AMEX Engineering Statement of Direction-
Txn_AI_Agent

Document status	Submitted [V1.0]
Document owners	Nitin Lodhe Rituraj Dashore Veerabhadrayya Math Ambar Prakash Neeraj Parihar Varuni Ramkrishna
Product Owners	Prerna Tagade[Product Lead] Bhakti Thakkar Karthikeya Kssk
QA	Veerbhadrayya Math
Agile Champion	Yadu Mani Kar
Reviewers	Sibish Basheer, Debashish Das, Thirumaran Chinniah Ambalam
Approvers	Matthew Peters

TABLE OF CONTENTS

1. ABOUT THIS DOCUMENT

1.1. Summary

The ESOD is intended to detail the technical requirements and dependencies for delivering the proposed next gen conversational intelligence solution. This document focuses on building of an AI agent for the transaction and dispute journeys in conversational intelligence askamex automation platform which powers self-servicing capabilities for the card member through chat servicing channels. The scope of development for MVP (minimum viable product) is to build conversational experience related to transaction and dispute journeys. The MVP will also leverage Large Language models (Azure Open AI) for conversational understanding and provide appropriate self service capabilities for card members.

1.2. Artifacts

- PRD
- Architecture Artifacts
- Journey Flows

Sr.	Business Journey	Link	MVP	Journey tool mapping
1	Dispute entry	Disputes Bot Entry (24-01-16)	Y	dispute_transaction, report_transacion_fraud, alert_pending_transaction. dispute_status
2	Dispute status	Dispute Status (23-09-19)	Y	dispute_status -> give_transaction_details
3	Dispute cancel	Cancel Dispute (24-01-16)	Y (No automated cancellation)	dispute_transaction -> give_transaction_details
4	Dispute handoff	Disputes Handoff - Web (23-05-09)	Y	dispute_transaction, report_transacion_fraud, alert_pending_transaction, dispute_status
5	Txn Entry	TXN (23-09-19)	Y	give_transaction_details, view_digital_receipt
6	Txn Mobile handoff	TXN (23-09-19)	Y	give_transaction_details, view_digital_receipt
7	Txn Digital receipt	Digital Receipts	Y	view_digital_receipt
8	Txn Alert me when posted	Alert Me When Posted	Y	alert_pending_transaction
9	Txn Pending fraud	Pending Charges - Fraud transfer	Y	alert_pending_transaction

10	Txn Pending dispute	Pending Charges - Dispute Prompt	Y	alert_pending_transaction

1.3. Change Log

Date	Version	Changes
2024-03-18	0.1	Initial Draft
2024-03-20	1.0	For Review & Pre-Read for Product Counsil

2. EXECUTIVE SUMMARY

The objective of this document is to layout various capabilities and components which will be developed as part of the transaction AI agents which uses the latest tools sets and Large Language models for enabling the self-servicing capabilities through chat servicing channels. This document emphasizes the development of Transaction and Dispute journeys on the new transaction AI Agent on the AskAmex platform, aligning with the existing architecture. We are planning to cover below capabilities as part of the MVP.

- Current Transaction-Dispute functionalities in Internal framework with Chat-GPT LLM
  - Ability to view existing disputes and their statuses.
  - Ability to raise new dispute.
  - Ability to view the transactions.
  - Ability to set email alert for pending transactions.
  - Ability to report fraud.
  - Ability to view resolution letters and digital receipts.
  - Hand off to bots on CDMS platform
- Ability to hand off to Fraud bot, Payment Late bot, Card Declined bot.
- Mobile handover for extended transaction details
- Ability to connect CM from mobile app to chat with transaction context.

The essential integration which are required for this use case are:

- FINS – Integration with Global Financials (a.k.a FINS). Restful APIs help primarily to deal with the financial details of a particular card account such as Transaction details.
- CBIS – Onboard with CBIS (Claim Based Identity Service) to generate application client secret to establish secure A2A connection between GPT and txn/dispute bot.
- One Data – Integration with One Data platform to enable various functions like check dispute status, cancel dispute, etc.
- One Identity – Integration with One Identity to make secure and seamless connectivity between GPT and txn ai agent.
- Enterprise Receipts Platform – To fetch the digital receipts of the transaction.

3. INPUT FOR ENGINEERING

3.1. Business Goals

The goal of this exercise is to leverage the capability of LLM models to achieve the higher automation rate for the defined journey so more CM satisfaction. We can achieve this by utilizing AI Agents and Open AI LLM (ChatGPT-4) to overcome previously inaccessible inefficiencies in the transactions/disputes journey. In the MVP, we set out to target opportunities for automation where the current bot:

Past the MVP

- We estimate an uplift of 11.5% to containment rate of the current Transactions/Disputes bot as per PRD.
- We anticipate that building this bot on an LLM based framework will enable a faster time to market on any additional capabilities we identify as necessary for further containment rate/RTF gains.

3.2. Summary of learnings from Discovery

The Transaction and Disputes journey is the highest volume journey in AskAmex today, accounting for 30% of total chats in July 2023 across all channels.
Transaction-Disputes journey enables the Cardmember to manage their transactions and raise disputes if eligible, view existing disputes and report fraud too.

Card Selector

Card selector enables agents to ask for a user's card/ set a user's card as soon as the chat session starts. To develop a seamless and user-friendly card selecting process for the users (customers), we consider three major scenarios associated with selecting/setting a card in the chat:

1. User does not mention their card in their initial message and have only one active card.
2. User does not mention their card in their initial message and have multiple cards.
3. User mentions their card (full card name or partial information regarding the card (e.g., my plat card, my card ending - 12345))

We leverage LLM capabilities in the card selecting process to extract (and potentially map user's card input to full card name) if the user decides to use free form typing (in scenario 2), or if the user mention their card name (full name or partial information) in their message. In addition, we use LLM capabilities to validate the user's response regarding the card selecting process. For instance, if the user mentions a card that does not exist on their profile as an active card, we will detect this and inform the user to choose a valid card.

We need the following pieces of information to be extracted from the available cards on user profile during the card selecting process:

- Account token

- Product description (full card name)
- Display account number (last five digits)
- Supplementary or primary (required for dispute eligibility)

Transaction Search

Current NLP model is deriving the entities from the utterance but there are some short comings like MRMG approval process for any updates, limited capability compared to LLM models. With LLM models we can reduce escalations.

- A significant number of chats are being escalated today due to the ongoing failure to accurately identify the relevant entities, resulting in unnecessary escalations.
- The integration of LLM is expected to enhance the accuracy of identifying relevant entities from Cardmember utterance, which will subsequently improve the ability to recognize correct transactions.

Performing transaction search is a main capability that other capabilities, such as disputing a transaction, depend on. The search results depend on the available information about the transaction to be searched (e.g., merchant name, transaction date, transaction amount). The user can choose to provide some information about the transaction, or they may not. If the user provides information regarding which transaction they are looking for, we will leverage the LLM's capability to extract this information as entities and we use the extracted information to perform search. Otherwise, the search function will return the 10 most recent transactions to the customer allowing them to click on the transaction or specify more information regarding the transaction they are looking for (e.g., user says: I think it was from Amazon). Potential challenges:

- User has a typo in the provided information regarding the transaction or they use abbreviations (e.g., AMZ instead of Amazon or Amaxon instead of Amazon)
    - Possible solutions: semantic search, leverage LLM capability to find transactions with similar merchant name, fix most common typos by leveraging past conversational data
- Dates should be formatted such that it matches the format of the posted date associated with the transaction.
    - Possible solution: Instruct LLM to map the uttered date to a specific format.
- SOR failure
    - Possible solution: detect SOR failure based on SOR status codes and escalate if required

Information needed from returned transaction:

- Transaction ID
- Merchant name
- Transaction date
- Transaction amount
- Transaction type (debit or credit)
- Transaction status (pending or posted)
- Transaction sub-category (lodging or hotel vs other sub-category)
- Transaction description

Transaction search is also accompanied by providing "options" (depending on the transaction's status, type, and sub-category) regarding a transaction to the user once that transaction is found and confirmed by the user. These options include:

- I'm all set
- Dispute this charge

- Alert me when posted
- View another transaction
- View digital receipt
- Acquire more information (e.g., transaction description)

Required Process/ Flow:

- Card selector

Required fields for SOR call:

- Account token
- Transaction information (if available, e.g., merchant name, transaction date, transaction amount)

Dispute Transaction

Disputing a transaction requires identifying the user's card and transaction initially. Thus, we need to first acquire information regarding the user's card (mainly for transaction search in the account and ensuring primary card holder) and then enable transaction search to get the transaction ID associated with the transaction to be disputed. Once we have the ID, we proceed with checking the dispute eligibility using SOR call (alternatively, by checking a transaction dictionary built real-time for the user within the chat session). If eligible, we will utter the dispute link to the user (note: we do not automate the dispute submission process), otherwise, depending on the transaction type and sub-category, we provide alternative "options" (e.g., alerting when posted) to the user.

The dispute transaction process will link to the card selector and transaction search flows.

Required Process/Flow:

- Card selector
- Transaction search

Required fields for SOR call:

- Account token
- Transaction information (if available, e.g., merchant name, transaction date, transaction amount)

Dispute Status

Similar to the dispute transaction process, getting the dispute status for a specific dispute-id requires setting the card initially (setting the card leads to identifying the account token required for SOR call). Once the account token (card) is identified, there are 2 scenarios:

1. User already provided the dispute-id (e.g., I would like to check the status of for dispute-id D-12345)
2. User has not provided dispute-id

In the first scenario, we instruct the LLM to extract dispute-id as an entity from the user's utterance and after validating (e.g., via regex) the extracted dispute-id, we make an SOR call to the ReadTransactionDisputesActivityForAccounts.v1 endpoint (passing the extracted account token and

dispute type as the following list: [“OPEN”, “DRAFT”]), which returns a list of available disputes. Then, we use the extracted and validated dispute-id to search for the dispute with the given dispute-id and return the dispute status to the user. If the dispute-id is not available in the list of available disputes, we will return the full list of available disputes to the user and inform them that the given dispute-id is not available in the given disputes list. On the other hand, for the second scenario, once we have the list of available disputes, we present that list to the users (as buttons) so they can inform the bot about the dispute. Once the user provides the dispute information (types the dispute-id or clicks on the dispute), we use the dispute-id to retrieve the dispute from the disputes list and return the dispute status to the user.

Required Process/Flow:

- Card selector

Required fields for SOR call:

- Account token
- Dispute types = [“OPEN”, “DRAFT”]

Cancel Dispute

The cancel dispute process is similar to the dispute status process as it requires invoking the card selector to obtain the account token (once a card is identified), which is a required field for making SOR call for obtaining the dispute status (and all other required information) and canceling the dispute (i.e., ReadTransactionDisputesActivityForAccounts.v1 and DeleteTransactionDispute.v1). Once we have the account token, similar to “Dispute Status”, if the dispute-id is available from the user, we retrieve the available disputes list from the SOR call, look for the dispute using the dispute-id, obtain the required information from that dispute (dispute category) and make the second SOR call to cancel the dispute. Otherwise, if the dispute-id is not available from the user, we present the list of available disputes to the user, once they identify the dispute (and the dispute-id), we repeat the steps above and make an SOR call to delete/cancel the submitted dispute. The SOR call (if successful) will return a “Validation Message” and we inform the user about the returned message from the SOR call.

Required Process/Flow:

- Card selector

Required fields for SOR call:

- Account token
- Dispute id
- Dispute category (Billing, Payment)

Fraud Handoff

The fraud handoff is a flow that follows transaction flow where cardmember is allowed to report a particular transaction as fraudulent. On selecting the transaction and getting the consent from card member, conversation control is transferred to the fraud bot.

CCP Escalation

There are four major scenarios that result in escalating to a CCP:

1. Out of Scope messages

2. If a user comes to the chat while drafting a dispute application (escalate to dispute CCP)
3. If bot detects fraud, conversation will be escalated to fraud CCP

3.3. Requirements

3.3.1 Functional requirements

The following main capabilities have been prioritized for the MVP. The following section details the features that have been identified as part of those capabilities during the discovery phase.

New LLM Framework integrated into Architecture (Conversational Flow):

The LLM Framework built into our existing architecture will leverage existing parts of the platform (i.e. orchestration, memory, etc) while integrating with an external LLM that could be easily swapped out with a replacement in the future. The prompts, tools and actions will be broken down in a logical manner to facilitate easy additions of new customer journeys.

Rebuild core bot flows/capabilities in the new LLM agnostic framework

The journeys defined for MVP includes journeys for dispute flow as well as transaction flow. We are going to club the journey under the single agent txn_ai_agent. The journey includes view, create, cancel (excludes automated cancellation by agent) for disputes and view transactions, Show digital receipts, alert me when posted and mobile handoff journeys for transaction.

Although bot to bot transfer is not something a priority for MVP but given that we will be running the agent on same cdms instance we can achieve bot to bot transfer for below scenarios - from transaction agent to

- **dispute_bot** - dispute bot
- **cd_bot** - card declined
- **fraud_bot** - fraud bot
- **pl_bot** - payment late.

Rebuild Card Selector (Enhanced)

This capability is dedicated to better identifying the Card for a Multi-Card Member when the CM has provided details or a vague response. By using the LLM to help identify non-exact entity matches and prompt the Card member when they have responded vaguely to the Card Selector we will improve automation by keeping the CM within the journey.

3.3.2 Non-functional requirements

Two key non-functional requirements have been mapped to this moment (list being expanded):

1. Bot should be available 99.9% of time.
2. Bot should be able to handle ~790K conversations on monthly basis (Web + Mobile).

Following are current and anticipated increases in volume on both CDMS and Agner to support transaction and dispute bot journeys and various API integrations.

Number of Requests based on traffic on current CDMS bot

Number of Requests	Transaction Agent	Reference
Per Day	22109	Sample Splunk Query
Per Hour	1879	
Per Minute	223	
Per Second	21	

Ramp up Strategy:

This bot initiates traffic at a min volume of around 2% using green_gold_experience and gradually will increase the traffic to 100% with two steps by adding premium_experience to reach approximately 35% traffic and finally by adding non_premium_experience to reach 100%.

Current Response times:

- Average CDMS TXN bot response time: 1742 milliseconds
- Average CDMS Dispute bot response time: 868 milliseconds

Assumptions:

- Considering 15 business hours/day
- Estimated TXN/Dispute bot volume 79k per month (Includes both Web and Mobile Channel)
- Latency Requirements – Review the latency of LLM API's and revisit the platform latency thresholds.
- Logging Requirements – The standard application logging will be done as part of MVP along with logging of the LLM predicted Slots & Predicted Actions. Any Other additional logging/Bot monitoring requirements are not considered in scope for MVP.

Latency Requirements:

The below table provides the latency comparison based on the current platform and anticipated increase in latency with LLM integration (The LLM latency stats are based of other use case implementation in askamex. These stats must be revisited once engagement is completed and the models that will be used will be determined).

Current CDMS Response time (Avg)	***Anticipated Agents response time (Avg)***	***Anticipated LLM Response time (Avg)***	***Current Platform timeout Configurations***	***Total***
~1500ms	*~1500ms*	~3000 to ~5000 ms	*6000 ms*	*~6500 ms*

With the addition of LLM, the average response time is estimated at 10 Seconds due to latency. Once the response time for the new model and integration is obtained, the latency configuration must be revisited.

Based on the previous LLM integration use case the max allowable tokens are 80K per minute and it will be revisited once the engagement & use cases are finalized.

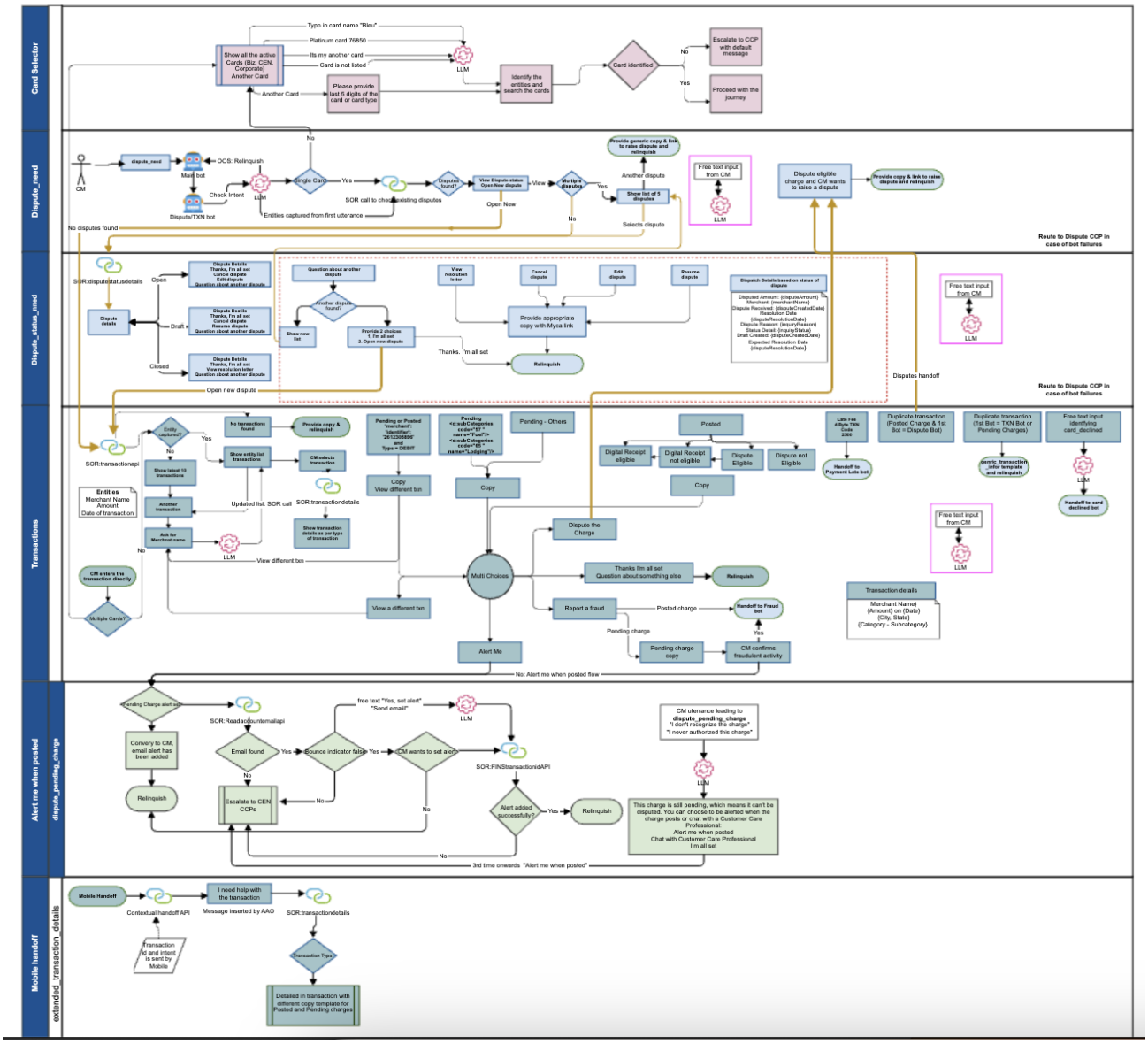
With the latency with LLM integration, the bot response time to user will increase and the user possible can respond before the bot response, can potentially result in chat escalation and impact to automation rate as well. The response time must be a watch item to be tracked with regards to user experience impacts.

Assumptions:

- With the integration of LLM, there will be additional latency while processing the CM request.
- The chatbot's automation rate is at least, on par with that of the existing CDMS Dispute + Txn Bot
- The chatbot's customer RTF is at least, on par with that of the existing CDMS Dispute + Txn Bot
- Incoming call volumes will decrease as more people opt for the chat functionality, due to a better experience.
- The chatbot automation will reduce the workload on servicing CCPs.
- Integrating with an LLM will not significantly increase latency to negatively impact customer experience (Which will be revisiting once the engagements for LLM are finalized, and the model/performance numbers are obtained)
- LLM User case approvals, Compliance and other approval required for GenAI initiatives will be obtained at the project level.

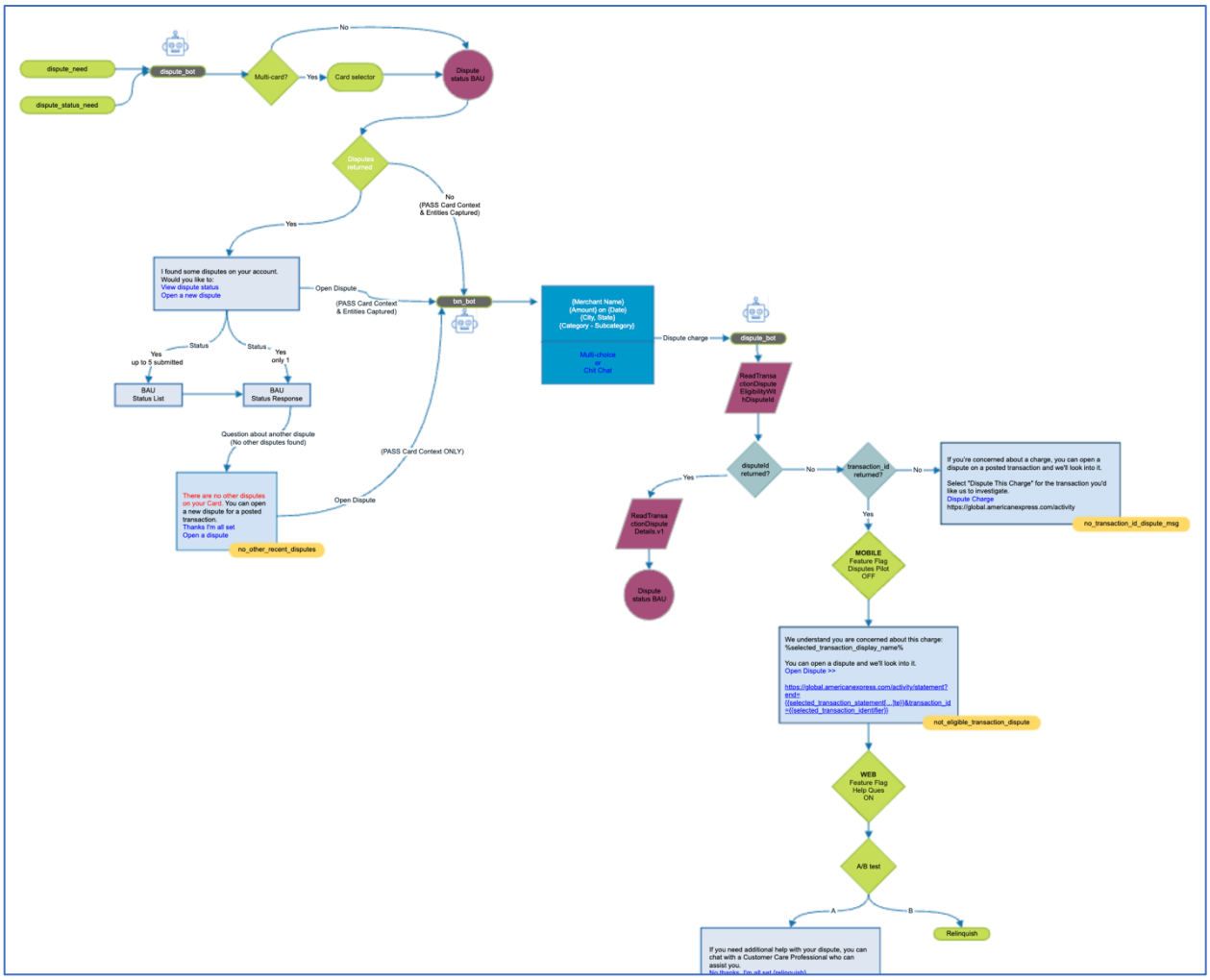
As part of MVP the new agent's platform will be routed through CDMS and Functions which hosts applications in IPC2 Zone1 and IPC Zone 2 with 2 pods/zone & 5 workers/pod, which results in 20 worker threads/application. In summary 20 worker threads serving approximately 3 requests/second at response time of 2-3 milliseconds. Hence, concluding that this basic infrastructure in place is good enough to handle the existing volumes.

3.4. Product flow

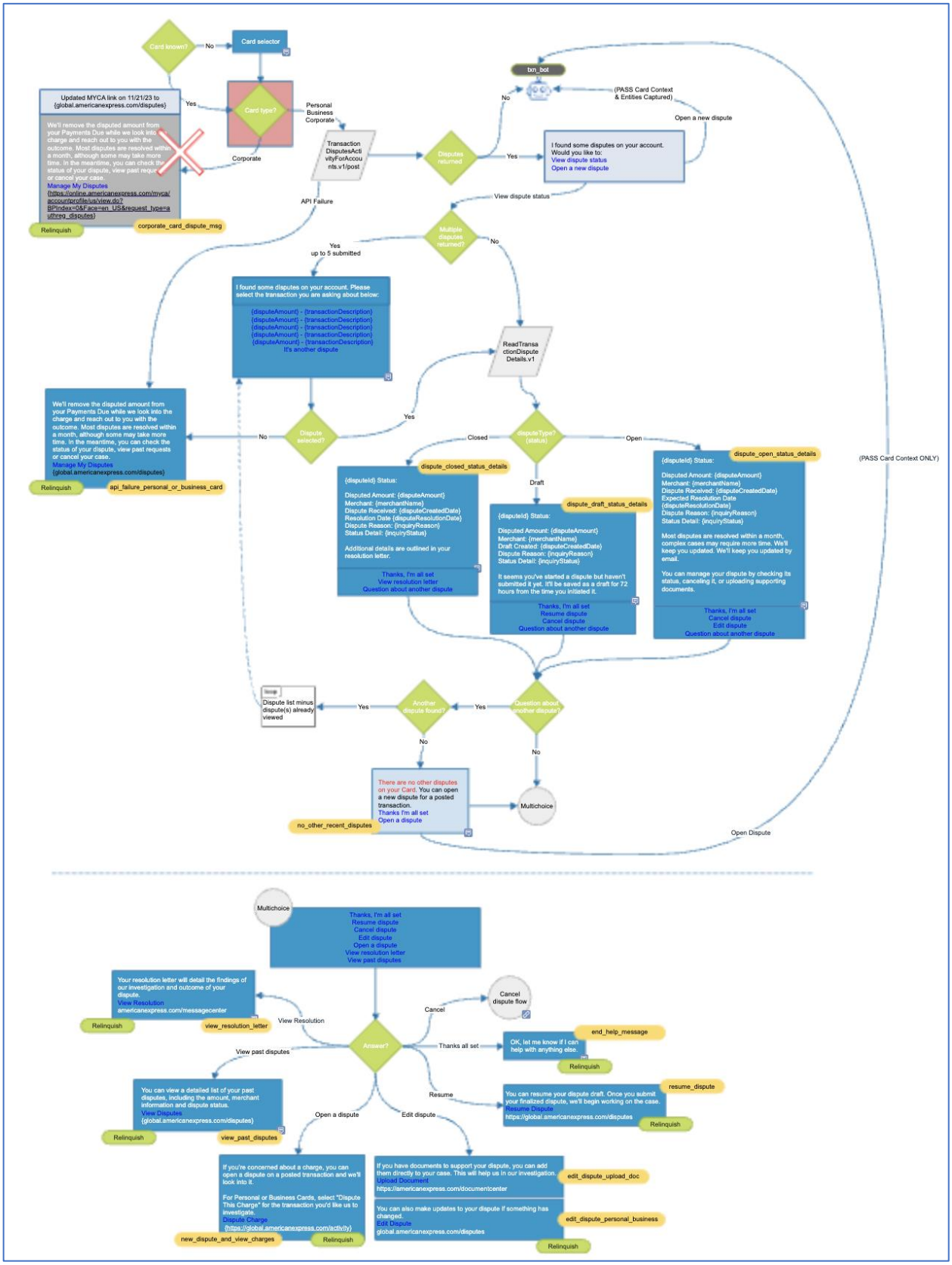


Breakout flows as below -

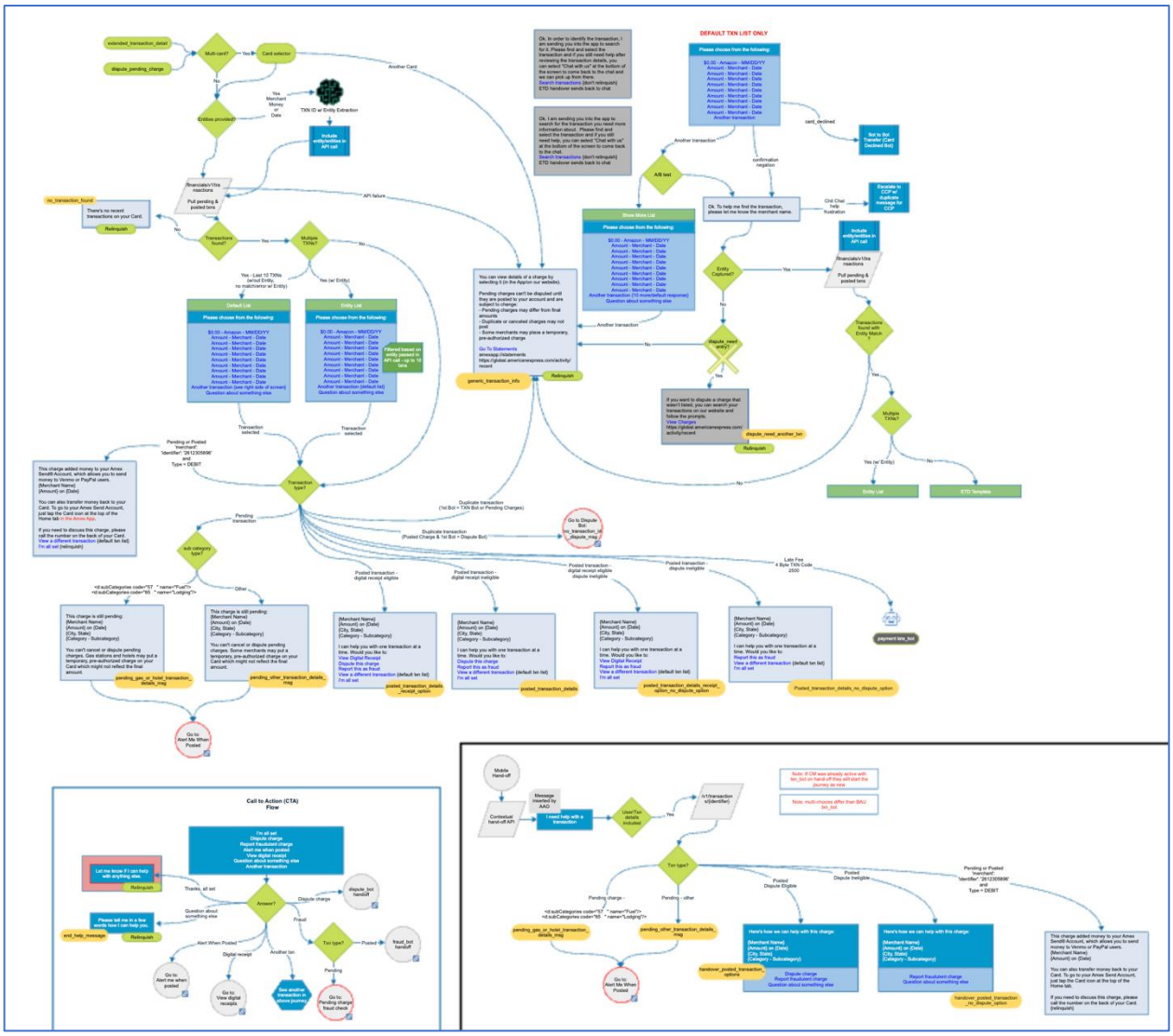
Dispute Create Flow:



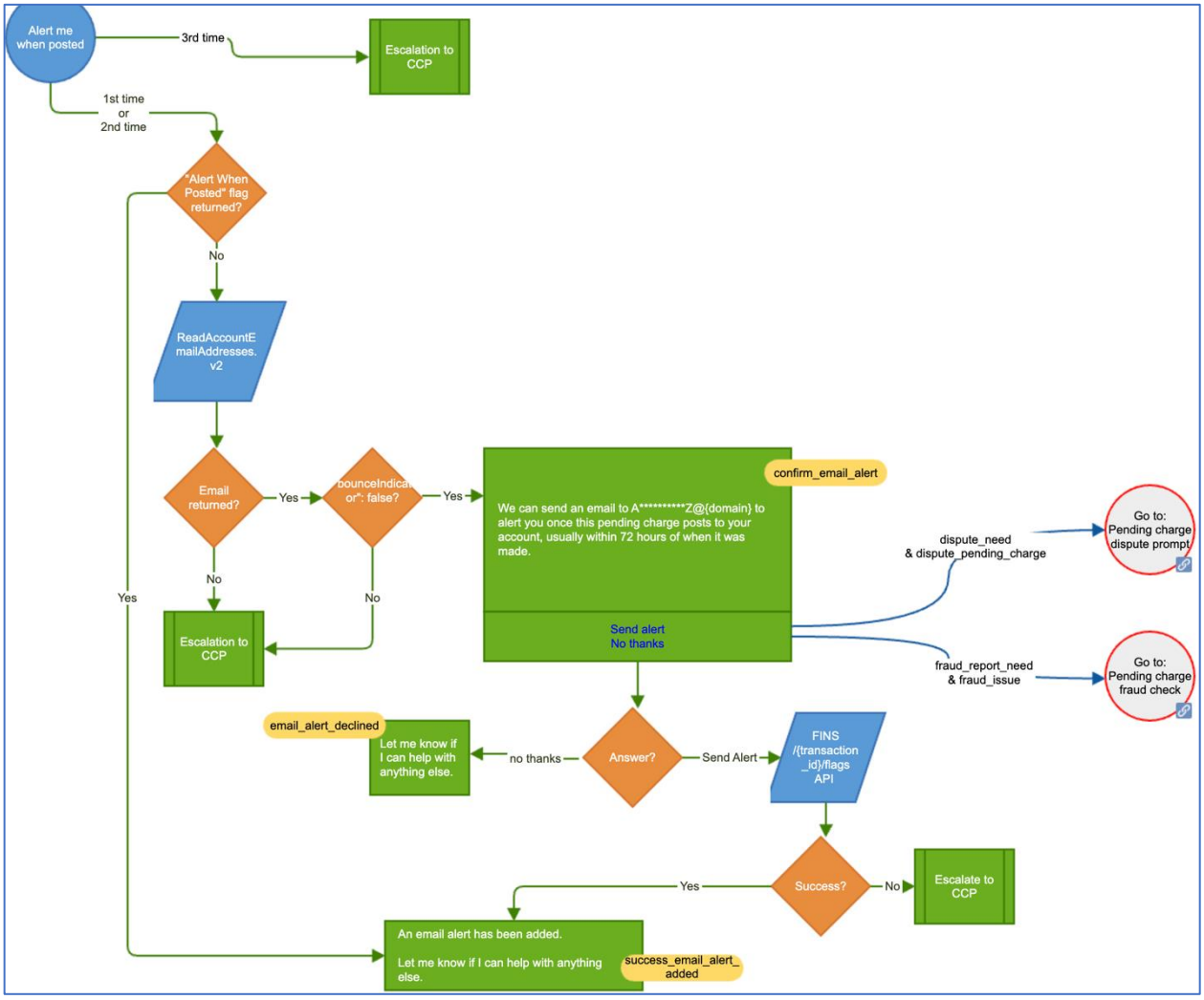
Dispute Status Flow:



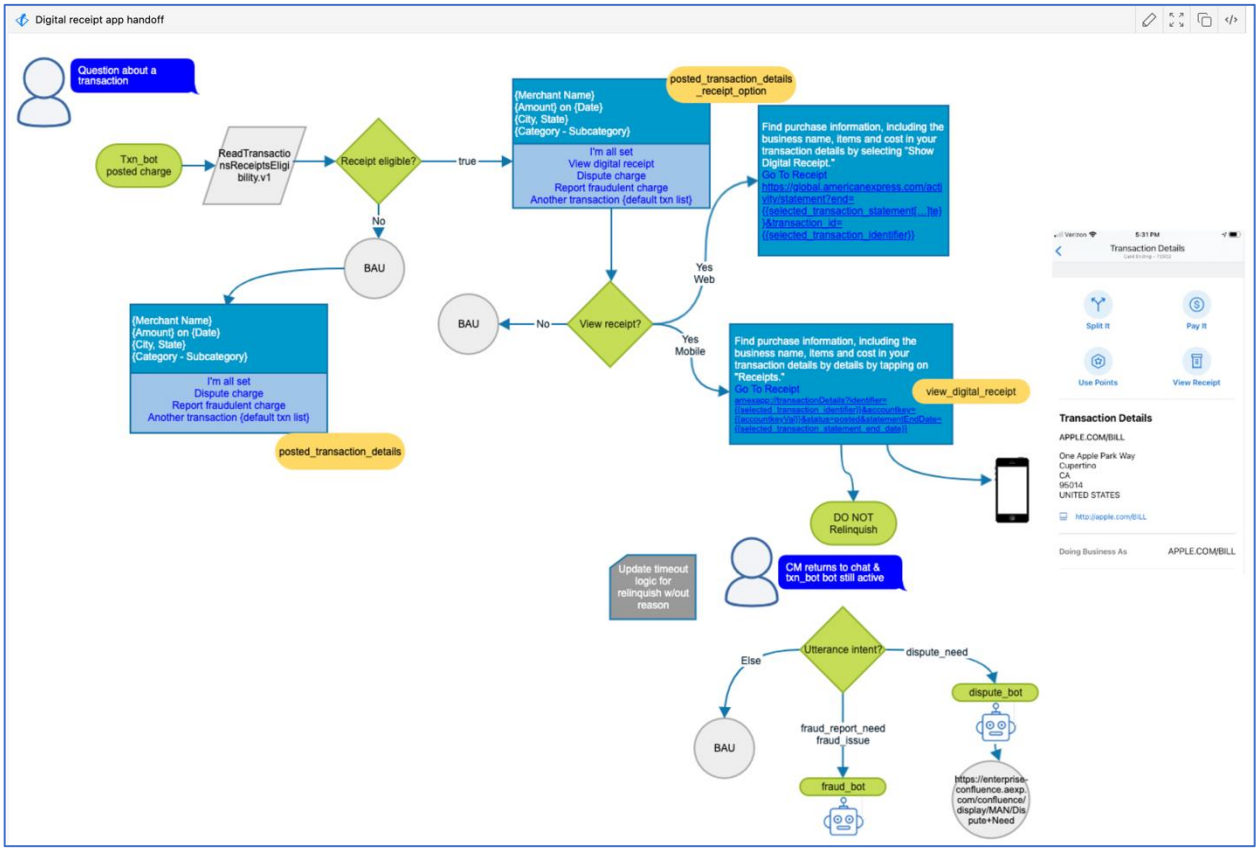
Transaction Flow:



Alert Me When Posted:



Digital Receipt -



3.5. Existing Solutions

The new LLM-based framework to support the business needs of transaction agents will make use of a few of the framework components that are currently being used in the existing platforms.

Existing framework and platform capabilities:

- Deployment process and architecture would remain same as part of this MVP requirement. No new deployable applications will be introduced.
- Unless otherwise stated, most of the critical communication focused on components around this change is on top of HTTP/HTTPS protocol.
- Agents will be built in a channel agnostic like the other microbots hosted on CDMS.

User interaction and session management:

- If user types any utterance/resolves to intent which is not handled by defined flow, it will be considered as out of scope from bot standpoint & Requests will be redirected to Main Bot.
- Any exceptions/errors throughout the process would result in requests routing to Servicing CCPs.

Data and model integration:

- Will make use of existing card type and card number NER capabilities to capture card name and numbers wherever possible.

The following is a high-level overview of different components and libraries involved in running the microbots in CDMS platform and Functions

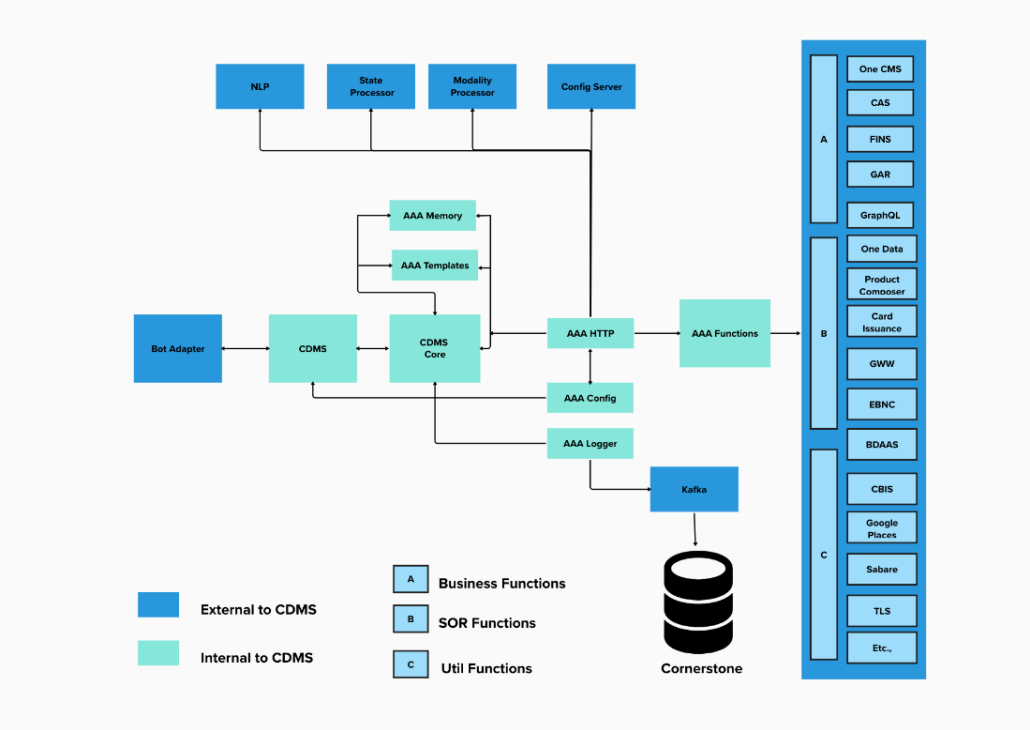


Figure: *Ask Amex CDMS and other dependent libraries.*

The following is a closer look at different microbots hosted in CDMS as of today, and various SORs that the functions layer interacts with to support bot needs

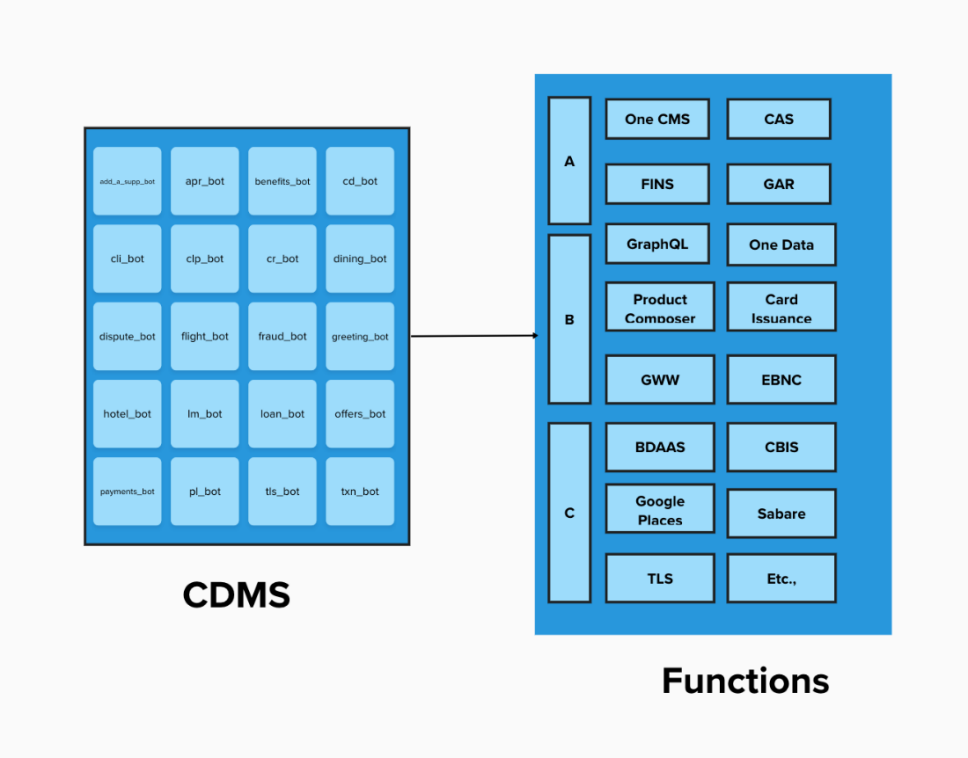


***Figure**: Different Bots hosted in CDMS & Critical SOR integrations within Functions.*

4. DATA SCIENCE

Following are high level steps throughout the process of model delivery to accommodate LLM transaction agent needs:

- Data Collection
- Data Labelling
- Agents Pipeline Optimization
- Agents Pipeline/ Model Evaluation
- Model On-boarding (E2 testing)
- Model Documentation
- Model (E2 Integration testing)
- Model Validation (MRMG)
- Model E3 deployment

4.1. Input Data, Constraints, and KPIs for Objective Function

Past Data from cornerstone will be used to evaluate the overall actions in pipeline and conversation accuracy.

4.2. Data Science Approach and Feasibility

Model and Timeline Alignment

<< TBD - this table will be updated once I get it from data science team >>

	wk1	wk2	wk3	wk4	wk5	wk6	wk7	wk8	wk9	wk10
	19-Feb	26-Feb	4-Mar	11-Mar	18-Mar	25-Mar	1-Apr	8-Apr	15-Apr	22-Apr
Data collection										
Pilot pipeline										
Data labeling										
Pipeline optimization										
model evaluation										
model onboarding(e2 testing)										
model documentation										
model validation(MRMG)	plan		checkin							

Prompt Details:

As part of MVP, we will be using prompts for communicating and directing the behavior of Large Language Models (LLMs). As part of MVP the following prompt attributes will be passed on to LLMs (Not limited to these few mentioned below)

- Conversation History (Limited to current Bot and current bot session context)
- Journey details supports and details descriptions of the journeys
- Slots details like card product, last 5 digits, payment amount, status

The LLM will be providing the details of slots collected, predicted journey and additional parameters which could assist in understanding user request and switch between journeys/Single responses.

The prompts are subjected to MRMG approval process and based on the recommendations, prompts has to be updated. The MRMG approvals are mandatory for enabling these LLM capabilities to productions.

The below journeys are considered in scope for MVP

- Dispute Create
- Dispute Status
- Dispute Cancel
- Dispute handoff
- Transaction List
- Transaction posting Alert
- Transaction handoff

5. ARCHITECTURE

5.1. Target Architecture Overview

The following is a high-level view of the target architecture with critical Ask Amex components in it. We have shown the transaction agent which will support the journey to handle the transaction and dispute journeys.

Ask Amex Agent Core: Framework utilized by the AI agents to manage conversations with the card member.

Agent Tools: Hosts functions utilized by the agent that are connected to Amex SORs to accommodate needs during the conversation.

Generative Core: Extensible frameworks for integration with genAI models, as part of MVP the integration will be using Open AI.

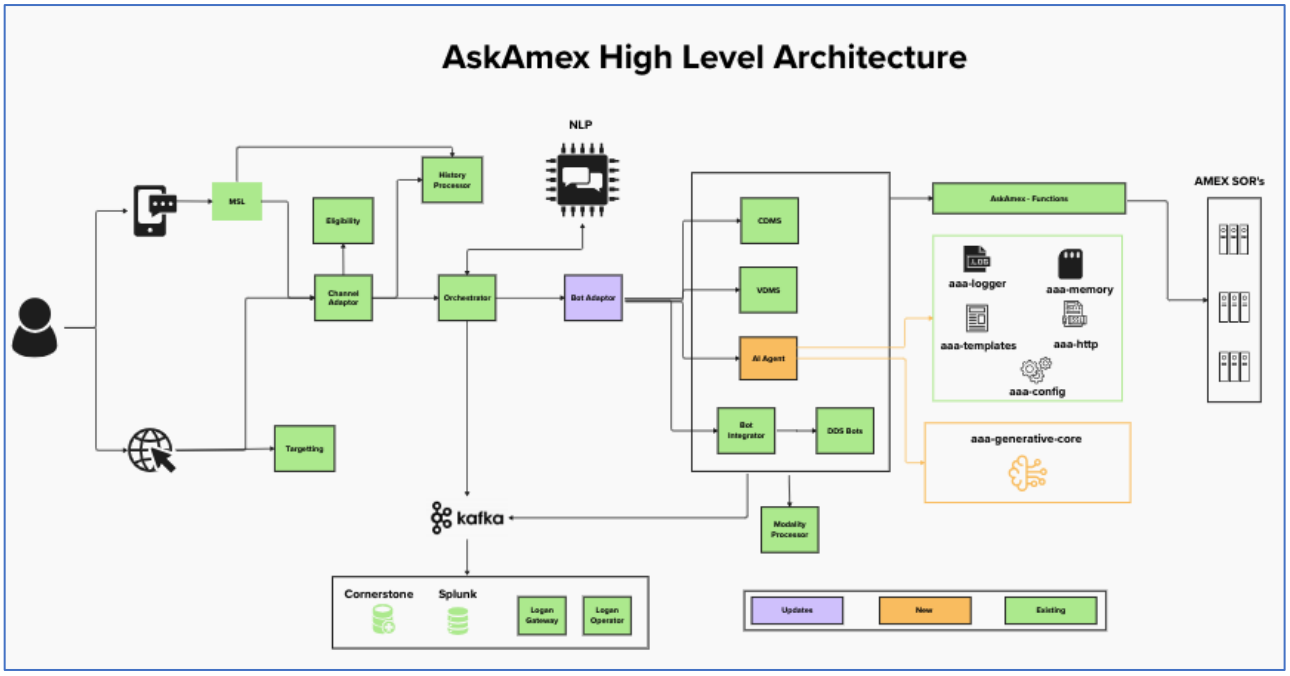


Figure: *AskAmex Architecture - Chat Ecosystem*

5.2. Solution architecture overview

The AI Agents framework consists of the following components which will be developed to support conversation journeys.

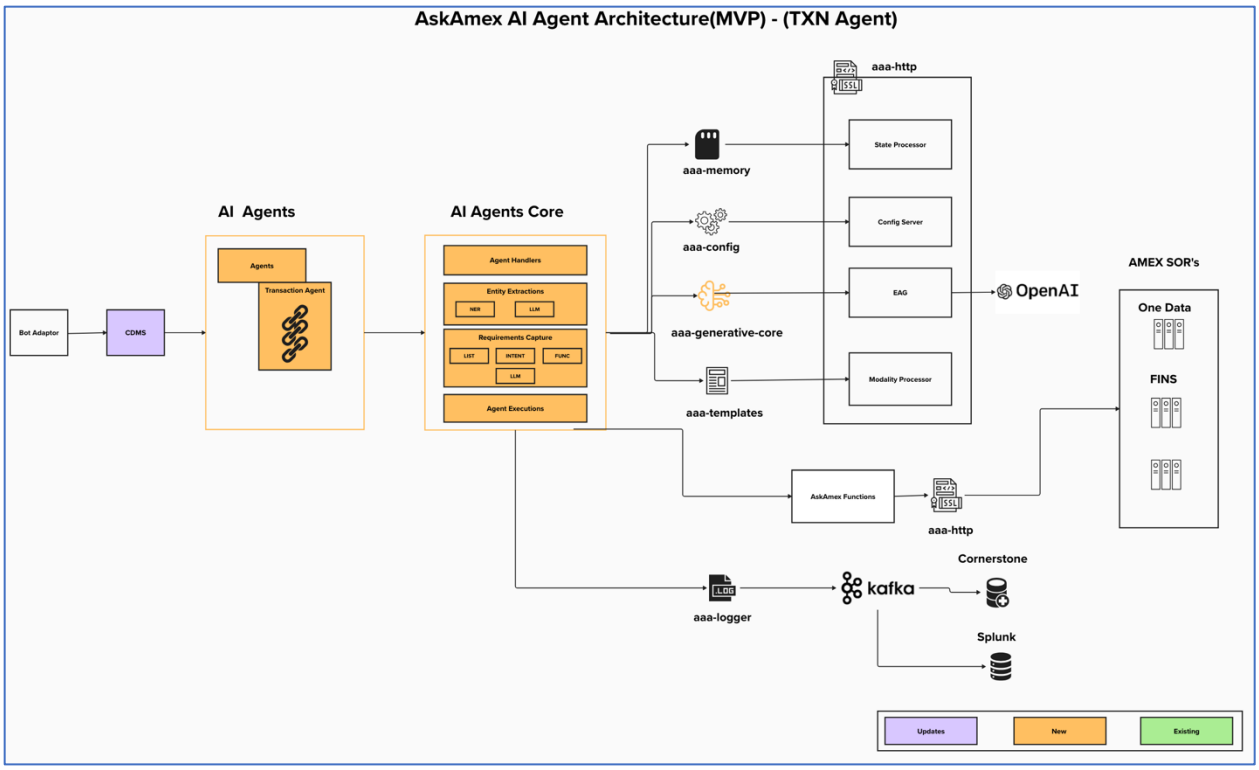
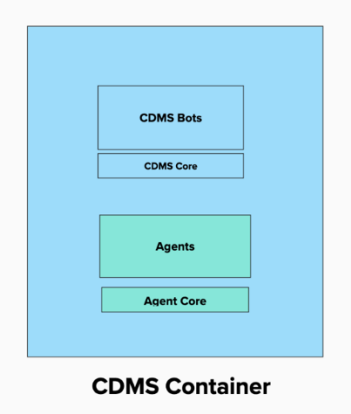


Figure: *Detailed MVP architecture of the AI agents and core frameworks*



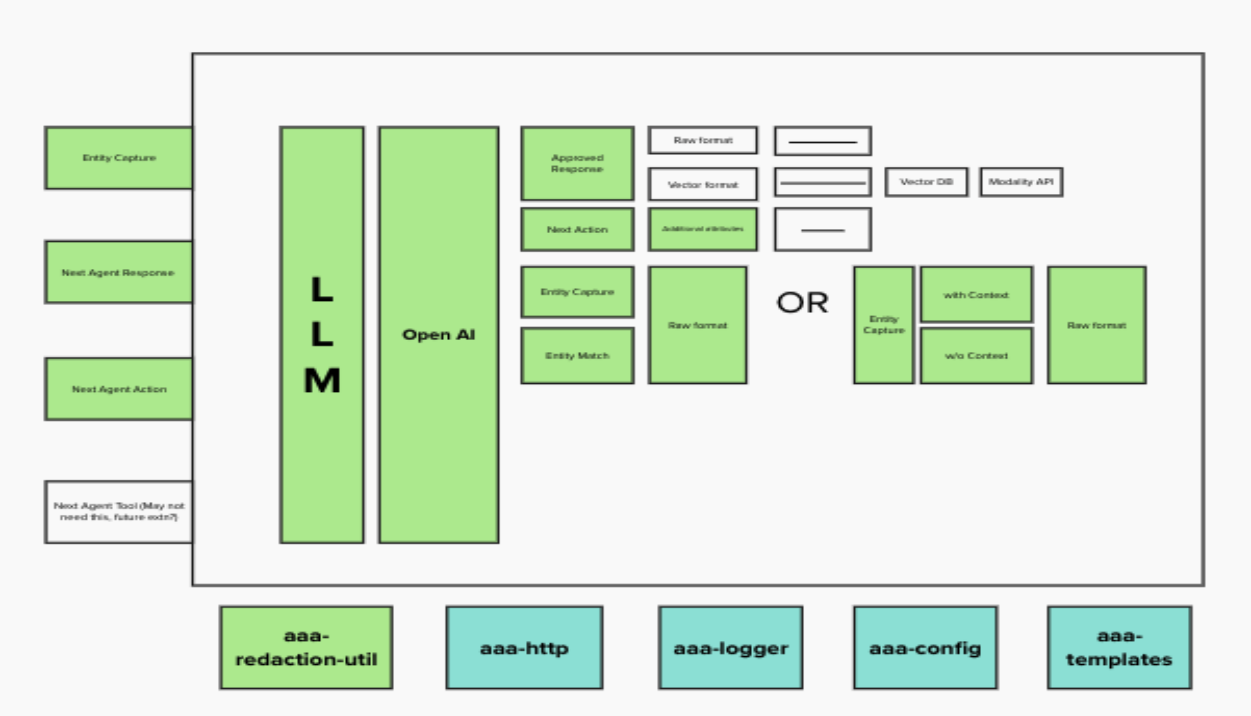


Figure: *Detailed MVP architecture for askamex-generative-core*

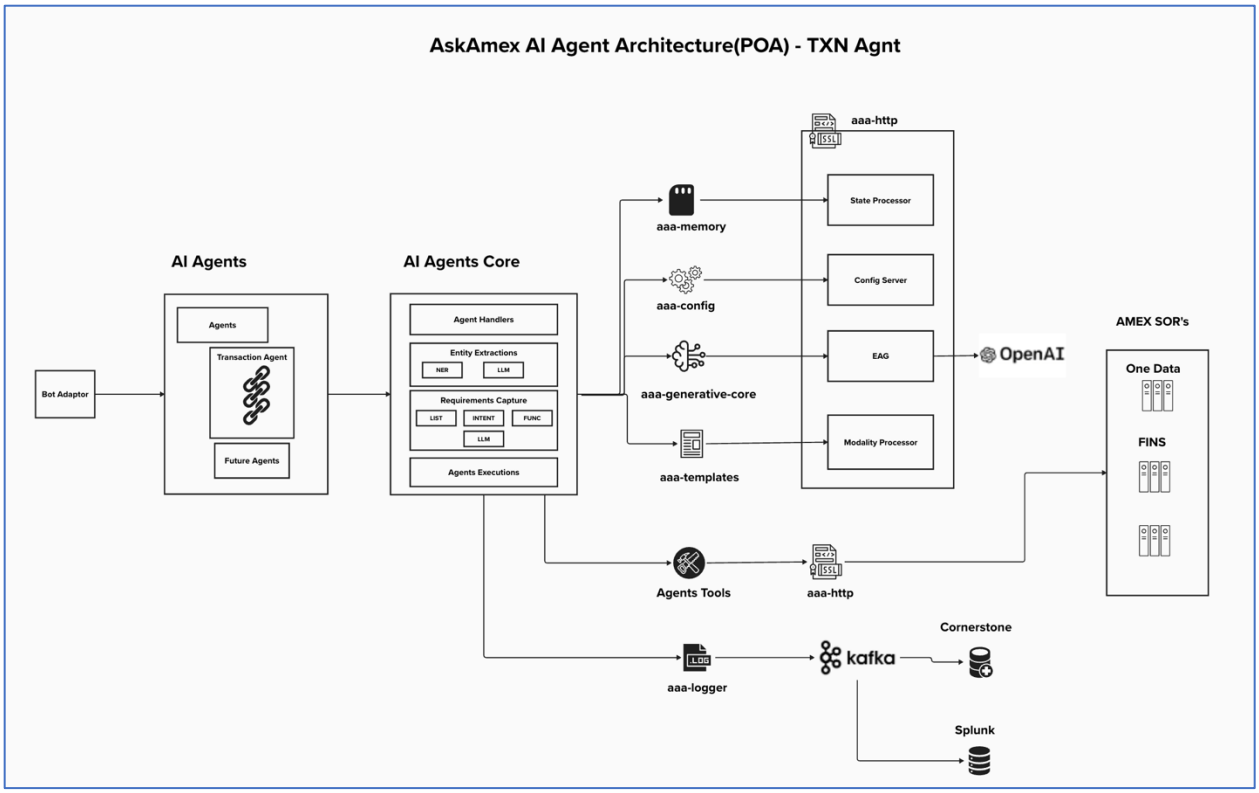
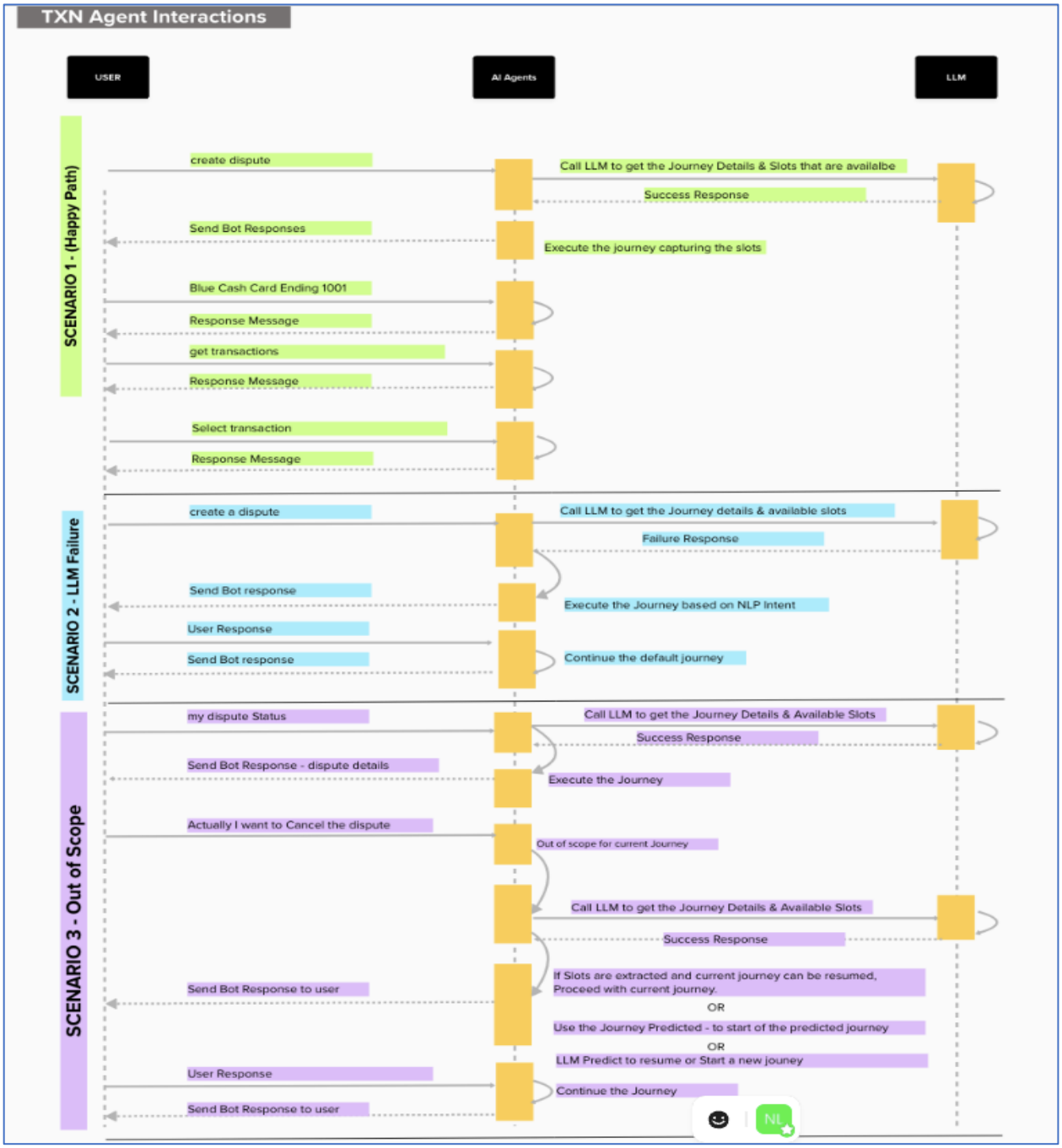


***Figure**: Detailed POA (Point of Arrival) architecture of the AI agents and agent core framework*

Process/User Interaction's:



5.3. Integration and Existing Landscape

As part of MVP the intents will be routed to CDMS and CDMS will be routing the request to AI agents. This will be an interim solution until the platform team makes changes to route traffic to the new AI agent systems.

The following are the Business and SOR functions that will be used as part of MVP

Business Functions:

- **get_account_transaction_v3** (get_account_transaction_details_sor_v1)**:**

  Returns the list of all transactions with core transaction details, that belong to the given input account id (account_id) for the requested statement end date or date range. Client can request for posted transactions, pending charges, or fraud charges resource by providing 'status' input

- **read_account_email_address_v2:**

  Returns an electronic mail address for the given account token.

- **alert_when_posted_v1:**

  This API applies the specified flag type on the provided transaction identifier.

SOR Functions:

- **get_account_dispute_activity_sor_v1:**

  Returns the list of all dispute associated with the accounts.

- **get_dispute_details_sor_v1:**

  Returns dispute details of the selected disputes.

- **delete_transaction_dispute_sor_v1:**

  Cancels the transaction dispute.

- **read_transaction_dispute_eligibility_sor_v1:**

  Allows the user to check if the selected transaction is eligible for dispute creation or not. Also returns associated latest dispute details if there are any duplicate disputes available on file.

- **one_xp_variant_find_sor_v1:**

  Retrieves the experiment based on locator.

- **create_transaction_dispute_sor_v1:**

  This function will create a dispute for a given account and a transaction.

- **update_transaction_dispute_questionnaire_sor_v1:**

  run through a series of interactive questions during the dispute creation journey. The responses/answers provided during this journey will dynamically determine the following questions and thus determine the creation of an appropriate dispute type in Global Dispute Management System.

- **get_account_transaction_details_sor_v1:**

  Returns the list of all transactions with core transaction details, that belong to the given input account id (account_id) for the requested statement end date or date range. Client can request for posted transactions, pending charges, or fraud charges resource by providing 'status' input

- **read_transactions_receipts_eligibility_sor_v1:**

  Returns the eligibility for receipts for a set of transaction ids, receipts could be available via a receipt provider like Ethoca, or specific merchants like Square, iTunes, or be user uploaded.

As part of the MVP, we will be leveraging these existing APIs, and no additional changes are required as part of this integration.

The following is a list of other APIs & Tools which applications will continue to make use to support conversational experience to the user.

- Modality API
- Session Context API
- Conversation Control API
- Request control API
- Receive API
- Bot Admin Portal

The Existing libraries will be used as part of the new agent framework to interact with askamex ecosystem

CDMS Core:

- Enhancements to route the user request to AI agents

Template Library:

- Content management system where AI agent responses are being configured and managed

Automation http Library:

- This library is used to manage all the network calls required by the applications. AI framework will be using this existing framework to manage network interactions.

Automation memory Library:

- This library is used to state management required by the applications. The AI Agent framework will be using this existing framework to manage agent's state. These frameworks interact with SP components to access Couchbase.
- As part of MVP the content will be available at the transaction agent level and the context will not be available once the journey is completed.
- To maintain the conversation context we will be adding additional document in the couchbase which will have the same lifecycle as an existing cdms memory document with the key appended with "_conversation"

Automation Logger Library: Manage logging and Kafka publishing event to Cornerstone and Splunk systems.

Automation Config Library: Configuration Management framework

6. RISKS AND DEPENDENCIES

6.1. Engineering Risks and Dependencies

Risk/Dependencies	Description	Category (e.g., business, integration)	Mitigation strategy
Scalability LLM based bot framework	High dependence on LLM for the bot ecosystem, In case of LLM API Downtimes, there will be High impact to bot platform and conversation will be escalated to CCP	Business & Integration	To avoid a high dependence of LLM models, the usage will be done only when the agents are not able to understand the user. In case the agent can understand the user, it will proceed with the default journey. There by mitigating the risk of automation drop and increase in chat volumes that are being escalated to ccp's.
Maintainability	Changes to actions/tools will have approval dependencies with MRMG groups.	Business & Integration	The MRMG approval processes are required only when changes are being Made to Prompts, which can add to development timelines.
Scalability	LLM latency & Availability metrics	Integrations	LLM-Performance details - Cost basis, Token Size frequency and TPS support, response time etc.
Integrations	Changes are required at platform level to integrate With new LLM based agent frameworks	Integration	The mitigation plan is to be route the journey to existing CDMS bots and CDMS bots will be routing the request to Agent's framework. Once the Routing changes are prioritized in the platform the switch can be done to the newer agent's platforms.
BAYA-Review process (CAR to CAR Access)	Application integration with LLM required to go through compliance review process	Business & Integration	The implementation of the proposed solutions is dependent on the BAYA Compliance review process.
Bot Contextual Information	The current bot contextual information will be available in the memory and will be used for this MVP.	Integration	As part of MVP the current bot context will be used. The approach will be revalidated as we iterate on the approach post MVP.

	Once the bot completes the end of flow the bot context will not be available		
LLM Model Version & API contracts	The API contract and Model details for integration will be available only after the Engagement	Integration	This can be a potential blocker if the model version and API contracts are not finalized.
PRSA approval	PRSA	Approval	Need to secure PRSA Approval

6.2. Data Science Risks and Dependencies

Dependency	Dependency Group	Mitigation strategy
Prompt Approval	MRMG	The Prompt required for using LLM will have to undergo approval process with the MRMG group, and this is a dependency for production launches
Model/Pipeline Evaluation	DS/MRMG	Model/LLM needs to be evaluated in terms of performance in predicting the journey and entity extraction
Model Explainability	DS/MRMG	Evaluate the model explainability in predicting the output

6.3. Open Items

Open Item	Solution Approach	Resolved?
Open items in the Scope excel listing requirements	Discussion with product team for the resolution of queries	N
KPI Indicators	Key Performance indicators – to measure the pre and Post journey Evaluations and compare the various other Solutions	N
LLM latency & Availability metrics	LLM-Performance details such as latency & availability Metrics(Cost Basis, Token Size frequency and TPS support)	N
Production Access	Access to prod endpoints, connectivity details and Version of LLM API must be confirmed	N
Chat LLM use case approvals	Approval requirements for the Chat LLM use case must be determined	N

Security / One Identity Onboarding for LLM-Access	One Identity Onboarding must be completed since the application will be part of the new CAR.	N
Request Control/Convo Control API’s	The Request Control and Convcontrol APIs are currently available in existing platform and Building These capabilities on the new platform in contingent on the timeline and capacity	N