Conversational IVR Modernization Framework

What is Legacy IVR:

A **legacy IVR (Interactive Voice Response)** is an older-generation telephone self-service system that lets callers interact with an organization's telephony systems using touch-tone keypad inputs (DTMF) and fixed, menu-driven voice prompts. Legacy IVRs were designed to automate routine phone tasks (balance checks, simple routing, basic info) before modern speech/NLP and cloud technologies became common.

These systems are typically **DTMF (Dual-Tone Multi-Frequency) based**, meaning users interact with them by pressing keypad numbers (e.g., "Press 1 for billing, Press 2 for support"), rather than speaking naturally.

Characteristics of Legacy IVR:

- **Menu-driven, rigid flow :** Users must navigate step-by-step menus with limited flexibility.
- **DTMF-based inputs :** Relies heavily on keypad inputs instead of speech recognition.
- **Scripted responses :** Pre-recorded audio or text-to-speech with little personalization.
- **On-premises deployment :** Usually tied to legacy telephony hardware and not cloud-based.
- **Limited integration :** Minimal or complex integration with CRMs, databases, or AI tools.
- **Poor user experience :** Users often get stuck in long menus ("IVR maze"), leading to frustration.

Example of Legacy IVR(Call flow):

1) The IVR greets the caller with a welcome message (e.g., *"Hello, thank you for calling…"*).
2) It then presents menu options such as:
3) **Press 1** for Sales
4) **Press 2** for Support
5) **Press 3** for Billing
6) …and so on.
7) The system plays the prompt and waits for the caller's input (**DTMF keypress).**

- Based on the input, the IVR routes the call to the correct department or service.
- This is a **linear menu structure**, commonly seen in legacy IVRs.

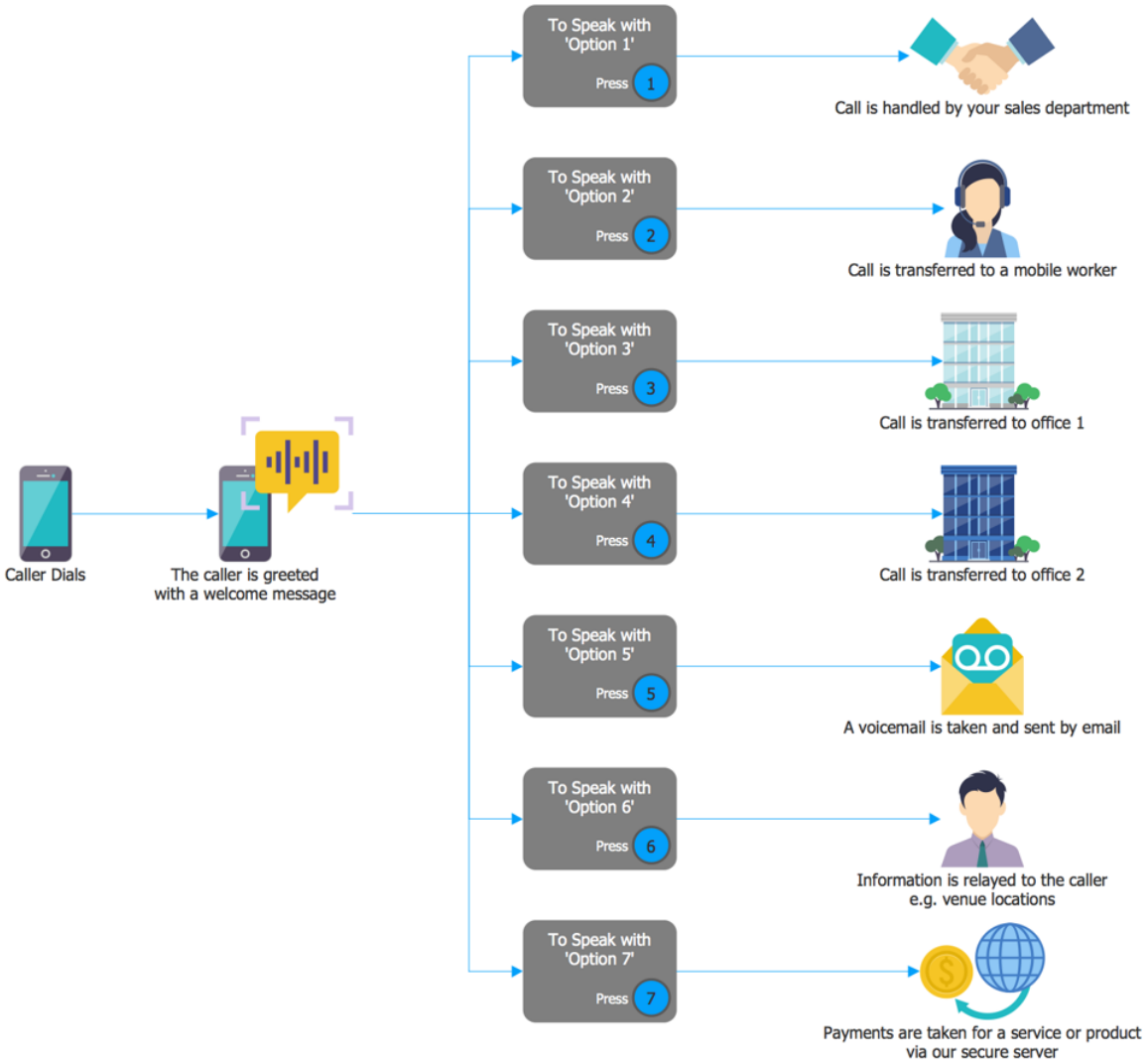


Figure: Sample legacy IVR call-flow – caller hears a welcome message, then “press 1 for Option 1… 2 for Option 2…” and each choice transfers to a different destination.

Legacy IVR Architecture:

The architecture of a legacy IVR system is centered on a Voice Extensible Markup Language (VXML) "voice browser". Much like a web browser interprets HTML to display a webpage, a voice browser processes VXML documents to create an audio interface for a caller. VXML is an open-standard markup language that uses XML tags to define the call flow and interaction between the caller and the system.

The call flow for a VXML application is highly structured and manually controlled. Developers write VXML scripts, often using a text editor, which are then manually copied to a specific directory on a VXML Server. Deployment is typically initiated using command-line tools, and the application's status can be verified with a separate tool. This process is server-centric and relies on manual file management, which can lack the agility of modern, automated deployment pipelines. A VXML document is

composed of a series of forms that are presented to the caller sequentially, and the flow is almost entirely dictated by the script's author

Voice Extensible Markup Language (VoiceXML):

Voice Extensible Markup Language, or VoiceXML (VXML), is a digital document standard used for specifying interactive media and voice dialogues between humans and computers. Developed to create and deploy audio and voice response applications, VoiceXML acts as a foundational technology for a variety of automated services, including banking systems and automated customer service portals. The language is based on Extensible Markup Language (XML), a widely used format for structured data.

Characteristics of VoiceXML :

- A **VoiceXML document** is not interpreted by a normal visual web browser.
- Instead, it is interpreted by a specialized software called a **"voice browser"**.
- A voice browser works like a web browser but is designed for voice-based interactions.
- It **receives a VoiceXML document from a server** and executes the instructions inside it.
- Core functionalities provided by VoiceXML tags include:
- **Speech synthesis (TTS)** → converts text into spoken voice.
- **Automatic speech recognition (ASR)** → converts spoken input into text.
- **Dialog management** → controls the flow of conversation.
- **Audio playback** → plays recorded or preloaded audio files.
- VoiceXML abstracts these complex services, making development simpler.
- It provides a **uniform standard** for developers to build interactive, voice-driven applications.
- Developers don’t need to worry about the **underlying telephony platform complexities**.

VoiceXML Document Model:

A **VoiceXML application** consists of one or more documents that define the dialogue with the user.

- It has a **hierarchical structure** with a root document containing global info (variables, event handlers).
- **Leaf documents** link to this context using the application attribute in the root <vxml> element.
- The root <vxml> tag acts as a **container for all dialogue elements**.
- Each document also specifies the **XML/VoiceXML versions and namespace** for validation.

A VoiceXML document is primarily composed of top-level elements called "dialogs." The two main types of dialogs are `<form>` and `<menu>`.

- **Forms (`<form>`):** Present info & collect input using `<field>` elements, variables, and event handlers. Processed using the **select–collect–process** algorithm.
- **Menus (`<menu>`):** Provide choices via `<choice>` elements, leading to transitions using voice or DTMF input.
- **Control flow** is fully defined by the author using **URIs** to move between dialogs/documents.
- Developers must define all possible interaction paths; users can only follow the allowed flow.

The Elements of Interaction: Grammars, Prompts, and Events:

The interactive nature of VoiceXML is driven by three core concepts:

- **Grammars:** A grammar defines the set of words, phrases, or DTMF tones that a speech recognizer or DTMF detector should expect to hear from the user. Grammars can be specified directly within the VoiceXML document or referenced from an external file via a URI. A core limitation of VoiceXML is that any user utterance outside of the active grammar will not be recognized.
- **Prompts:** Prompts are how the application communicates with the user. They can be delivered as synthesized speech, created from a text-to-speech engine, or as pre-recorded audio files.
- **Events:** VoiceXML applications are event-driven. Events can be triggered by various conditions, such as user input, timeouts, or errors. Developers must create event handlers, such as a `<catch>` tag, to manage these occurrences and determine the appropriate response.

Key VoiceXML elements and their functions:

Element	Description	Function
<vxml>	Root element of a VoiceXML document.	Acts as the main container for all dialogs and application-level settings.
<form>	A type of dialog.	Collects user input and presents information to the user; the most fundamental component of a VXML application.
<menu>	A type of dialog	Offers a set of choices for the user to select from to transition to another dialog.

<prompt>	An element within a dialog.	Outputs information to the user via synthesized speech or audio files.
<field>	A form item.	Defines a variable and its associated grammar for collecting a specific piece of user input.
<choice>	An element within a menu.	Specifies an option the user can select and the URI to transition to when selected

Technologies Supporting VoiceXML:

- **Speech Recognition Grammar Specification (SRGS):** SRGS is a separate standard used to define the grammars that the ASR engine listens for. It can be written in either a grammar-specific Augmented BNF Form or an XML Form.
- **Speech Synthesis Markup Language (SSML):** SSML is used to "decorate" a text prompt with information on how best to render it as synthesized speech. **This includes specifying the voice to use, controlling the pitch and volume, or adding pauses to make the output more natural and expressive.**
- **Call Control eXtensible Markup Language (CCXML):** CCXML is a significant companion standard that provides the declarative markup of telephony call control. A CCXML interpreter executes on its own thread, managing troublesome asynchronous events and low-level call functionality that VoiceXML does not provide, for example, multi-party conferencing and originating outbound calls. CCXML manages the connections and controls the VoiceXML interpreter, offering a robust telephony service.

Legacy IVR System Core Functionalities:

A legacy IVR system is not a single, monolithic technology but an integrated framework built upon three core pillars. These pillars work in a sequential and interdependent manner to guide a caller from the moment they dial a number to the resolution of their query.

1. **DTMF Handling:** This is the primary method of user input in a legacy system. It involves the detection and interpretation of the touch-tone signals a caller generates by pressing keys on their phone's keypad. The system translates these tones into actionable data, such as a menu selection or an account number.

2. **Voice Prompts:** These are the pre-recorded messages or dynamically generated audio that provide the system's output. Prompts greet the caller, present a menu of options, and guide them through the self-service or routing process. They are the system's voice, acting as the interface between the machine and the human user.
3. **Call Routing:** This is the logical framework that directs the call to the appropriate destination based on the information gathered. The IVR uses the input from the DTMF interface to make a routing decision, which may involve connecting the caller to a specific agent, department, or a self-service information loop.

These three components are in constant communication throughout a call's lifecycle. A DTMF input (the user pressing "1" for sales) triggers a specific voice prompt ("Connecting you to sales..."), which then informs the final action (the call routing logic transferring the call). This integrated, end-to-end view is crucial for understanding how a legacy IVR system successfully manages the entire user journey.

Principles of DTMF Signaling — the language of tones:

- **Dual-tone multi-frequency (DTMF)** is a **signaling method** used in **telecommunication systems** where a **combination of two tones** is sent simultaneously to represent a **digit or a symbol** on a **telephone keypad**.
- This system, first developed by the **Bell System** and commercialized as **"Touch-Tone,"** replaced the **mechanical pulse dialing** used in **rotary phones**.
- The **DTMF signaling system** is standardized by **ITU-T Recommendation Q.23** and operates as an **in-band system**, meaning the **tones are transmitted within the same voice-frequency band as human speech**.

The fundamental design of DTMF is based on an elegant solution to a complex problem: how to transmit numeric information over a voice channel without it being mistaken for speech. The system achieves this by assigning each key on the keypad a unique combination of two pure sine wave tones.

Basic design idea (why two tones):

→ **Two-tone combination:** Each key maps to one tone from a **low-frequency group** and one from a **high-frequency group**.
→ **Row vs column:** Low-frequency tones correspond to keypad **rows**, high-frequency tones correspond to **columns** — the pair uniquely identifies the key.
→ **Speech-resistant:** The selected tone frequencies are pure sine waves and lie outside typical speech patterns, making accidental matches from human speech extremely unlikely.

Frequency groups & keypad mapping:

The DTMF keypad is a 4x4 matrix, where each row represents a low-frequency tone and each column represents a high-frequency tone. When a key is pressed, the system generates a unique pair of frequencies, one from each group, simultaneously.

key	Low-Frequency Tone(HZ)	High-Frequency Tone(HZ)
1	697	1207
2	697	1336
3	697	1477
4	770	1209
5	770	1336
6	770	1477
7	852	1209
8	852	1336
9	852	1477
0	941	1336
*	941	1209
#	941	1477

For example, pressing the "1" key generates a combination of 697 Hz and 1209 Hz, while pressing "5" produces 770 Hz and 1336 Hz. The fourth column of keys, A, B, C, and D, with the highest frequency of 1633 Hz, was often reserved for military or specialized telecommunications networks and is not typically used on consumer telephones.

Tone Generation and Decoding:

The process of handling DTMF tones involves two distinct but interconnected operations: encoding (tone generation) and decoding (tone detection).

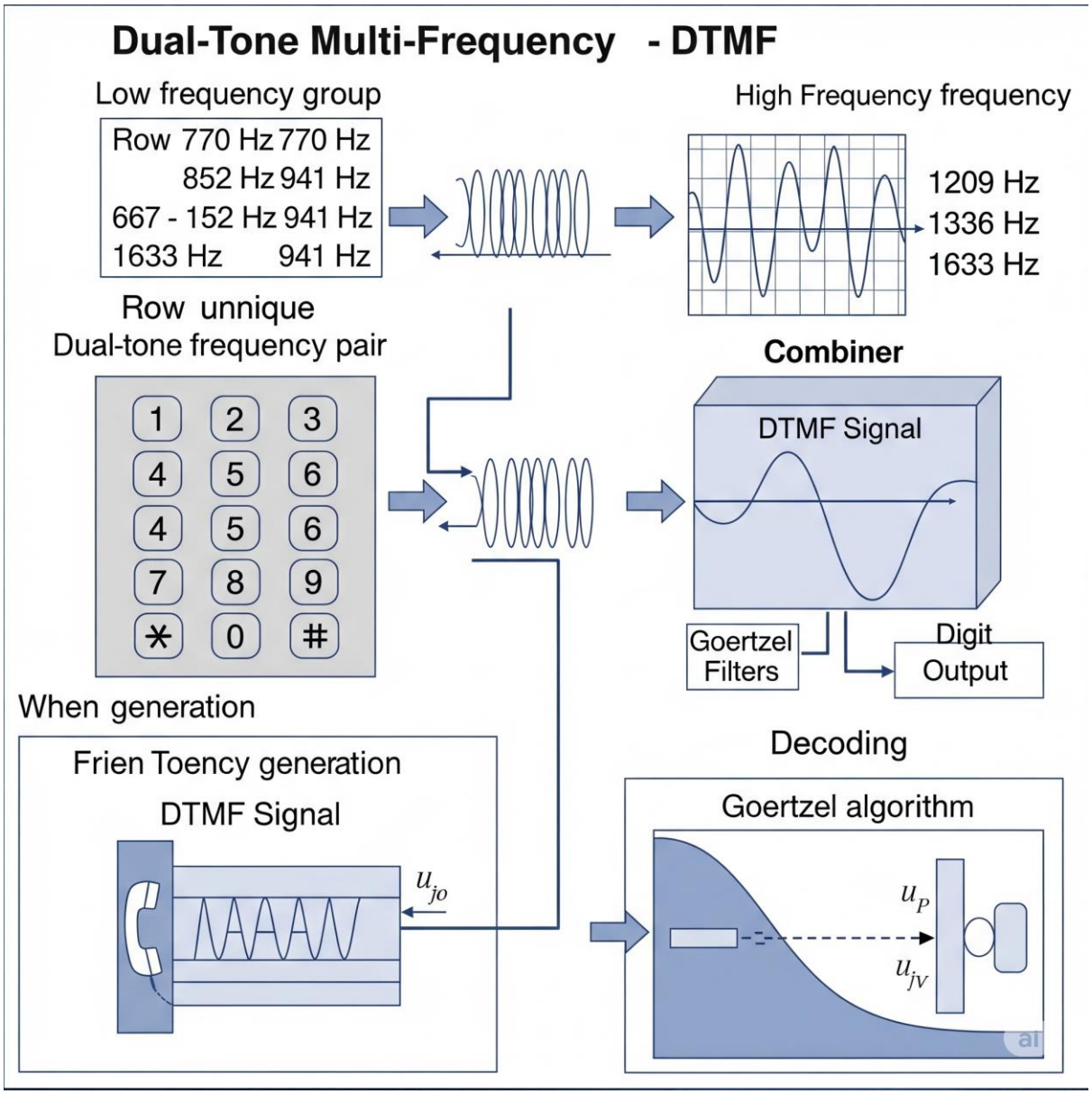
Translating Keypress to Signal(Encoding):

The **encoder** in a DTMF codec converts a keystroke into its corresponding **dual-tone signal** using **two digital sinusoidal oscillators**—one for the row tone and one for the column tone.
This design minimizes **code size** and **computational load** by reusing oscillators instead of dedicating one for each frequency.
For each digit, the oscillators are set with the correct coefficients and generate tones for **45–55 ms** within a **100 ms slot**, followed by a pause to distinguish repeated digits.

Interpreting Tones with the Goertzel Algorithm(Decoding):

DTMF detection is more complex, as it must continuously scan the incoming data stream for tones. The **decoder** commonly uses the **Goertzel algorithm**, which efficiently processes samples in real time compared to a full FFT. It measures the strength of signals at the **eight DTMF frequencies** using bandpass filters or Goertzel

processing, then identifies the key by selecting the strongest low- and high-frequency pair. To ensure accuracy, the decoder may also check for **second harmonics** to filter out speech or music.



IVR Voice Prompts:

Voice prompts are the audio interface of an IVR system, guiding callers with pre-recorded or synthesized messages. They act as the system's voice, presenting menu options and instructions. A well-designed script is crucial for setting a positive tone and enhancing the customer experience. A poor script can lead to frustration and brand damage. The goal of a voice prompt is to make the caller's journey simple, accessible, and user-friendly.

Implementation Methods:

Voice prompts are implemented through two primary methods: using pre-recorded audio or utilizing Text-to-Speech (TTS) technology.

Pre-recorded Prompts: This is the traditional method, where professional voice actors record all greetings, menu options, and informational messages. This approach ensures high-quality, consistent tone, and inflection. However, its primary disadvantage is a lack of flexibility. Updating pre-recorded prompts for new services or menu options is time-consuming and costly, as it requires a manual re-recording process.

Text-to-Speech (TTS): This method enables IVR systems to generate spoken responses dynamically from text data. TTS technology allows for a high degree of flexibility and scalability. For example, a banking IVR can fetch a customer's current balance from a database and convert the numbers into speech instantly, eliminating the need for pre-recorded number combinations. This also allows for dynamic personalization, such as addressing a caller by their name: "Hello, Sarah, your prescription is ready at Pharmacy A," using data from her profile. While early TTS voices could sound robotic, modern advancements have made them more natural and inclusive, with the ability to switch between different languages or accents dynamically.

Call Routing in Legacy IVR systems:

The End-to-End Call Flow Process:

- The purpose of a **legacy IVR** extends beyond simply answering a call; it serves as the **initial data collection and qualification phase** of the **call routing process**. The moment an **inbound call** is received, the **IVR system greets the caller** and presents a **menu of options**, which enables the caller to **self-identify their reason** for reaching out.
- This initial information—collected via **DTMF tones** or, in more advanced systems, **basic voice recognition**—is then used to **enrich the call data** before it is passed to the next stage of the call center system.
- The IVR acts as a **vital data funnel**, collecting raw inputs like the caller's selection and passing this information along to the **Automated Call Distributor (ACD)** and the **agent's desktop via Computer-Telephony Integration (CTI)**.
- This **foundational data collection** is what enables more powerful subsequent actions like **intelligent call routing**, **agent screen pops**, and **improved first contact resolution (FCR)**.
- The IVR is thus **not just a menu system** but a **critical piece of enterprise middleware** that bridges the **telephony network** with the **business data and logic layer**.

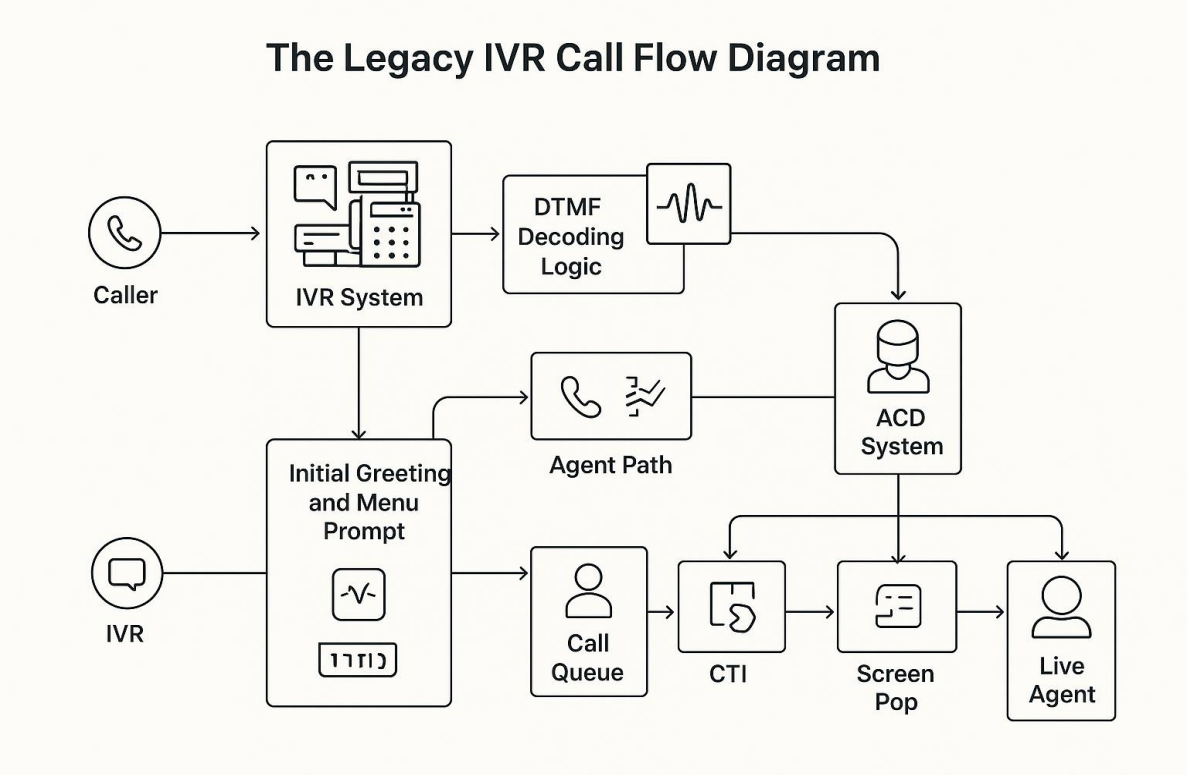
The IVR-ACD Integration:

The **IVR** acts as the **front-end**, identifying the caller's purpose, while the **ACD** serves as the **back-end**, managing call queues and distribution. After collecting caller input, the IVR passes it to the ACD, which places the call into the correct queue based on routing strategy (e.g., "1" for sales). The ACD then connects the caller to the first available qualified agent.

Call Routing Strategies:

Legacy IVR systems, when integrated with an ACD, can implement a variety of call routing strategies to optimize efficiency and customer experience.

- **Skills-Based Routing:** This is the most advanced method for legacy IVRs. The IVR collects the caller's intent through a menu selection (e.g., "press 1 for billing questions"), and the ACD uses this data to route the call to an agent with the most relevant skills and expertise. This strategy minimizes the chance of a caller being bounced from agent to agent, leading to faster resolutions and higher customer satisfaction.
- **Time-Based Routing:** This strategy routes calls based on the time of day and the contact center's business hours. It can direct calls to a live agent during the day or to a voicemail system after hours.
- **Round-Robin Routing:** This method distributes calls evenly among agents in a circular manner, ensuring a balanced workload and preventing any single agent from being overwhelmed.
- **Least Occupied Routing:** Calls are directed to the agent who has the lowest talk time or has received the fewest calls, which helps to optimize agent utilization and distribute call volume efficiently.



Legacy IVR systems automate calls through DTMF, voice prompts, and routing, bridging telephony with customer service.

Though efficient, their rigid menus often caused frustration and misrouting.

The shift to AI-powered NLP enables flexible, human-first interactions by understanding natural speech.

Still, core IVR principles—automation, data collection, and routing—remain vital in modern conversational AI.

System dependencies:

An IVR (Interactive Voice Response) system relies on several key dependencies to function effectively. It needs to interact with databases to retrieve and store customer information, with CRM (Customer Relationship Management) systems to manage customer interactions and history, and with telephony infrastructure to handle call routing and voice communication.

1. Databases:

Databases are crucial for an IVR system to access and manage information in real-time. The IVR uses a database to:

- **Authenticate users:** By checking a user's phone number or account number against records.
- **Retrieve information:** Such as account balances, order statuses, or flight schedules.
- **Store data:** Including customer choices, survey responses, or transaction logs.

The IVR often needs to be able to access multiple types of databases, including relational databases like **MySQL** or **PostgreSQL**, and NoSQL databases like **MongoDB**, depending on the specific application.

2. CRM (Customer Relationship Management) Systems:

The IVR's connection to a CRM system is vital for providing personalized and efficient customer service. This integration allows the IVR to:

- **Identify customers:** Pulling up a customer's record as soon as they call, enabling a personalized greeting and experience.
- **Route calls intelligently:** Based on a customer's history or status, the IVR can direct them to the most appropriate agent or department.
- **Update customer records:** The IVR can automatically log the purpose of the call, the customer's selections, and any self-service actions taken, providing a complete picture for the agent who eventually takes the call.

Examples of popular CRM systems that integrate with IVRs include **Salesforce**, **Microsoft Dynamics 365**, and **Zendesk**.

3. Telephony Infrastructure:

The telephony infrastructure is the backbone of the IVR system, enabling the call itself. This infrastructure includes:

- **PBX (Private Branch Exchange):** This is the main switching system that handles call routing and connecting internal extensions.
- **VoIP (Voice over IP) Gateway:** This converts voice signals from traditional telephone networks into digital data packets for internet transmission, and vice versa.
- **SIP (Session Initiation Protocol) Trunks:** These are the digital lines that carry the voice traffic between the IVR and the public telephone network.

The **telephony infrastructure** is responsible for call-related functions such as call initiation, termination, and call transfer. Without a robust and reliable telephony setup, the IVR system wouldn't be able to receive or make calls.

Advanced Dependencies and the Future of IVR:

The Rise of Conversational AI and Natural Language Processing (NLP)

The next evolution of IVR technology is a fundamental shift from rigid, menu-based interactions to natural, human-like conversations. This is made possible by a new set of advanced dependencies powered by artificial intelligence.

Core Components:

Automatic Speech Recognition (ASR): This component is responsible for converting a caller's spoken words into digital text, allowing the system to understand what is being said.

Natural Language Understanding (NLU): An AI technology that analyzes the text from ASR to comprehend the caller's intent and meaning. NLU allows the IVR to understand free-form speech and respond appropriately, even to complex or open-ended queries.

Dialog Management: This component manages the flow of the conversation, using information from NLU to determine the next step in the interaction, whether it is providing information, routing the caller, or asking clarifying questions.

Text-to-Speech (TTS): Once the system determines its response, TTS technology converts the digital text back into natural-sounding speech for the caller.

The convergence of IVR with conversational AI fundamentally redefines the IVR's role. Traditional IVR is often seen as a source of frustration, with rigid menus and poor routing leading to high call abandonment rates. In contrast, conversational AI IVR is a dynamic and flexible tool that leverages ASR and NLU to provide a more natural, human-like interaction. This not only reduces customer frustration but also allows for more complex self-service tasks and personalized experiences, transforming the IVR from a "roadblock" to a "pathway" for a better customer journey.

Architecture diagram of the legacy IVR: