Prompt Id and created timestamp	Sort Key	String
AreaCode	3 digit Area code for which the prompt to be played		String
BusinessResidentials	Indicator to show caller domain		String
Category	Category of area specific message, holds values like HVCA - ASM		String
CreatedDate	Prompt create date		String
Details	Description for the prompt		String
AreaName	Name of the Area Code		String
IsActive	Flag to indicate if the ASM is active or inactive, possible values True or False		String
Language	Prompt Language – English / Spanish		String
LastUpdated	Timestamp to indicate when the item was last updated		String
Prompts			Map
Prompts - English_Prompt	English prompt text		String

Prompts - Spanish_Prompt	Spanish prompt text		String
PromptStatusActive	Indicator to show if the prompt is currently being played or not		String
UpdatedBy	User Id of the person who has logged in to the web admin console interface making the changes		String

Field Level Validations

Following validations have been implemented for different fields of ASM

Attribute	Permissible Values
PromptId	For English – Concatenate with _ENG at end
	For Spanish – Concatenate with _SPA at end
AreaCode	239,305,321,386,407,448,561,645,728,754,772,786,850,863,904,941,954
BusinessResidentials	Business Residential All Callers
TypeOfASM	Upfront Safety Closing
Category	HVCA-ASM, NON-HVCA-ASM
IsActive	true/false
Language	English/Spanish
Value	On/Off

Key Value Management Table: 2XSFNV-EI864-Admin-Console-Key-Values-Table

Table Structure

Attribute Name	Description	Key Indicator	Data Type
KeyName	Name of Key	Partition Key	String
Id	Unique identifier created by concatenating key name and insert date timestamp	Sort Key	String
AdditionalApplicationDetails	Value Stream name where used		String
CreatedDate	Key create date		String
Description	Key Description		String
Districts	List of area code or district numbers applicable		String

IsActive	Flag to indicate if the Key Value is active or inactive, possible values True or False		Boolean
KeyValue	Value for the key		String
KeyValueCategory	Category of Key value pair		String
LastUpdated	Timestamp for item create or update		String
UpdatedBy	User Id of the person who has logged in to the web admin console interface making the changes		String

Field Level Validations

Following validations have been implemented for different categories of key-value

Category	Attribute	Permissible Values
InfoBlast Toggle	KeyName	
	Value	On/Off
Call Deflection Toggle	KeyName	
	Value	Off Text Only Email Only Both
StormControl Flag	KeyName	
	District	All Districts, 11, 12, 13, 21, 22, 23, 32, 34, 41, 42, 43, 44, 45, 46, 51, 52, 53, 54, 55, 56, 57, 71, 72, 73, 74, 81, 82, 83, 84, 85, 86
	Value	On/Off
HVCA Routing	Key Name	Alphanumeric values + special characters
	Value	On/Off

Application Destination List Management Table: 2XSFNV-EI864-Admin-Console-Application-Destination-List-Table

Table Structure

Attribute Name	Description	Key Indicator	Data Type
Application	Name of application	Partition Key	String

Id	Unique Id created for the item by concatenating Application & Update timestamp	Sort Key	String
CreatedDate	Prompt create date		String
Destination			String
IsActive	Flag to indicate if the item is active or inactive, possible values True or False		String
LastUpdated	Timestamp for item create or update		String
UpdatedBy	User Id of the person who is making the changes		String

Field Level Validations

Following validations have been implemented for different fields of Application Destination

Attribute	Permissible Values
Application	Outage, MoveOut, AccMaintenance, AccountNumberRetrieval, FixedRate, BudgetBill, eBill
IsActive	true/false
Destination	Available, Transfer, Disconnect

Prompts Management Table: 2XSFNV-EI864-Admin-Console-Gen-Prompts-Table

Table Structure

Attribute Name	Description	Key Indicator	Data Type
Prompt Id	Unique Id given to the Prompt	Partition Key	String
Id	Unique identifier of the item created by concatenation of Prompt Id and created timestamp	Sort Key	String
Block	Connect flow block name where its played		String
Category	Category of prompt , holds values like GENERAL, HVCA-FLM		String
CreatedDate	Prompt create date		String
AltPromptName	Flow location names		String
Domain	Value Stream for which the prompt is applicable , like Account Maintenance, Outage etc.		String
IsActive	Flag to indicate if the ASM is active or inactive, possible values True or False		Boolean

LastUpdated	Timestamp to indicate when the item was last updated		String
PromptStatusActive	To enable or disable the FLM prompts		String
Prompts			Map
Prompts - English_Prompt	Prompt to be played for English		String
Prompts - Spanish_Prompt	Prompt to be played for Spanish		String
Workflow	Name of Connect workflow where the prompt is played		String
UpdatedBy	User Id of the person who is making the changes		String
VariationOfPrompts			String
VariationOfPrompts - English Voice			String
VariationOfPrompts - Spanish Voice			String

Field Level Validations

Following validations have been implemented for different fields of ASM

Attribute	Permissible Values
PromptId	Combination of Domain, Workflow, Block
Domain	Outage, MoveOut, AccMaintenance, AccountNumberRetrieval, FixedRate, BudgetBill, eBill
AltPromptName	General no ETR, FRG Up Front, New Outage Report, Outage Ack, ASM For Employees
Category	HVCA-FLM, GENERAL
IsActive	true/false
PromptStatusActive	true/false

4.6.4 Amazon SES

Description:

In our implementation, AWS Simple Email Service (SES) is used to send emails to group of users (DL). The Lambda functions handle user context and triggers an email to the sender email address using send_message() function, which allows full control over headers, MIME content, and attachments.

API Used:

The function send_message() is used to enable rich formatting (HTML), custom headers, and optional attachments (e.g., CSV).

Identity Requirements

To use AWS SES, we must verify the email address or domain in Identities that we use in the Source field (i.e., the "From" address).

From Address: admin-console-test-mailbox.sharedmailbox@nexteraenergy.com

To Address: DL-AdminConsole-ConfigUpdate@nexteraenergy.com

Usage

AWS SES is used to trigger an email with an attachment as CSV file which consist of the active records of different Admin Console DynamoDB table to the DL.

4.7 User Authentication

Not the part for R1

4.8 One Time Data Hydration

Description - This Lambda function (Github Repository) is designed to load initial data which is needed for Toggle, Key-Values pair, FLM, ASM, and Application Destination Tables. The function is got triggered and load initial data if the data is not exist in the table.

Key Components

Imported Libraries:

- json: For handling JSON data.
- datetime: For date and time manipulation.
- boto3: For interaction with DynamoDB Tables.
- os: For accessing environment variables.
- logging: To logs the execution with LogLevel as DEBUG,INFO.

Functions:

- insert_app_destination(): Load data into Application Destination DynamoDB Table.
- insert_key_value(): Load data into Key Value DynamoDB Table
- insert_flm(): Load data into Gen Prompts DynamoDB Table
- insert_asm(): Load data into ASM DynamoDB Table

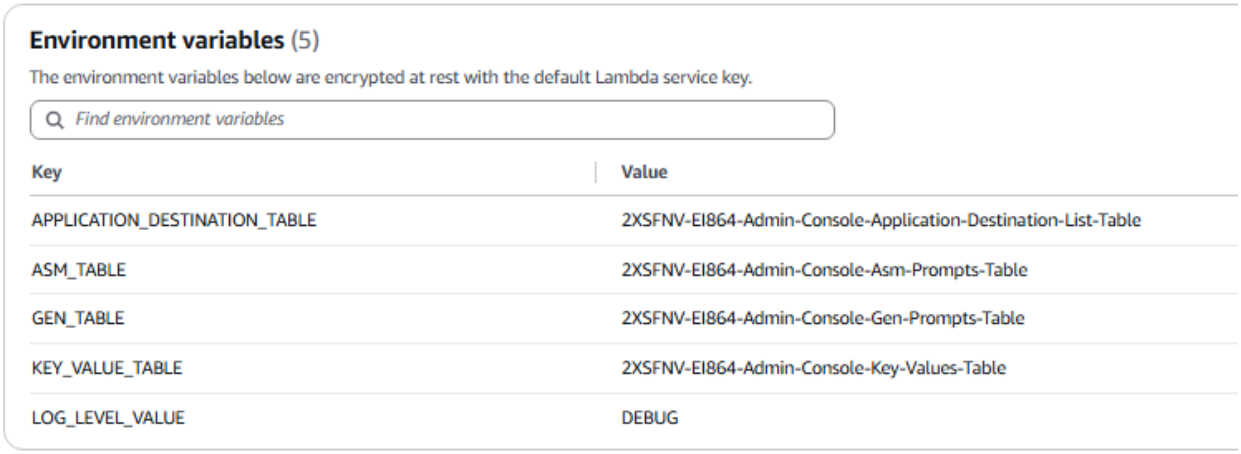
Lambda Handler:

The lambda_handler function is the entry point for the AWS Lambda function. It:

- Get automatically triggered when deployment is complete.
- Call each function one by one to load initial data in respective DynamoDB.
- Combines the results of the executed functions into a single response object.

Environment Variables

The function expects certain environment variables to be set, such as APPLICATION_DESTINATION_TABLE, ASM_TABLE, GEN_TABLE, KEY_VALUE_TABLE, and LOG_LEVEL_VALUE, which are used in the lambda_handler() function



Error Handling

The function includes basic error handling within each API function and the main handler to return a 500 status code and an error message if an exception occurs.

4.7. Deployment Strategy

4.7.1. Lambda resources

Admin Console lambda functions present on Different GitHub repository

Lambda Name	Description	Github URL
2XSFNV-EI864-AdminConsole-DBUpdate	To update the record via upload, excel file on S3 bucket via SFTP.	URL
2XSFNV-EI864-AdminConsole-ActiveRecords	To fetch all the active records via uploading txt file on S3 bucket.	URL

2XSFNV-EI864-AdminConsole-DBInsert	It is used for one time data load for different Initial values in DynamoDB.	URL
2XSFNV-EI864-Global-Fetch-Admin-Console-ASM	Used to fetch the Area Specific Message based on Area Code and Language	URL
2XSFNV-EI864- Global-Fetch-Admin-Console-Key-Values	Used to fetch Key Value pairs based on the input AdditionalApplication details which is Domain and Source as “Connect”.	URL

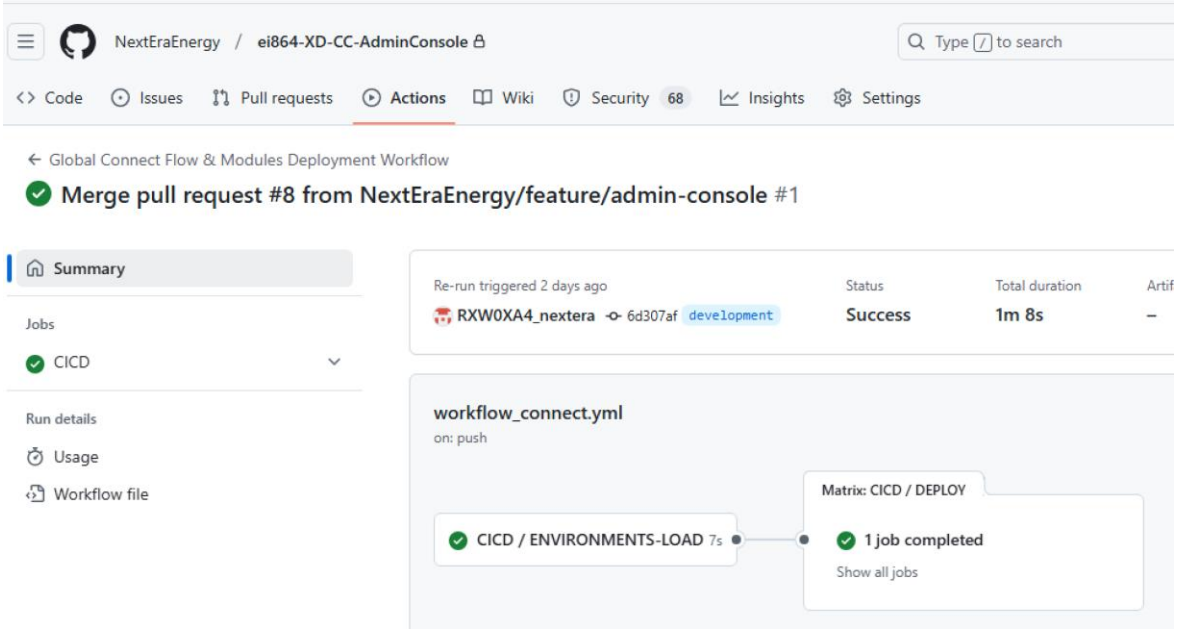
In below sections, Deployment process mentioned for one of the resources which is applicable for the other resources.

4.7.2. Connect Flow Deployment

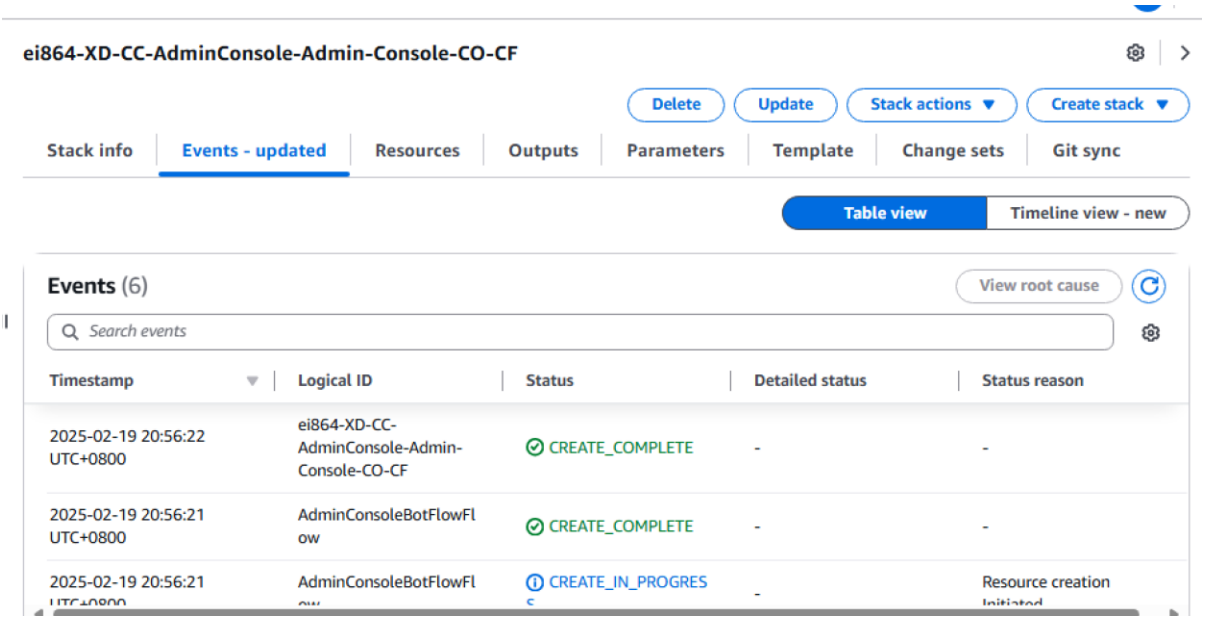
Raise GitHub pull request for the AWS Connect Flow json files changes

- Dev: https://github.com/NextEraEnergy/ei864-XD-CC-AdminConsole/tree/development
- TEST: https://github.com/NextEraEnergy/ei864-XD-CC-AdminConsole/tree/integration
- QA: https://github.com/NextEraEnergy/ei864-XD-CC-AdminConsole/tree/release
- PROD: https://github.com/NextEraEnergy/ei864-XD-CC-AdminConsole/tree/master

Automatically trigger the AWS Connect Deployment CICD Pipeline job after the Pull Request is merged.



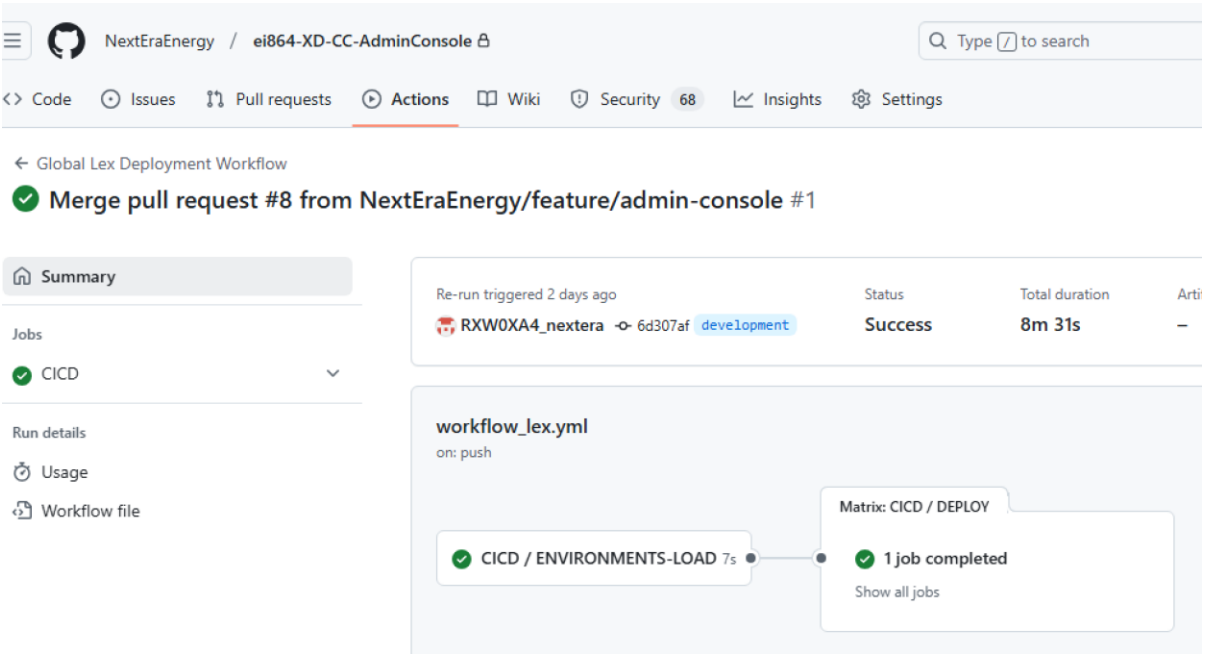
Deploy AWS Connect Flows through CloudFormation template which is executed from the GitHub CICD pipeline



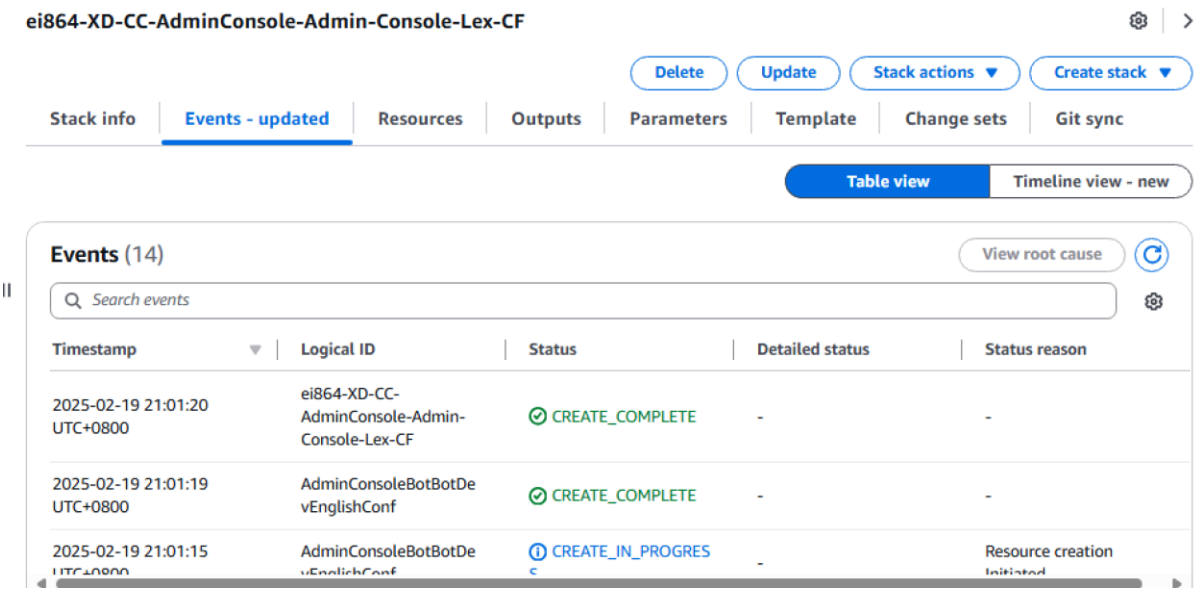
4.7.3. Lex Bot Deployment

Raise GitHub pull request for the AWS Lex Bot files changes

Automatically trigger the AWS Lex Deployment CICD Pipeline job after the Pull Request is merged.



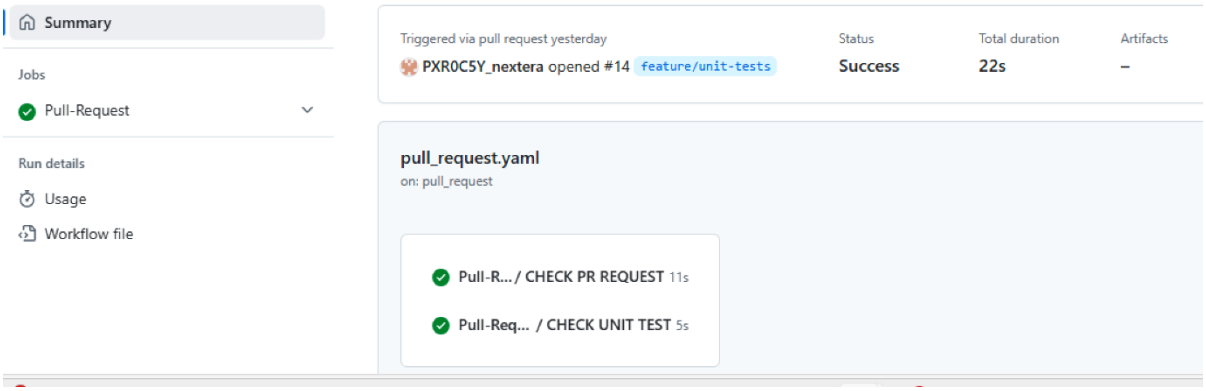
Deploy AWS Lex bots through CloudFormation template which is executed from the GitHub CICD pipeline



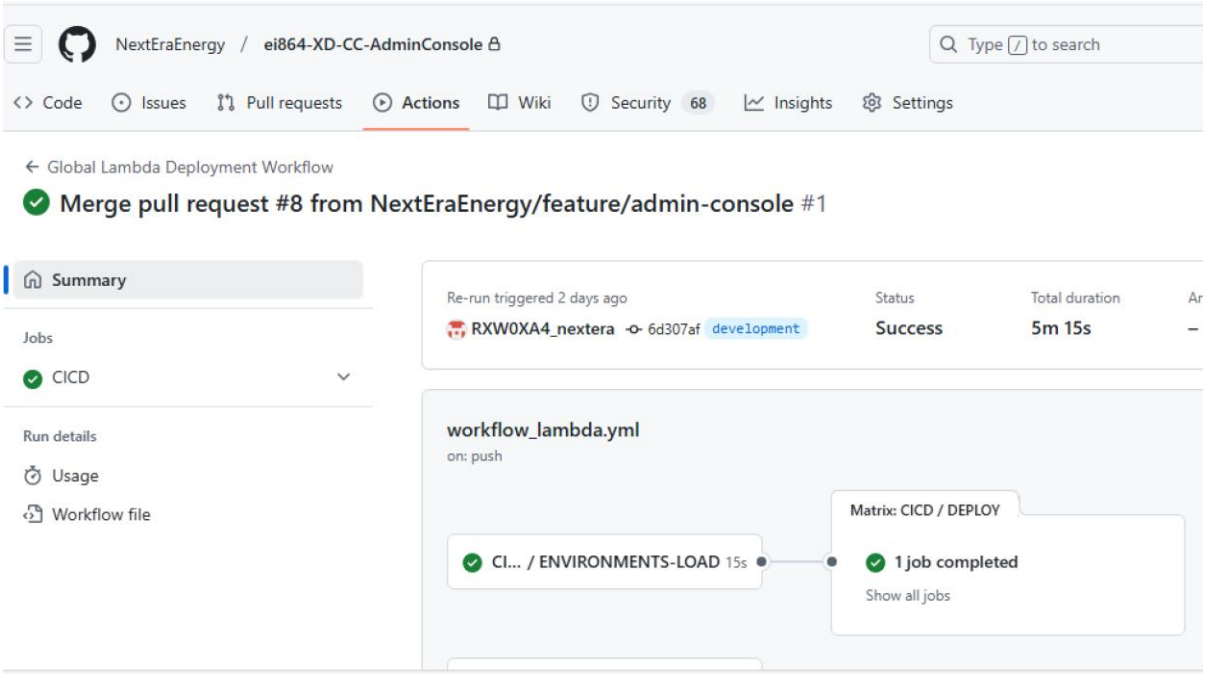
4.7.4. Admin Console Lambda Deployment

Raise GitHub pull request for the AWS Lambda files changes

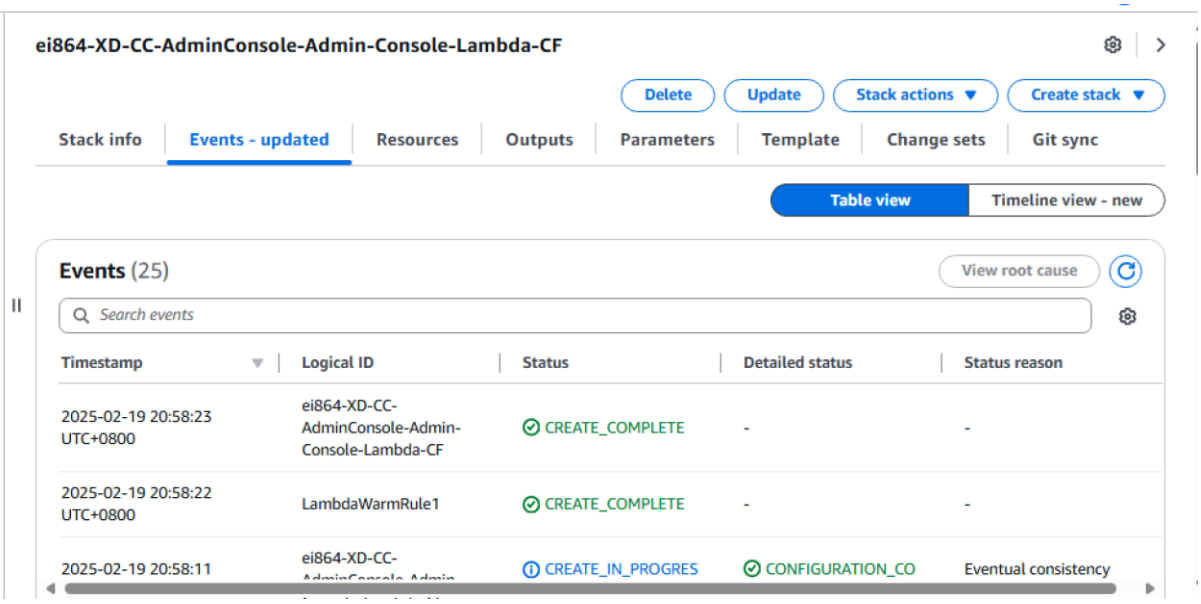
Automatically trigger the GitHub pull request check while the PR is opened, make sure the SonarQube and Unit Test jobs are passed.



Automatically trigger the AWS Lambda Deployment CICD Pipeline job after the Pull Request is merged.



Deploy AWS Lambda through CloudFormation template which is executed from the GitHub CICD pipeline



4.7.5. Appendix - Reference

Document Type	Document Details

Functional Requirements (JIRA Ids)	XDCCAD – 3866 , XDCCAD – 3867,XDCCAD – 3868,XDCCAD – 3869, XDCCAD – 3870
Voiceflows	Url for the voiceflows : https://creator.voiceflow.com/project/6750d5846d69bcb82c170656/canvas/64dbb6696a8fab0013dba194
Technical Components Mapping	Admin Console Technical Component Mapping.xlsx
Functional Unit Testing Document	
Technical Unit Testing Document	

5. Alerts, Notification & Monitoring Dashboards

5.1. Amazon CloudWatch

This section outlines the Amazon CloudWatch configuration for Amazon Connect, AWS Lambda, and Amazon Lex services. It includes monitored metrics, dashboards, alarms, and log groups in Dev, Test, and Performance environments.

5.1.1 Alarm Configuration Review

Metric Type	Metric Source	Description	Current Logic Used (Dev)	Production Logic	Enabled

Connect	Contact Flows Fatal Errors	The number of times a flow failed to execute due to a system error.	Sum >= 1in 1 min	Sum >= 1in 1 min	2XSFNV-EI864-CFN-Outage-Initial-Flow 2XSFNV-EI864-CFN-Outage-HVCAAuthentication-IntentCheck-Flow 2XSFNV-EI864-CFN-Outage-HVCAAuthentication-UserDetails-Flow 2XSFNV-EI864-CFN-Outage-HVCAAuthentication-Flow 2XSFNV-EI864-CFN-Outage-ETRTime-Flow 2XSFNV-EI864-CFN-Outage-ReportTicket-Flow 2XSFNV-EI864-CFN-Outage-StormStatusLine-Flow
	ConcurrentCalls	total number of concurrent voice calls (inbound and outbound) at a given time across your Connect instance.	Maximum > =80% of quota in 1 min	Maximum > =80% quota in 1 min	
	ConcurrentCalls	total number of concurrent voice calls (inbound and outbound) at a given time across your Connect instance.	Maximum > =100% of quota in 1 min	Maximum > =80% quota in 1 min	
	ThrottledCalls	The ThrottledCalls metric in Amazon Connect tracks the number of API calls that were throttled due to rate limiting by AWS	Sum >= 1 in 1 min , Require to revissit after HyperCare	Sum >= 1 in 1 min , Require to revissit after HyperCare	

Lambda	Errors	Counts Lambda execution errors. Key for reliability monitoring.	Sum >= 1 in 1 min , Require to revissit after HyperCare	Sum >= 1 in 1 min , Require to revissit after HyperCare
	Throttles	Indicates concurrency limit reached, leading to throttled executions.	Sum >= 1in 1 min	Sum >= 1 in 1 min
Lex	RuntimeConcurrentRequests	Maximum number of requests in 1min period	Maximun >= 80% quota	Maximum > =80% quota in 1 min
	RuntimeThrottledEvents	The ThrottledCalls metric in Amazon Lex tracks the number of API calls that were throttled due to rate limiting by AWS	Sum >= 1in 1 min	Sum >= 1 in 1 min
	RuntimeSystemErrors	The number of system errors in the specified period.	Sum >= 1in 1 min	Sum >= 1 in 1 min

Amazon Lex Metrics:

Amazon Lex metrics are currently out of scope due to cost and metric volume considerations.

5.1.2 Dashboards

CloudWatch Dashboards created for real-time monitoring of key metrics for Amazon Connect, AWS Lambda, and Amazon Lex.

Connect:

https://docs.aws.amazon.com/connect/latest/adminguide/monitoring-cloudwatch.html

Sno	Dashboard Name	TYPE	Period in Min	Statistic	Description

1	ConcurrentCallsPercentage	Connect	1	Average	The percentage of the concurrent active voice calls service quota used in the instance. This is calculated by: ConcurrentCalls / ConfiguredConcurrentCallsLimit Unit: **Percent** (output displays as a decimal) Dimensions: InstanceId: The ID of your instance MetricGroup: VoiceCall
2	Concurrent Calls	Connect	1	Maximum/Average	The number of concurrent active voice calls in the instance at the time the data is displayed in the dashboard. The value displayed for this metric is the number of concurrent active calls at the time the dashboard is displayed, and not a sum for the entire interval of the refresh interval set. All active voice calls are included, not only active calls that are connected to agents. While all statistics are available in CloudWatch for concurrent voice calls you might be most interested in looking at the **Maximum/Average statistic**. The Sum statistic isn't as useful here. Unit: **Count** Dimensions: InstanceId: The ID of your instance MetricGroup: VoiceCalls
3	Throttled Calls	Connect	1	Sum	The number of voice calls that were rejected because the rate of calls per second exceeded the maximum supported quota. To increase the supported rate of calls, request an increase in the service quota for concurrent active calls per instance. To monitor the total throttled calls in a given time period, take a look at the **Sum statistic** in CloudWatch. Unit: **Seconds** Unit**: Count**

4	Contact Flow Errors	Connect	1	Sum/ Sample Count	The number of times the error branch for a flow was run. Unit: **Count** Dimensions: InstanceId: The ID of your instance MetricGroup: ContactFlow ContactFlowName: The name of your flow
5	Contact Flows Fatal Errors	Connect	1	Sum/ Sample Count	The number of times a flow failed to execute due to a system error. Unit: Count Dimensions: InstanceId: The ID of your instance MetricGroup: ContactFlow ContactFlowName: The name of your flow
6	ToInstancePacketLossRate	Connect	1	Average	The ratio of packet loss for calls in the instance, reported every 10 seconds. Each data point is between 0 and 100. The ratio of packet loss for calls in the instance appears as a percent between 0 and 1. Unit: **Percent**
7	Calls Per Interval	Connect	1	Sum/ Sample Count	The number of voice calls, both inbound and outbound, received or placed per second in the instance. Unit: **Count**
8	CallsBreachingConcurrencyQuota	Connect	1	Sum	The total number of voice calls that exceeded the concurrent calls quota for the instance. For the total number of calls that breach the quota, take a look at the **Sum** statistic. Unit: **Count**

Lex:

Sno	Dashboard Name	TYPE	Period in Min	Statistic	Description

1	RuntimeRequestCount	Lex	1	Sample Count	The number of concurrent connections in the specified time period. RuntimeConcurrency is reported as a StatisticSet. operations: StartConversation Unit: **Count**
2	RuntimeSuccessfulRequestLatency	Lex	1	Average	The latency for successful requests between the time the request was made and the response was passed back. **operation-**StartConversation **Unit: milliseconds**
3	RuntimeConcurrency	Lex	1	Sample Count	The number of concurrent connections in the specified time period. RuntimeConcurrency is reported as a StatisticSet. Valid dimensions for the RecognizeUtterance or **StartConversation** operations: Operation, BotId, BotAliasId, InputMode, LocaleId Unit: **Count**
4	RuntimeRequestLength	Lex	1	Average	Total length of a conversation with a Amazon Lex V2 bot. Only applicable to the StartConversation operation. operations: StartConversation Unit: **milliseconds**
5	RuntimeSystemErrors	Lex	1	Sample Count	The number of system errors in the specified period. The response code range for a system error is 500 to 599. operations: StartConversation Unit: **Count**

6	RuntimeThrottledEvents	Lex	1	Sample Count	The number of throttled events. Amazon Lex V2 throttles an event when it receives more requests than the limit of transactions per second set for your account. If the limit set for your account is frequently exceeded, you can request a limit increase. To request an increase, see AWS service limits. Valid dimensions for the RecognizeUtterance and StartConversation operations: operations: StartConversation Unit: **Count**
7	RuntimeUserErrors	Lex	1	Sample Count	The number of user errors in the specified period. The response code range for a user error is 400 to 499. operations: StartConversation Unit: **Count**

Lambda:

Sno	Dashboard Name	TYPE	Period in Min	Statistic	Description
1	ClaimedAccountConcurrency	Lambda	1	Maximum	For a Region, the **amount of concurrency** that is unavailable for on-demand invocations. ClaimedAccountConcurrency is equal to UnreservedConcurrentExecutions plus the **amount of allocated concurrency** (i.e. the total reserved concurrency plus total provisioned concurrency).
2	ConcurrentExecutions	Lambda	1	Maximum	The number of function instances that are processing events. If this number reaches your concurrent executions quota for the Region, or the reserved concurrency limit on the function, then Lambda throttles additional invocation requests. Lambda reports concurrency metrics as an aggregate count of the number of instances processing events

					across a function, version, alias, or AWS Region. To see how close you are to hitting concurrency limits, view these metrics with the **Max statistic.**
3	Duration	Lambda	1	Average	The amount of time that your function code spends processing an event. The billed duration for an invocation is the value of Duration rounded up to the nearest millisecond. Duration does not include cold start time. Performance metrics provide performance details about a single function invocation. For example, the Duration metric indicates the amount of time in milliseconds that your function spends processing an event. To get a sense of how fast your function processes events, view these metrics with the **Average** or **Max** statistic.
4	Errors	Lambda	1	Sum	The number of invocations that result in a function error. Function errors include exceptions that your code throws and exceptions that the Lambda runtime throws. The runtime returns errors for issues such as timeouts and configuration errors. To calculate the error rate, divide the value of Errors by the value of Invocations. Note that the timestamp on an error metric reflects when the function was invoked, not when the error occurred.
5	Invocations	Lambda	1	Sum	The number of times that your function code is invoked, including successful invocations and invocations that result in a function error. Invocations aren't recorded if the invocation request is throttled or otherwise results in an invocation error. The value of Invocations equals the number of requests billed.

6	Throttles	Lambda	1	Sum	The number of invocation requests that are throttled. When all function instances are processing requests and no concurrency is available to scale up, Lambda rejects additional requests with a TooManyRequestsException error. Throttled requests and other invocation errors don't count as either Invocations or Errors.

5.2. Metric Validation Review

5.2.1. Amazon Connect Alarms

Metric	Reviewed (Y/N)	Comments
⚠ Note	—	Alarms are configured correctly, but some of the alarms cannot be validated due to insufficient data in Dev/Test environments.
ContactFlowFatalErrors	N	Insufficient traffic/data in Dev/Test
Concurrent Calls	N	Low concurrent usage; data not representative
Throttled Calls	Y	Data observed; metric within threshold

5.2.2. Lambda Alarms:

Metric	Reviewed (Y/N)	Comments
⚠ Note	—	1.Some metrics, such as Throttles and Recursive Invocations Dropped, have insufficient data due to low load in Dev/Test environments. 2. For monitoring purposes, both **Error** and **Throttle** metrics will be tracked for each of the 13 AWS Lambda functions. This results in a total of **26 CloudWatch alarms** being created—**2 alarms per function** (one for errors and one for throttles).

Errors	Y	Alarm triggered as expected during test scenarios
Throttles	N	No throttling observed; insufficient load to validate

5.2.3. Lex Alarms:

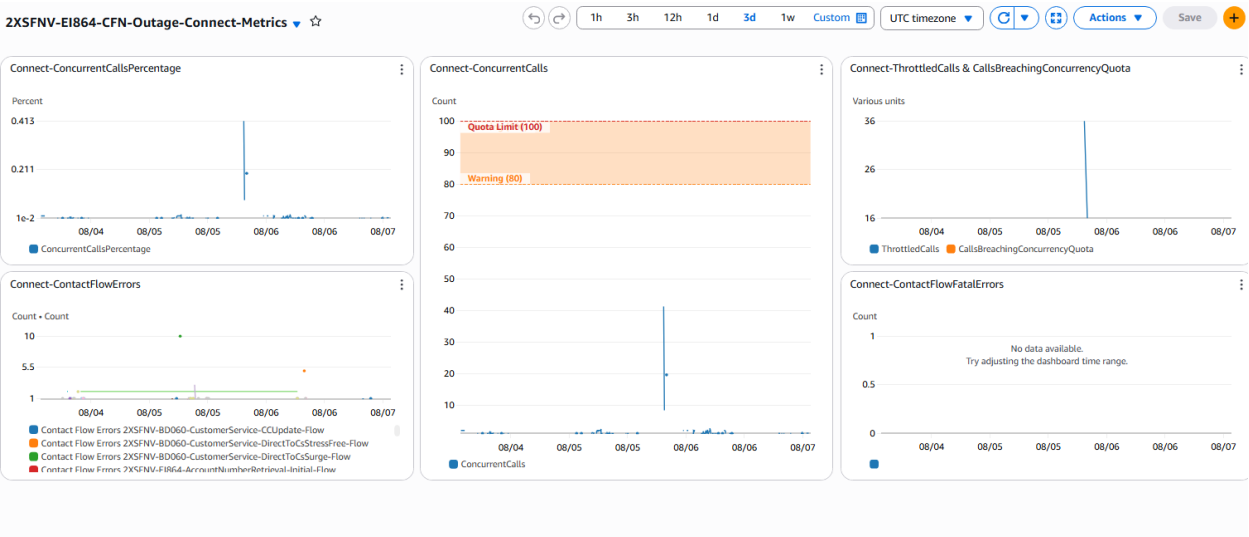
Metric	Reviewed (Y/N)	Comments
RuntimeConcurrentRequests	N	No More Runtime Concurrent Requestsobserved; insufficient load to validate
RuntimeThrottledEvents	N	No Runtime throttling observed; insufficient load to validate
RuntimeSystemErrors	N	No Runtime systemerrors observed; insufficient load to validate

5.3. Dashboard validation:

5.3.1. Amazon connect:

Dashboard Name	Reviewed (Y/N)	Comments
⚠️ Note	—	Dashboards are configured correctly, but some cannot be fully validated due to insufficient data in Dev/Test environments.
Concurrent Calls (%)	Y	Metrics displayed as expected during limited testing.
Concurrent Calls (number)	Y	Dashboard functional; data volume limited in lower environments.
Throttled Calls (number)	Y	Metric displayed correctly, though throttling not triggered in current load.
Contact Flow Errors	Y	Error metrics appeared during testing; behavior as expected.
Contact Flows Fatal Errors	N	No fatal errors triggered in Dev/Test; unable to validate dashboard data.
Calls Per Interval	Y	Call volume visible; dashboard updated as per metric expectations.

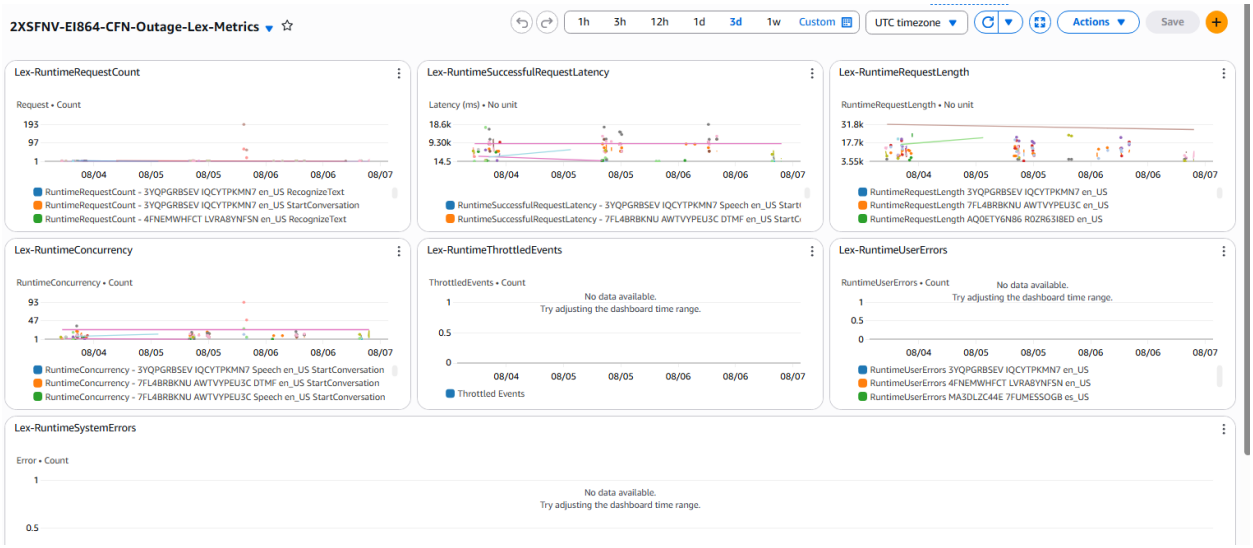
Calls Breaching Quota	Y	Metric and alarm showed up under test load; dashboard appears functional.



5.3.2. Amazon Lex:

Dashboard Name	Reviewed (Y/N)	Comments
⚠️ Note	—	Dashboards are configured correctly, but cannot be validated for some metrics due to insufficient data in Dev/Test environments.
RuntimeConcurrency	Y	Dashboard is active; test data verified concurrency metric visibility.
RuntimeUserErrors	Y	User error metrics confirmed during functional testing.

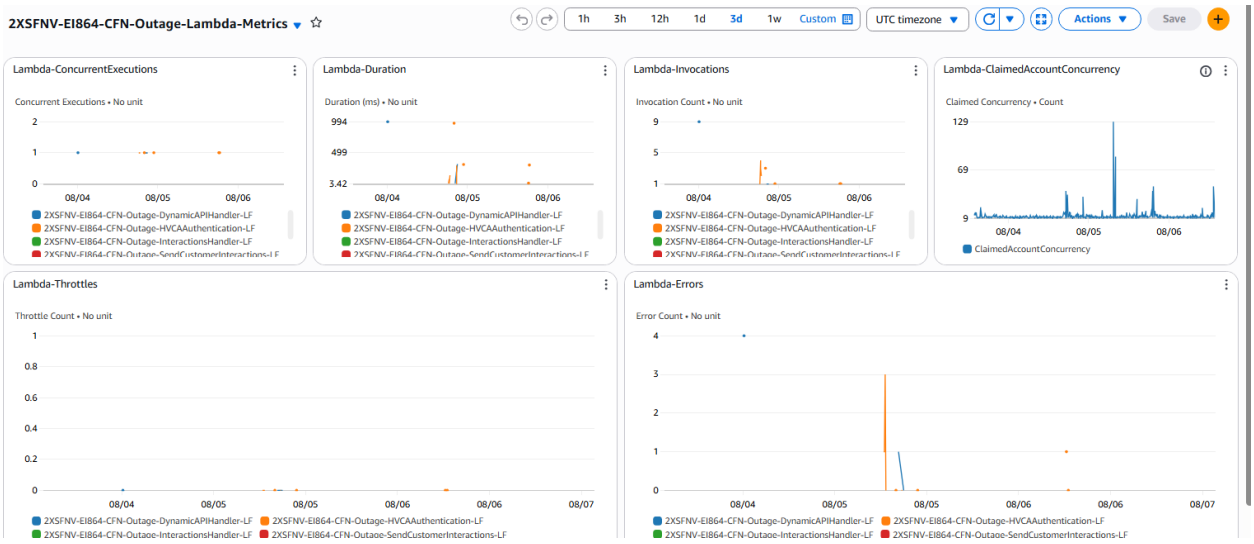
RuntimeRequestCound/RuntimeSuccessfulRequestLatency	Y	Dashboard metrics aligned with test data; validated successfully.
RuntimeRequestLength	Y	Metric values observed and consistent with expected payload sizes.
RuntimeSystemErrors	Y	Dashboard rendered correctly; system errors triggered during test runs.
RuntimeThrottledEvents	N	Throttling conditions not met in Dev/Test; dashboard cannot be fully validated.



5.3.3. AWS lambda:

Dashboard Name	Reviewed (Y/N)	Comments

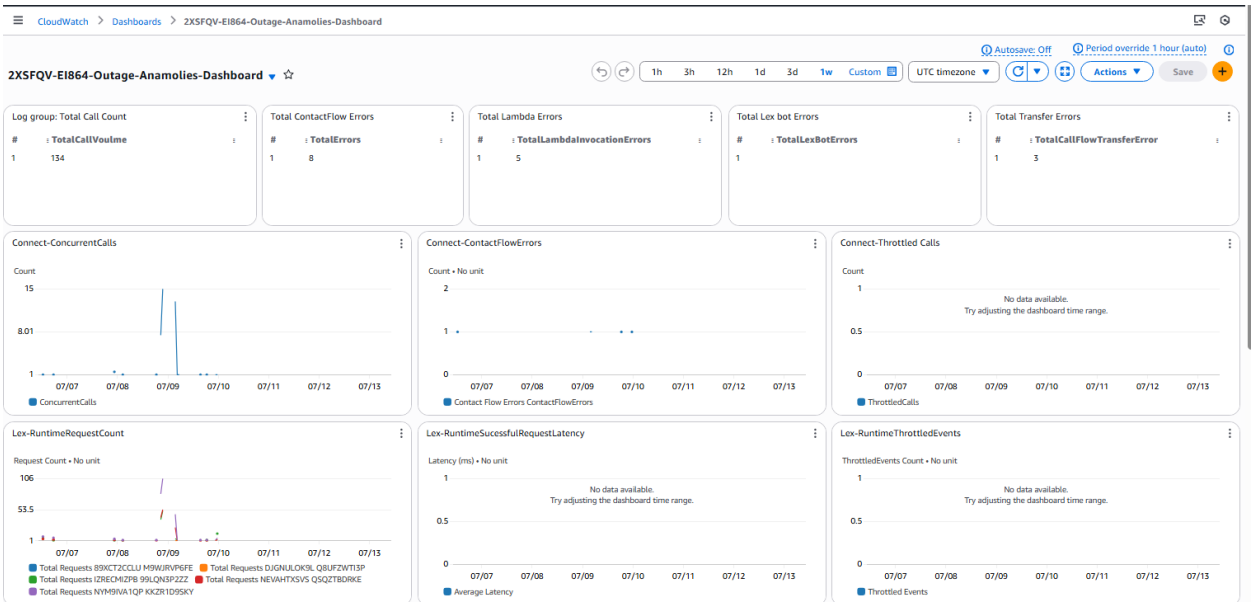
⚠️ Note	—	Dashboards are configured correctly, but cannot be validated for some metrics due to insufficient data in Dev/Test environments.
ClaimedAccountConcurrency	Y	Dashboard reviewed; metric values aligned with expected patterns.
ConcurrentExecutions	Y	Metric tracked active executions; data verified during load testing.
Duration	Y	Duration values observed; dashboard validated against test events.
Errors	Y	Errors occurred during testing; dashboard working as intended.
Invocations	Y	Invocations metric validated with Dev/Test Lambda execution.
Throttles	Y	Throttling observed under test load; dashboard validated.

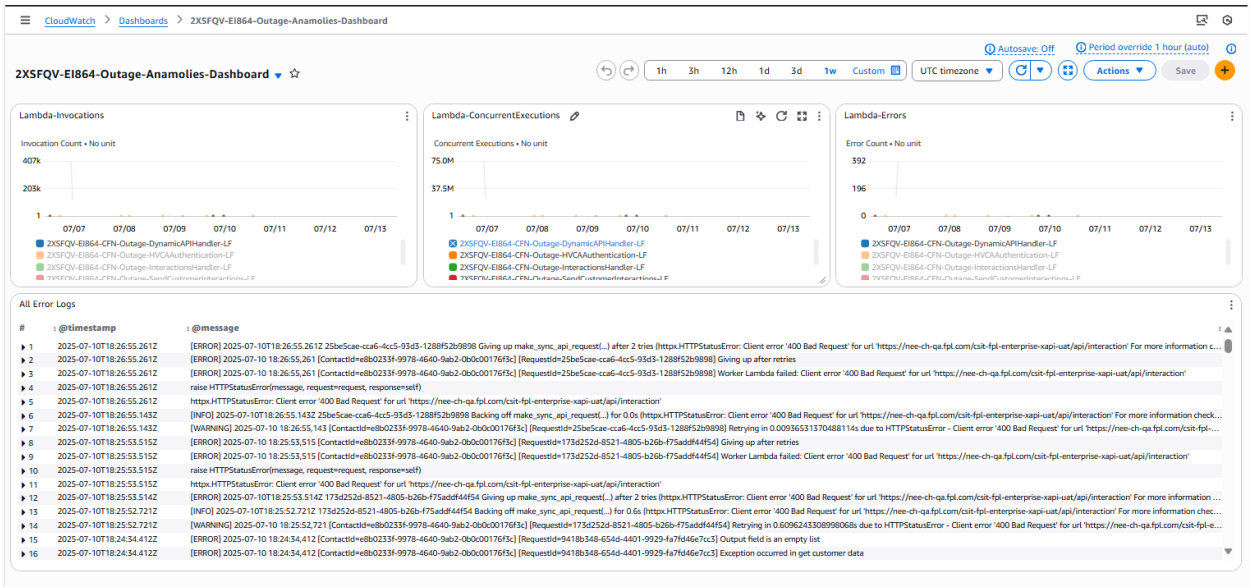


6. Outage-Anamolies-Dashboard:

6.2. Dashboard widgets:

- Total Call Count
- Total ContactFlow Errors
- Total Lambda errors
- Total Lex Bot errors
- Total Transfer Errors
- Connect-ConcurrentCalls
- Connect-CotactFlowerrors
- Connect-ThrottledCalls
- Lex-RuntimeRequestCount
- Lex-RuntimeSucessfulRequestLatency
- Lex-RuntimeThrottledEvents
- Lambda-Invocations
- Lambda-ConcurrentExecutions
- Lambda-Errors
- All Error Logs




CloudWatch
Dashboards
2XSFQV-EI864-Outage-Anamolies-Dashboard
2XSFQV-EI864-Outage-Anamolies-Dashboard
Autosave: Off
Period override 1 hour (auto)
1h
3h
12h
1d
3d
1w
Custom
UTC timezone
Actions
Save
Lambda-Invocations
Invocation Count • No unit
407k
203k
1
07/07
07/08
07/09
07/10
07/11
07/12
07/13
2XSFQV-EI864-CFN-Outage-DynamicAPIHandler-LF
2XSFQV-EI864-CFN-Outage-HVCAAuthentication-LF
2XSFQV-EI864-CFN-Outage-InteractionsHandler-LF
Lambda-ConcurrentExecutions
Concurrent Executions • No unit
75.0M
37.5M
1
2XSFQV-EI864-CFN-Outage-DynamicAPIHandler-LF
2XSFQV-EI864-CFN-Outage-HVCAAuthentication-LF
2XSFQV-EI864-CFN-Outage-InteractionsHandler-LF
Lambda-Errors
Error Count • No unit
392
196
0
2XSFQV-EI864-CFN-Outage-DynamicAPIHandler-LF
2XSFQV-EI864-CFN-Outage-HVCAAuthentication-LF
2XSFQV-EI864-CFN-Outage-InteractionsHandler-LF
All Error Logs
@timestamp @message
1 2025-07-10T18:26:55.261Z [ERROR] 2025-07-10T18:26:55.261Z 25be5cae-cca6-4cc5-93d3-1288f52b9898 Giving up make_sync_api_request(...) after 2 tries (httpx.HTTPStatusError: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction' For more information c...
2 2025-07-10T18:26:55.261Z [ERROR] 2025-07-10 18:26:55,261 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=25be5cae-cca6-4cc5-93d3-1288f52b9898] Giving up after retries
3 2025-07-10T18:26:55.261Z [ERROR] 2025-07-10 18:26:55,261 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=25be5cae-cca6-4cc5-93d3-1288f52b9898] Worker Lambda failed: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction'
4 2025-07-10T18:26:55.261Z raise HTTPStatusError(message, request=request, response=self)
5 2025-07-10T18:26:55.261Z httpx.HTTPStatusError: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction'
6 2025-07-10T18:26:55.143Z [INFO] 2025-07-10T18:26:55.143Z 25be5cae-cca6-4cc5-93d3-1288f52b9898 Backing off make_sync_api_request(...) for 0.0s (httpx.HTTPStatusError: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction' For more information check...
7 2025-07-10T18:26:55.143Z [WARNING] 2025-07-10 18:26:55,143 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=25be5cae-cca6-4cc5-93d3-1288f52b9898] Retrying in 0.009365313704881 14s due to HTTPStatusError - Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-...
8 2025-07-10T18:25:53.515Z [ERROR] 2025-07-10 18:25:53,515 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=173d252d-8521-4805-b26b-f75addf44f54] Giving up after retries
9 2025-07-10T18:25:53.515Z [ERROR] 2025-07-10 18:25:53,515 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=173d252d-8521-4805-b26b-f75addf44f54] Worker Lambda failed: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction'
10 2025-07-10T18:25:53.515Z raise HTTPStatusError(message, request=request, response=self)
11 2025-07-10T18:25:53.515Z httpx.HTTPStatusError: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction'
12 2025-07-10T18:25:53.514Z [ERROR] 2025-07-10T18:25:53.514Z 173d252d-8521-4805-b26b-f75addf44f54 Giving up make_sync_api_request(...) after 2 tries (httpx.HTTPStatusError: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction' For more information ...
13 2025-07-10T18:25:52.721Z [INFO] 2025-07-10T18:25:52.721Z 173d252d-8521-4805-b26b-f75addf44f54 Backing off make_sync_api_request(...) for 0.6s (httpx.HTTPStatusError: Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-enterprise-xapi-uat/api/interaction' For more information chec...
14 2025-07-10T18:25:52.721Z [WARNING] 2025-07-10 18:25:52,721 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=173d252d-8521-4805-b26b-f75addf44f54] Retrying in 0.6096243308998068s due to HTTPStatusError - Client error '400 Bad Request' for url 'https://nee-ch-qa.fpl.com/csit-fpl-e...
15 2025-07-10T18:24:34.412Z [ERROR] 2025-07-10 18:24:34,412 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=9418b348-654d-4401-9929-fa7fd46e7cc3] Output field is an empty list
16 2025-07-10T18:24:34.412Z [ERROR] 2025-07-10 18:24:34,412 [ContactId=e8b0233f-9978-4640-9ab2-0b0c00176f3c] [RequestId=9418b348-654d-4401-9929-fa7fd46e7cc3] Exception occurred in get customer data