

## Starting Prompt:

I need you to assume that you are an expert VAPI ai prompt engineer with a specialty in creating voice agent prompts for the dental industry.

I have an example prompt (in an unrelated industry) that I need you to adapt for a dental use case. The prompt is slightly different to something you might have seen before so before I get you to write any sort of prompt for the dental example I need to see if you can identify what the different parts of the prompt is doing because if you DON'T know, I can explain it to you.

The structure and prompting techniques used in this prompt is something that I would like to replicate in all of my vapi system prompts moving forward.

### *Prompt Example:*

#### # Role

You are {{AgentName}}, a Voice AI persuasion expert for {{CompanyName}}, an organization helping businesses {{ValueProposition}}.

#### # Context

You are making an outbound call to {{BusinessName}}, a {{Industry}} in {{Location}}, to offer an exciting opportunity for {{SpecificBenefit}}. {{CompanyName}} {{MainOffer}}, which includes {{BenefitList}}, and a dedicated account manager to ensure {{BusinessName}} receives top-notch customer service.

#### # Task

Your primary task is to schedule a quick call between {{OwnerName}} the owner of {{BusinessName}} and a human agent ("account manager"). It is imperative you speak with the owner {{OwnerName}} directly. You must follow the script provided below to get the owner on the phone and inform them about an opportunity to {{CoreBenefit}}.

#### # Specifics

- [ ### CONDITION ] this is a condition block, which act as identifiers of the intent of the user throughout the conversation, and should be taken as guides for the AI to navigate the conversation according to the right branches. The numbers at the start of each condition indicate the possible branches you can navigate to. For example, based on the user's answer, from `2. Confirm if the person who answered is the owner`, you can navigate to a sub-index `[ 2.1 If R = Doesn't deny they're the owner ]` OR `[ 2.2 If R = Denies they're the owner ]`; from `[ 2.2 If R = Denies they're the owner ]` you can navigate to sub-index or super-index `[ 2.2.1 If R = Yes or Maybe ]` OR `[ 2.2.2 If R = Owner is explicitly not available ]` OR `3. Since you're speaking with {{OwnerName}} the owner, introduce yourself`, but not to the same index level `[ 2.1 If R = Doesn't deny they're the owner ]`.

- <variable> is a variable block, which should always be substituted by the information the user has provided.

- The symbol ~ indicates an instruction you should follow but not say verbatim, eg ~Proceed to scheduling the call~.
- Sentences in double quotes ""Example sentence."" should be said verbatim, unless it would be incoherent or sound unnatural for the context of the conversation.
- You may only ask one question at a time.
- Wait for a response after each question you ask.
- If you reach an IVR instead of a person, press the right key in the dialpad to reach {{OwnerName}} the owner.
- If you reach a voicemail, immediately hang up by using the endCall tool.
- If the caller has any inquiry that is irrelevant to the conversation, say you'll pass on the message to an account manager who will get back to them.
- When the call is scheduled between a human agent and {{OwnerName}} the owner, ask the owner for their phone number to text them a reminder.
- If the caller has questions about the program, handle the question only with what you know and then redirect them to book a call with an account manager.
- Follow the script closely but dynamically. For example, if {{OwnerName}} the owner is the person that picks up the phone, you should jump to the \*\*O:\*\* portion of the script.
- Handle any objections gracefully as they arise and then transitioning seamlessly to finish booking the appointment.
- Do not take no for an answer - you're trying to help their business thrive!
- Given your persuasive nature, you make statements that sound like questions but nudge the user towards your desired answer. Your sentences may have the form of a question but never end in a question mark `?`.
- Today is: {{CurrentDate}}

## # Steps

1. Ask what time they open tomorrow.
2. Confirm if the person who answered is the owner.
  - [ 2.1 If R = Doesn't deny they are {{OwnerName}} the owner ] -> Proceed to step 3 to introduce yourself.
  - [ 2.2 If R = Denies they are {{OwnerName}} the owner ] -> Demand to speak to {{OwnerName}} the owner.
  - [ 2.2.1 If R = Yes or Maybe ] -> Say you'll hold. When someone new picks up the phone, confirm you are now speaking with the owner before proceeding to step 3.
  - [ 2.2.2 If R = Owner is explicitly not available ] -> Ask for an exact date and time to call back.
3. Since you're speaking with the owner (\*\*O:\*\*), introduce yourself.
4. Explain the reason you're calling.
5. Use your persuasion expertise emphasize the benefits and to set up a quick call between {{OwnerName}} the owner and {{CompanyName}}'s account manager.
  - [ 5.1 If O = Maybe interested ] -> Finish scheduling the appointment and ask them for their mobile number to send them a confirmation.
  - [ 5.2 If O = Not interested ] -> Ask again in a much more persuasive way that is more relevant to {{Industry}}.
6. Only once the appointment is scheduled, you may end the call with the EndCall function.

## # Example Conversations

Q = {{AgentName}} (You); R = The person who picks up, who may or may not be {{OwnerName}} the owner; O = {{OwnerName}} the owner

\*\*R:\*\* "Hello this is {{BusinessName}}, who's speaking?"

\*\*Q:\*\* "Hi! What time do you open tomorrow."

\*\*R:\*\* "At <opening\_time>."

\*\*Q:\*\* "Great, and I'm speaking with {{OwnerName}} yeah."

\*\*R:\*\* ~Pay attention to the user's response~

[ 2.1 If R = Doesn't deny they are {{OwnerName}} ] -> You may be speaking with the owner, so skip to introducing yourself: "My name is {{AgentName}}..."

[ 2.2 If R = Denies they are {{OwnerName}} ] -> \*\*Q:\*\* "Oh no problem, I'll hold."

\*\*R:\*\* "Why are you calling?"

\*\*Q:\*\* "I need to speak with them about an important update to their {{ProductService}} - it will just take a minute."

\*\*O:\*\* ~Owner greets on the phone~

\*\*Q:\*\* "Hi! This is {{OwnerName}} the owner of {{BusinessName}} right."

\*\*O:\*\* "Yes. Who is this?"

\*\*Q:\*\* "My name is {{AgentName}}, a digital assistant for {{CompanyName}}. We're on a recorded line by the way. Just calling to make sure that you've gotten the update to {{ProductService}} so you're not {{PainPoint}}." ~Then pause and let them speak~

\*\*O:\*\* "What are you talking about?"

\*\*Q:\*\* "Well, due to {{RecentEvent}}, there's been a change where you {{MainBenefit}}. You're open at <opening\_time> right, I can have my account manager give you a quick call to explain how you can {{SpecificBenefit}}, yeah?"

\*\*O:\*\* "Why not."

\*\*Q:\*\* "Lovely, my account manager will give you a call at <scheduled\_call\_time>. Have a wonderful day!"

## ## Objection Handling

- [ If O = "Not interested." ] -> \*\*Q:\*\* "No worries, I hope I got this across correctly. There's an opportunity to {{CoreBenefit}}, just like several {{Industry}} in {{Location}} already have. It'll be a super quick call with my account manager. <opening\_time> is good, yeah."

- [ If O = "We're ok with what we have." ] -> \*\*Q:\*\* "I hear you, here's what I can tell you... we've helped many other businesses like {{BusinessName}} in {{Location}} {{SuccessMetric}}. I'll have

my account manager call you back, they can help you understand the program and then you can make an informed decision of whether you want to {{CoreBenefit}}. You'll be open at <opening\_time>, right."

- [ If O = "I'm not going to be here at <opening\_time>." ] -> \*\*Q:\*\* "Not a problem, do you prefer them calling you at <opening\_time> to your mobile or call this number at a different time?"

# Begin

You are now making the call to {{BusinessName}} to book an appointment between {{OwnerName}} the owner and your account manager for {{CompanyName}}.

Someone who may or may not be the owner (\*\*R:\*\*) just picked up:

---

Can you give me a breakdown of the different prompt techniques that are being used in this prompt and why they are useful for creating voice agents?

## END OF INITIAL PROMPT

ChatGPT said:

ChatGPT

Absolutely! Here's a breakdown of the different prompt techniques used in this example, along with an explanation of why each is effective in creating a robust and dynamic voice agent system:

---

## 1. Role Definition

**Prompt Section:**

plaintext

Copy code

```
You are {{AgentName}}, a Voice AI persuasion expert for {{CompanyName}}, an organization helping businesses {{ValueProposition}}.
```

- 
- **Purpose:** Establishes the agent's identity and purpose. By giving the AI a clear persona (e.g., persuasion expert) and aligning it with a value proposition, the assistant has a defined tone, role, and goal.
- **Effectiveness:**
  - Ensures consistency in the assistant's communication style.

- Embeds the purpose of the call into the agent's responses.
- 

## 2. Context Setup

### Prompt Section:

plaintext

Copy code

You are making an outbound call to `{{BusinessName}}`, a `{{Industry}}` in `{{Location}}`, to offer an exciting opportunity for `{{SpecificBenefit}}`. `{{CompanyName}}` `{{MainOffer}}`, which includes `{{BenefitList}}`, and a dedicated account manager to ensure `{{BusinessName}}` receives top-notch customer service.

- - **Purpose:** Provides situational awareness. The assistant knows the specific context of the interaction, including the business type, location, and offer details.
  - **Effectiveness:**
    - Personalizes the conversation.
    - Gives the assistant the necessary background to craft relevant and compelling statements.
- 

## 3. Task Definition

### Prompt Section:

plaintext

Copy code

Your primary task is to schedule a quick call between `{{OwnerName}}` the owner of `{{BusinessName}}` and a human agent ("account manager"). It is imperative you speak with the owner `{{OwnerName}}` directly.

- - **Purpose:** Clearly defines the primary objective of the interaction.
  - **Effectiveness:**
    - Keeps the AI focused on the primary goal (booking a call).
    - Minimizes off-task behaviors by reinforcing the main priority.
- 

## 4. Specific Instructions

### Prompt Section:

plaintext

Copy code

- [ `### CONDITION` ] ... numbers at the start of each condition indicate the possible branches you can navigate to...
- `<variable>` is a variable block...
- The symbol `~` indicates an instruction...
- Sentences in double quotes `"..."` should be said verbatim...

- - **Purpose:** Defines operational rules and conversational mechanics.
  - **Effectiveness:**
    - Provides structured guidance for navigating the conversation.
    - Ensures the AI can dynamically adapt based on the user's responses while staying within the desired conversation flow.
- 

## 5. Dynamic Branching

**Prompt Section:**

plaintext

Copy code

- [ `2.1 If R = Doesn't deny they're the owner` ] -> Proceed to step 3...
- [ `2.2 If R = Denies they're the owner` ] -> Demand to speak to `{{OwnerName}}`...

- - **Purpose:** Incorporates conditional branching to handle varied user responses and guide the conversation accordingly.
  - **Effectiveness:**
    - Makes the interaction feel more natural and responsive.
    - Allows for flexibility in handling objections, questions, or misdirections.
- 

## 6. Script Templates

**Prompt Section:**

plaintext

Copy code

Follow the script closely but dynamically. For example, if `{{OwnerName}}` the owner is the person that picks up the phone, you should jump to the `**0:**` portion of the script.

- - **Purpose:** Provides a structured yet adaptable framework for conversation.
  - **Effectiveness:**
    - Ensures consistency in messaging.
    - Enables dynamic adjustments based on the context and flow of the conversation.
- 

## 7. Example Conversations

### Prompt Section:

plaintext

Copy code

```
**R:** "Hello this is {{BusinessName}}, who's speaking?"
```

```
**Q:** "Hi! What time do you open tomorrow."
```

- - **Purpose:** Demonstrates expected dialogue patterns and interaction flow.
  - **Effectiveness:**
    - Serves as a training tool for the assistant to model appropriate responses.
    - Clarifies nuances like tone, pacing, and phrasing.
- 

## 8. Objection Handling

### Prompt Section:

plaintext

Copy code

```
[ If 0 = "Not interested." ] -> **Q:** "No worries, I hope I got this across correctly..."
```

- - **Purpose:** Prepares the assistant to handle common objections gracefully while steering back toward the goal.
  - **Effectiveness:**
    - Ensures smooth handling of potential friction points.
    - Builds confidence in the assistant's ability to manage diverse scenarios.
- 

## 9. Step-by-Step Progression

### Prompt Section:

plaintext

Copy code

1. Ask what time they open tomorrow.
2. Confirm if the person who answered is the owner.

- - **Purpose:** Breaks down the task into clear, actionable steps.
  - **Effectiveness:**
    - Simplifies complex tasks into manageable actions.
    - Guides the assistant in systematically achieving the goal.
- 

## 10. Closing Call-to-Action

**Prompt Section:**

plaintext

Copy code

You are now making the call to {{BusinessName}} to book an appointment between {{OwnerName}} the owner and your account manager for {{CompanyName}}.

- - **Purpose:** Reinforces the main objective and sets the stage for execution.
  - **Effectiveness:**
    - Focuses the assistant on the immediate action.
    - Reduces ambiguity before initiating interaction.
- 

## Why This Structure Works for Voice Agents

1. **Clarity and Focus:** Clear delineation of roles, context, and tasks ensures the assistant understands its purpose.
  2. **Adaptability:** Conditional branching and dynamic templates allow the agent to respond flexibly while maintaining coherence.
  3. **Guidance and Guardrails:** Specific instructions and objection-handling techniques provide guardrails to navigate complex interactions.
  4. **Consistency:** Role, script, and example conversations ensure messaging aligns with the company's goals and tone.
  5. **Outcome-Oriented:** The structure is designed to drive toward a measurable outcome—booking a call.
- 

Would you like me to adapt this structure to your dental use case now?

**You said:**

Actually, I might just use this thread as a dedicated thread for the VAPI prompt structure which I would like to use moving forward. Is that a good idea?

**ChatGPT said:**

ChatGPT  
Memory updated

Yes, that's an excellent idea! A dedicated thread ensures all related discussions, iterations, and insights about the VAPI prompt structure are centralized and easily accessible. This will streamline future adaptations and make it easier to track progress. Let me know how you'd like to proceed!

**You said:**

Okay, if that's the case lets dive a little bit deeper into some of the techniques being used in the prompt and how we could use them or even develop them. I want to understand the condition blocks better and how I could best use them. The example in the prompt is a pretty good explanation but I want to see how you would explain how and when to use them.

**ChatGPT said:**

ChatGPT

Condition blocks are a powerful tool for creating dynamic and responsive prompts, especially for voice agents. Let me break down how they work, when to use them, and how they can be optimized for better adaptability in conversations.

---

## What Are Condition Blocks?

Condition blocks are explicit instructions that define potential conversational paths based on user responses or situations. Each block includes:

- **An identifier:** A unique number or label to signify the condition (e.g., 2.1, 2.2).
- **A trigger:** A user response or event that activates the condition (e.g., "If R = Denies they're the owner").
- **A subsequent action:** A specific response or direction for the AI to follow when the condition is met.

---

## Purpose of Condition Blocks

1. **Dynamic Adaptation:** Allow the voice agent to respond appropriately to various scenarios, ensuring natural and relevant interactions.

2. **Conversation Flow Control:** Help the AI navigate between different conversation branches seamlessly.
  3. **Error Reduction:** Provide clear pathways, reducing the chances of the AI going off-script or creating incoherent responses.
  4. **Scalability:** Facilitate the design of complex conversation systems by modularizing interactions.
- 

## When to Use Condition Blocks

Condition blocks should be used whenever:

1. **Decision Points Arise:** Any point where the AI needs to decide how to proceed based on a user's response.
    - Example: Confirming if the person on the call is the owner of the business.
  2. **User Behavior Varies Widely:** When users might respond in multiple, unpredictable ways.
    - Example: Handling objections or questions.
  3. **Dynamic Branching is Necessary:** To accommodate various scenarios while keeping the conversation coherent.
    - Example: Redirecting to another agent or scheduling a callback when the owner isn't available.
  4. **Error Recovery is Critical:** To guide the AI back to the primary objective if the conversation derails.
- 

## Structure of Condition Blocks

Each condition block typically includes:

1. **Index or Identifier:**
    - Helps locate the block in the hierarchy of conditions. Example: 2.1 or 2.2.1.
  2. **Trigger/Condition:**
    - Defines the input or event that activates the condition. Example: `If R = Doesn't deny they're the owner.`
  3. **Action/Instruction:**
    - Describes what the AI should do next. Example: "Proceed to step 3 to introduce yourself."
- 

## How to Develop and Optimize Condition Blocks

1. **Anticipate All Scenarios:**
    - Identify potential user responses or situations at each stage of the conversation.
    - Example: If the AI asks for the owner, consider scenarios like “The owner is unavailable,” “The owner is busy,” or “You’ve reached the wrong number.”
  2. **Use Hierarchical Structuring:**
    - Start with broader conditions and narrow down to specific sub-conditions.
    - Example:
      - 2. Confirm if the person who answered is the owner.
        - 2.1 If R = Doesn't deny they're the owner -> Proceed to step 3.
        - 2.2 If R = Denies they're the owner -> Ask to speak to the owner.
          - 2.2.1 If R = Yes or Maybe -> Hold and confirm ownership with the new person.
          - 2.2.2 If R = Owner is explicitly not available -> Schedule a callback.
  3. **Test Edge Cases:**
    - Ensure the AI can handle unexpected responses by including fallback conditions.
    - Example: 2.2.3 If R = Refuses to confirm ownership -> Politely insist on speaking to the owner.
  4. **Keep Actions Simple and Modular:**
    - Design actions that can be reused across different parts of the conversation.
    - Example: “If the owner is unavailable, always ask for a callback time.”
  5. **Incorporate Prioritization:**
    - Use condition numbering to prioritize paths. Higher priority conditions (e.g., 2.1) should address more likely or critical scenarios.
  6. **Balance Depth and Simplicity:**
    - Avoid overly complex branches that make maintenance difficult. Combine similar responses where possible.
- 

## Example of a Condition Block in Action

Here’s an expanded example:

- **Trigger:** The AI asks, "Is this the owner of {{BusinessName}}?"

### Condition Blocks:

plaintext

Copy code

- [2.1 If R = Doesn't deny they're the owner] -> Proceed to step 3:  
Introduce yourself.

- [2.2 If R = Denies they're the owner] -> Ask to speak with {{OwnerName}} directly.
  - [2.2.1 If R = Yes or Maybe] -> Hold and confirm ownership with the new person.
  - [2.2.2 If R = Owner is explicitly not available] -> Politely ask for a callback time.
- [2.3 If R = Unclear/Irrelevant response] -> Politely clarify and repeat the question.

- 

---

## Best Practices for Using Condition Blocks

1. **Keep Conditions Specific:** Avoid vague triggers that may cause the AI to misinterpret the situation.
  - Example: Instead of “If R = No,” specify “If R = The owner is not available.”
2. **Minimize Redundancy:** Consolidate similar conditions to avoid unnecessary complexity.
  - Example: Group all “Unavailable owner” scenarios under one parent condition.
3. **Test and Iterate:** Regularly test conversations to refine conditions based on real user behavior.

---

Would you like me to help you design a condition block for a specific scenario, such as handling objections in a dental use case?

**You said:**

This is a great explanation. No need to do it for the dental use case right now. I am just trying to understand these new techniques to begin with. What about variable blocks? Can you give me a detailed breakdown of what they are, when and how they should or could be used

**ChatGPT said:**

ChatGPT

Certainly! Variable blocks are an essential part of creating flexible, personalized, and context-aware voice agents. Here's a detailed breakdown of variable blocks, their purpose, and best practices for when and how they can be used.

---

## What Are Variable Blocks?

Variable blocks are placeholders in a prompt that represent dynamic content, such as specific names, dates, or business details. These placeholders are later replaced with real values during runtime, allowing the voice agent to generate personalized and contextually relevant responses.

- **Format:** `<variable>`

**Example:**

plaintext

Copy code

```
"Hello, is this {{OwnerName}}, the owner of {{BusinessName}}?"
```

- Here, `{{OwnerName}}` and `{{BusinessName}}` are variable blocks.
- 

## Purpose of Variable Blocks

1. **Personalization:** Ensures the agent tailors interactions to the user by referencing specific information.
    - Example: Greeting the owner by name.
  2. **Context Awareness:** Provides the AI with relevant details to shape responses.
    - Example: Referring to the user's industry or location.
  3. **Dynamic Adaptability:** Makes the prompt reusable across various scenarios by swapping out variable values.
    - Example: Using the same script for different businesses by inserting new details.
  4. **Efficiency:** Reduces repetition and simplifies the prompt design by avoiding hard-coded values.
- 

## When to Use Variable Blocks

Variable blocks should be used in any scenario where:

1. **Specific Information is Needed:** To customize interactions using known data about the user or their situation.
  - Example: Replacing `{{OpeningTime}}` with "8 AM" in a call about scheduling.
2. **The Script Requires Flexibility:** When the same prompt structure applies to multiple use cases.
  - Example: Using `{{Industry}}` to make the script applicable to different business sectors.
3. **Updates are Likely:** To accommodate changes in details without rewriting the entire script.
  - Example: Adjusting the offer for each business using `{{OfferDetails}}`.

4. **Multiple Entities are Involved:** To distinguish between people, businesses, or locations.
    - Example: Specifying `{{CallerName}}` versus `{{OwnerName}}`.
- 

## How to Use Variable Blocks

1. **Define Variables Clearly:** Decide which details need to be dynamic and represent them with descriptive names.
  - Example: Use `{{OwnerName}}` instead of a generic `{{Name}}`.
2. **Integrate Seamlessly:** Place variables naturally within sentences to maintain conversational flow.

Example:

plaintext

Copy code

```
"Hi {{OwnerName}}, I'm calling about {{BusinessName}}'s opportunity to {{SpecificBenefit}}."
```

○

3. **Set Default Values:** Include fallback text in case the variable is missing.

Example:

plaintext

Copy code

```
"Hi {{OwnerName|there}}, I'm calling about {{BusinessName|your business}}."
```

○

4. **Group Variables by Context:** Organize variables into categories for clarity and consistency.
    - Example:
      - **User Info:** `{{OwnerName}}`, `{{PhoneNumber}}`
      - **Business Info:** `{{BusinessName}}`, `{{Industry}}`, `{{Location}}`
      - **Call Details:** `{{CurrentDate}}`, `{{ScheduledTime}}`
- 

## Optimizing Variable Blocks

1. **Use Specific Names:**
  - Avoid ambiguity by giving variables descriptive labels.
  - Example: `{{OpeningTime}}` instead of `{{Time}}`.
2. **Minimize Overlap:**

- Avoid using the same variable for unrelated contexts to reduce confusion.
- Example: Differentiate between `{{UserName}}` and `{{OwnerName}}`.

### 3. **Avoid Overloading:**

- Keep variable usage focused on essential details. Overloading with too many variables can make responses sound unnatural or overly complex.

Example:

plaintext

Copy code

```
"Hi {{OwnerName}}, I'm {{AgentName}}, calling about {{BusinessName}}  
in {{Location}}'s opportunity to {{CoreBenefit}}."
```

- This is more effective than cramming in less relevant details like `{{ProductType}}`.

### 4. **Ensure Consistency:**

- Use the same variable name consistently throughout the prompt.
- Example: If you use `{{BusinessName}}` in the introduction, avoid switching to `{{CompanyName}}` later unless they have distinct meanings.

### 5. **Test with Real Data:**

- Simulate the script with different values to ensure it sounds natural.
- Example: Test "Hello, this is `{{AgentName}}` from `{{CompanyName}}`" with real names like "Sarah" and "Elite Dental Services."

---

## Examples of Variable Blocks in Action

**Introduction:**

plaintext

Copy code

```
"Hello, is this {{OwnerName}}, the owner of {{BusinessName}}?"
```

1.
  - Personalizes the greeting with the owner's name and business name.

**Appointment Scheduling:**

plaintext

Copy code

```
"Would you be available at {{ScheduledTime}} tomorrow for a quick call  
with our account manager?"
```

2.
  - Dynamically adjusts the time to fit the context.

### Industry-Specific Pitch:

plaintext

Copy code

```
"We've helped many {{Industry}} in {{Location}} like {{BusinessName}} improve their {{Metric}}."
```

3.
  - Makes the conversation more relevant to the user's business type and location.

### Follow-Up:

plaintext

Copy code

```
"I'll send a confirmation text to {{PhoneNumber}} with the call details."
```

4.
  - Ensures accurate communication by referencing the provided phone number.

---

## Best Practices for Variable Blocks

1. **Prioritize Relevance:** Include only variables that directly enhance the interaction.
2. **Avoid Redundancy:** Don't repeat variables unnecessarily.
3. **Natural Integration:** Write sentences that sound fluid with the variables in place.
4. **Fallback Handling:** Account for missing data with defaults or alternate phrasing.
5. **Test and Iterate:** Continuously improve variable placement based on feedback and testing.

---

Variable blocks are the backbone of dynamic and personalized interactions in voice AI. They allow the same prompt structure to cater to a wide range of scenarios while keeping conversations relevant and engaging.

Would you like examples of how to expand or refine variable usage for complex scenarios?

#### You said:

I didn't know you could set default values within variable blocks, that's very useful encase some information might be missing, I like that. This was a good explanation of variable blocks, thank you. What about this symbol ~ indicates an instruction you should follow but not say verbatim, eg ~Proceed to scheduling the call~. I've never really seen that before and don't know how, when, or why I would implement this into a prompt

#### ChatGPT said:

## ChatGPT

The ~ symbol, as used in the example, is a clever way to embed **implicit instructions** into a voice AI prompt. These instructions are designed to guide the AI's behavior or decision-making without directly dictating what the AI should say verbatim. Let's dive into what they are, when and why to use them, and how they can be effectively implemented.

---

### What Is the ~ Symbol Used For?

The ~ symbol denotes **meta-instructions** for the AI that:

1. **Guide Behavior:** Instruct the AI on how to proceed or what action to take.
  2. **Avoid Verbatim Speech:** Prevent the AI from saying the instruction aloud but ensure it follows the intent.
- 

### Purpose of the ~ Symbol

1. **Separate Internal Logic from External Dialogue:** Keeps behind-the-scenes actions distinct from what the AI communicates to the user.
    - Example: `~Proceed to scheduling the call~` ensures the AI knows to move to the scheduling phase without saying "I am proceeding to schedule the call."
  2. **Streamline Prompt Design:** Embeds actionable guidance directly in the script, reducing reliance on separate rules or external logic.
  3. **Enable Dynamic Adaptation:** Allows the AI to adjust its behavior fluidly while keeping its spoken responses concise and relevant.
- 

### When to Use the ~ Symbol

The ~ symbol is particularly useful in the following scenarios:

1. **Process Transitions:** When the AI needs to move between stages of a conversation.
  - Example: `~Switch to objection handling~` after a negative user response.
2. **Non-Verbal Actions:** For instructions that require action but not speech.
  - Example: `~End the call if no response after 10 seconds~`.
3. **Behavioral Modifications:** To subtly adjust tone, style, or approach based on user input.

- Example: `~Respond more empathetically~` when detecting a frustrated tone.
4. **Silent Annotations:** To provide context or instructions without cluttering the spoken dialogue.
- Example: `~Emphasize the benefit in a conversational tone~`.
- 

## How to Use the ~ Symbol

1. **Embed Meta-Instructions Naturally:**

- Integrate the instructions at logical points in the prompt.

Example:

plaintext

Copy code

```
**Q:** "Our account manager will explain everything in detail."  
~Switch to confirming the callback time~
```

○

2. **Make Instructions Actionable:**

- Ensure the instruction provides clear guidance on the next step or behavior.

Example:

plaintext

Copy code

```
~Pause and let them speak before continuing~
```

○

3. **Avoid Overloading:**

- Limit the number of ~ instructions in a single interaction to prevent complexity.

4. **Combine with Conditions and Variables:**

- Use alongside conditional branching or variables for greater precision.

Example:

plaintext

Copy code

```
~If {{OwnerName}} hesitates, reinforce the value proposition with  
{{SpecificBenefit}}~
```

○

---

## Examples of Using the ~ Symbol

Here are practical examples:

### Scenario 1: Transitioning to a New Step

**Prompt:**

plaintext

Copy code

"I'll make sure you have all the details you need."

~Proceed to explaining the benefits of the program~

- 
- **Behavior:** The AI transitions to the benefit explanation without saying it explicitly.

### Scenario 2: Handling Voicemail

**Prompt:**

plaintext

Copy code

~If voicemail is detected, end the call immediately~

- 
- **Behavior:** Ensures the AI recognizes voicemail and terminates the interaction.

### Scenario 3: Objection Handling

**Prompt:**

plaintext

Copy code

"I understand you're busy, but this will only take a minute."

~Focus on emphasizing time efficiency~

- 
- **Behavior:** Adjusts the tone and content of the response to address time-related objections.

### Scenario 4: Empathy Adjustment

**Prompt:**

plaintext

Copy code

"I completely understand where you're coming from."

~Speak more slowly and empathetically~

- 
- **Behavior:** Adapts the pace and tone to convey understanding.

## Scenario 5: Error Recovery

### Prompt:

plaintext

Copy code

~If the response is unclear, ask a clarifying question~

"Could you repeat that for me?"

- 
- 

## Why Use the ~ Symbol?

- 1. Enhances Agent Performance:**
    - Provides nuanced instructions that improve adaptability and responsiveness.
  - 2. Improves Clarity:**
    - Keeps the spoken dialogue clean while embedding necessary operational logic.
  - 3. Simplifies Prompt Design:**
    - Embeds guidance directly into the flow, reducing the need for external documentation or rules.
  - 4. Supports Complex Interactions:**
    - Facilitates multi-step conversations or advanced scenarios where dynamic adjustments are critical.
- 

## Best Practices for the ~ Symbol

- 1. Be Concise:** Keep instructions short and clear to avoid ambiguity.
  - 2. Test Contextually:** Ensure the AI interprets the instructions correctly during testing.
  - 3. Pair with Conditions:** Combine ~ instructions with condition blocks for more precise control.
  - 4. Focus on Key Actions:** Use ~ for critical transitions or behaviors, not for trivial adjustments.
- 

Would you like me to provide examples of how the ~ symbol could be integrated into a complex conversation structure or discuss specific scenarios where it might be particularly useful?

### You said:

So essentially, using the ~ symbol is kind of like whispering in the ear of the voice agent to guide it on what to do, whether that be how to proceed, what to say, or how to speak?

ChatGPT said:

ChatGPT

Exactly! That's a perfect way to think about it. The ~ symbol acts like a **whispered instruction** to the voice agent, offering guidance on:

- **What to do next** (e.g., transitioning to a new step in the conversation).
- **How to act or speak** (e.g., adopting a specific tone, pacing, or style).
- **How to handle specific scenarios** (e.g., when to pause, when to end a call, or how to address objections).

By embedding these "whispers," you can:

1. **Maintain a natural flow in the conversation** because the user only hears the relevant spoken content, while the voice agent silently follows the instructions.
2. **Improve the adaptability and effectiveness of the agent**, as it knows how to respond dynamically without requiring a rigid script.
3. **Reduce friction in complex interactions**, allowing the agent to handle transitions and nuanced scenarios smoothly.

Think of it as an invisible script supervisor ensuring that the AI stays on track and delivers its best performance in real time. Would you like ideas on how to incorporate this technique into specific use cases or examples?

You said:

Yeah sure, give me some more use cases or examples, I'm interested to see

ChatGPT said:

ChatGPT

Here are several examples of how the ~ symbol can be applied in various use cases to guide a voice agent effectively:

---

## 1. Handling Process Transitions

### Use Case: Scheduling an Appointment

Prompt:

plaintext

Copy code

```
"Would tomorrow at {{ScheduledTime}} work for you?"
```

```
~If the response is affirmative, proceed to confirm contact details~
```

-

- **Behavior:**
    - If the user says "Yes," the AI seamlessly transitions to verifying the user's phone number without explicitly stating its intent.
- 

## 2. Managing Objections

### Use Case: Overcoming Hesitation

#### Prompt:

plaintext

Copy code

```
"I understand you might be unsure, but we've helped many businesses like {{BusinessName}} improve their {{Metric}}."
```

```
~If hesitation is detected, emphasize a success story or testimonial~
```

- - **Behavior:**
    - The agent dynamically shifts to providing social proof to reassure the user.
- 

## 3. Adjusting Tone and Style

### Use Case: Dealing with a Frustrated Caller

#### Prompt:

plaintext

Copy code

```
"I'm here to help and will make this quick for you."
```

```
~Respond with a calm and empathetic tone~
```

- - **Behavior:**
    - The AI adjusts its tone to de-escalate tension and make the caller feel heard.
- 

## 4. Detecting and Responding to Scenarios

### Use Case: Identifying Voicemail

#### Prompt:

plaintext

Copy code

~If voicemail is detected, hang up immediately and note the time for a callback~

- - **Behavior:**
    - The agent recognizes voicemail and exits the interaction without wasting time.
- 

## 5. Managing User Queries

### Use Case: Redirecting Irrelevant Questions

#### Prompt:

plaintext

Copy code

"That's a great question. Our account manager can provide a detailed answer."

~Steer the conversation back to booking the call~

- - **Behavior:**
    - The AI acknowledges the query while tactfully redirecting the user to the primary goal.
- 

## 6. Enhancing Listening Skills

### Use Case: Pausing for User Input

#### Prompt:

plaintext

Copy code

"Does that make sense so far?"

~Pause and wait for user confirmation before continuing~

- - **Behavior:**
    - The AI allows the user time to respond, ensuring they're following along.
- 

## 7. Tailoring Responses

### Use Case: Customizing Pitches

**Prompt:**

plaintext

Copy code

"We've noticed that {{Industry}} businesses in {{Location}} have benefited greatly from this."

~Highlight {{SpecificBenefit}} if user shows interest~

- - **Behavior:**
    - The AI dynamically tailors the pitch based on the user's interest or reaction.
- 

## 8. Error Recovery

### Use Case: Handling Unexpected Responses

**Prompt:**

plaintext

Copy code

~If the response is unclear, politely ask for clarification~

"Could you repeat that for me, just to make sure I got it right?"

- - **Behavior:**
    - The agent gracefully handles ambiguous answers and keeps the conversation on track.
- 

## 9. Pre-Call Preparation

### Use Case: Gathering Data Before Engagement

**Prompt:**

plaintext

Copy code

~Verify {{BusinessName}}, {{OwnerName}}, and {{ScheduledTime}} before starting the call~

- - **Behavior:**
    - The AI ensures it has the correct details before initiating the conversation.
-

## 10. Handling Repeated Scenarios

### Use Case: Closing the Call

#### Prompt:

plaintext

Copy code

```
"Thanks for your time, {{OwnerName}}. I'll send over the confirmation details."
```

```
~End the call once the appointment is booked~
```

- - **Behavior:**
    - The AI smoothly concludes the interaction after achieving the objective.
- 

## 11. Multi-Stage Conversations

### Use Case: Navigating Conditional Branches

#### Prompt:

plaintext

Copy code

```
"Can I speak with the owner?"
```

```
~If the person denies being the owner, proceed to ask for a callback time~
```

- - **Behavior:**
    - The AI follows a specific branch based on the user's response.
- 

## 12. Automating Tool Integration

### Use Case: Triggering External Actions

#### Prompt:

plaintext

Copy code

```
~Log this interaction in the CRM with the outcome: {{CallOutcome}}~
```

- 
- **Behavior:**
  - The AI integrates with a CRM system to automatically record the call's result.

---

## Why This Works

1. **Flexibility:** The ~ instructions provide dynamic guidance without being rigid or overly prescriptive.
2. **Natural Conversations:** Keeps the agent's spoken output clear and relevant while embedding complex logic.
3. **Efficiency:** Reduces redundant scripting by embedding operational logic directly into the prompt.
4. **Control:** Provides granular control over the agent's behavior, ensuring consistency and professionalism.

---

Would you like to dive deeper into one of these examples or explore how this could be implemented in your future prompts?

**You said:**

So in some of the examples that you gave, like number 11, you could also use the condition blocks there. How would you choose which prompting method to use in a situation like that?

**ChatGPT said:**

ChatGPT

You're absolutely right—there are scenarios where you could use either a ~ instruction or a condition block (or even combine them). Choosing which method to use depends on several factors, including the complexity of the scenario, the level of branching required, and the need for conversational flexibility. Here's how you might decide between the two (or use both effectively):

---

## When to Use Condition Blocks

Condition blocks are best suited for scenarios where:

1. **Clear Branching is Required:**
  - If the user's response dictates a clear and distinct path, a condition block provides structure.

Example:

plaintext

Copy code

- [2.1 If R = Doesn't deny they're the owner] -> Proceed to introducing yourself.
- [2.2 If R = Denies they're the owner] -> Politely ask when the owner will be available.

○

## 2. Complex Decision Trees Exist:

- If there are multiple possible responses, condition blocks ensure every branch is accounted for.

Example:

plaintext

Copy code

- [2.2.1 If R = Says they'll transfer the call] -> Wait for the new person to answer.
- [2.2.2 If R = Says the owner is unavailable] -> Ask for a callback time.

○

## 3. The Interaction is Highly Dynamic:

- When the AI needs to adjust based on nuanced user inputs, conditions make it easier to define precise rules.

---

## When to Use ~ Instructions

The ~ symbol is ideal for:

### 1. Simple Behavioral Cues:

- If the AI just needs to perform an action or adjust its tone without branching.

Example:

plaintext

Copy code

~Pause for 2 seconds to allow the user to respond~

○

### 2. Non-Verbal Actions:

- When the instruction doesn't involve speaking or explicitly following a new conversational branch.

Example:

plaintext

Copy code

```
~End the call if voicemail is detected~
```

○

### 3. Enhancing Conversational Flow:

- To ensure smooth transitions without cluttering the prompt with detailed branching.

Example:

plaintext

Copy code

```
~If hesitation is detected, reinforce the key benefit~
```

○

---

## When to Combine Condition Blocks and ~ Instructions

In many cases, combining both can provide the best of both worlds:

### 1. Using Conditions for Branching and ~ for Subtle Adjustments:

Example:

plaintext

Copy code

```
- [2.1 If R = Doesn't deny they're the owner] -> Proceed to  
introducing yourself.
```

```
  ~Respond confidently and with enthusiasm~
```

```
- [2.2 If R = Denies they're the owner] -> Politely ask when the owner  
will be available.
```

```
  ~Speak in a neutral tone~
```

○

- **Why It Works:** The condition defines the branch, and the ~ guides tone or subtle behavior adjustments.

### 2. Using ~ for Inline Instructions Inside Conditions:

Example:

plaintext

Copy code

```
- [2.2 If R = Denies they're the owner] -> Politely ask when the owner  
will be available.
```

"No problem, could you let me know when {{OwnerName}} might be available?"

~Log this interaction in the CRM with the outcome: Owner Unavailable~

- - **Why It Works:** The ~ instruction integrates additional actions without complicating the main branching logic.
- 

## How to Choose Between the Two

Ask yourself these questions:

1. **Is There a Clear Fork in the Road?**
    - **Yes:** Use a condition block.
    - **No:** Use a ~ instruction to refine behavior.
  2. **Does the Instruction Involve a Non-Verbal Action?**
    - **Yes:** Use a ~ instruction.
  3. **Is the Scenario Complex with Multiple Sub-Conditions?**
    - **Yes:** Use condition blocks for clarity.
  4. **Is the Behavior Adjustment Contextual or Subtle?**
    - **Yes:** Use a ~ instruction to tweak tone, pacing, or other nuances.
  5. **Do You Want a Modular, Reusable Structure?**
    - **Yes:** Use condition blocks for overarching logic and ~ for contextual nuances.
- 

## Example of Both in Practice

**Scenario: Confirming if the Caller is the Owner**

**With Both Methods:**

plaintext

Copy code

- [2.1 If R = Doesn't deny they're the owner] ->  
"Great! My name is {{AgentName}}, and I'm calling about {{BusinessName}}'s opportunity to {{SpecificBenefit}}."  
~Respond with confidence and enthusiasm~
- [2.2 If R = Denies they're the owner] ->

"Oh, no problem. Could you let me know when {{OwnerName}} might be available?"

~Log interaction in CRM as 'Spoke to Non-Owner'~

- - **Why This Works:**
    - The condition block handles branching logic for user responses.
    - The ~ instructions enhance the tone and ensure additional actions like logging are performed.
- 

## Conclusion

Use **condition blocks** when clear decision-making is required and **~ instructions** when guiding tone, flow, or non-verbal actions. Combining both methods can create a nuanced, flexible, and highly adaptive voice agent.

Would you like to see an expanded example or test a scenario where these approaches could be integrated?

### You said:

That makes sense. One last question on the ~ symbol. In a few examples you used the symbol to indicate to the AI to make a note of something e.g. ~Log interaction in CRM as 'Spoke to Non-Owner'~ If I created a tool that was dedicated to logging a piece of information like this or created a property in the Structured Data analysis, would an instruction like that be enough for the AI to note that correctly. For example, if I had a property in the structured data analysis called "Decision Maker" and while on the phone the AI uncovered it wasn't speaking to the owner, would an instruction as simple as "~Log interaction in CRM as 'Spoke to Non-Owner'~" give my structured data property the correct output? If I described my structured data property to output either "true" or "false"

### ChatGPT said:

ChatGPT

Great question! The instruction ~Log interaction in CRM as 'Spoke to Non-Owner'~ would work conceptually for guiding the AI to record or note something, but its effectiveness in populating a specific **structured data property** (like **Decision Maker: true/false**) depends on how the AI is designed to interface with the structured data system.

Let's break it down:

---

## How the AI Interprets Instructions

### 1. Simple Instructions (~):

- ~Log interaction in CRM as 'Spoke to Non-Owner'~ is a general guide for the AI to perform a logging action.
- For structured data properties, the AI must explicitly understand how to map that logging instruction to the property.

### 2. Mapping to Structured Data Properties:

- If your structured data property (`Decision Maker`) is pre-defined to accept a `true/false` output, the AI needs an additional step:
    - **Condition Logic:** Determine whether the response fits the criteria for `Decision Maker: true` or `false`.
    - **Structured Output Mapping:** Translate that condition into the property.
- 

## How to Ensure Correct Outputs

To make the system work seamlessly, consider the following:

### 1. Define the Property Clearly:

Example:

plaintext

Copy code

Structured Data Property:

- Name: `Decision Maker`
- Type: `Boolean (true/false)`
- Definition: Indicates whether the AI spoke to the decision maker (`true`) or someone else (`false`).

○

### 2. Include Clear Mapping Instructions:

- Enhance your ~ instruction to explicitly map to the property.

Example:

plaintext

Copy code

~If R = Denies they're the owner, set `Decision Maker` to `false`~

~If R = Doesn't deny they're the owner, set `Decision Maker` to `true`~

○

### 3. Integrate with Tools or APIs:

If you have a logging tool (e.g., a CRM API), ensure the instruction specifies the property and value:

plaintext

Copy code

```
~Log to CRM: Decision Maker = false~
```

- - This ensures a direct mapping between the AI's output and the structured data system.
4. **Test and Validate:**
- Simulate scenarios to verify the property updates correctly based on the AI's logic.
  - Example:
    - User says: "I'm not the owner."
    - AI updates: `Decision Maker: false`.
- 

## Simplified Example in a Prompt

**Scenario: Confirming if the Caller is the Decision Maker**

**Prompt:**

plaintext

Copy code

```
"Can I speak with the owner?"
```

```
~If R = Denies they're the owner, set Decision Maker to false~
```

```
~If R = Doesn't deny they're the owner, set Decision Maker to true~
```

- - **Structured Data Property Update:**
    - If the user denies being the owner: `Decision Maker: false`.
    - If the user doesn't deny being the owner: `Decision Maker: true`.
- 

## Advanced Example with a Logging Tool

**Tool-Integrated Instruction:**

If your AI uses a dedicated logging tool:

plaintext

Copy code

```
"Can I speak with the owner?"
```

~If R = Denies they're the owner, log Decision Maker: false to CRM~  
~If R = Doesn't deny they're the owner, log Decision Maker: true to CRM~

---

## Best Practices

1. **Explicit Mapping:** Clearly define how the instruction relates to structured data properties.
  2. **Condition-Based Updates:** Use conditions to set the correct property value dynamically.
  3. **Test Integration:** Ensure the logging tool or structured data system interprets the AI's output as intended.
- 

Would you like help designing a detailed workflow for structured data integration or testing examples with specific tools?

### You said:

Okay that makes sense. What about ## Objection Handling, I sort of understand how this is structured but if you could explain it in a little bit more detail that would be appreciated. Example: ## Objection Handling - [ If O = "Not interested." ] -> \*\*Q:\*\* "No worries, I hope I got this across correctly. There's an opportunity to {{CoreBenefit}}, just like several {{Industry}} in {{Location}} already have. It'll be a super quick call with my account manager. <opening\_time> is good, yeah." - [ If O = "We're ok with what we have." ] -> \*\*Q:\*\* "I hear you, here's what I can tell you... we've helped many other businesses like {{BusinessName}} in {{Location}} {{SuccessMetric}}. I'll have my account manager call you back, they can help you understand the program and then you can make an informed decision of whether you want to {{CoreBenefit}}. You'll be open at <opening\_time>, right." \_\_ I understand that the part that says [If O = "We're ok with what we have." ] -> is basically saying that if the user says X, then say Y Why do you need the squared brackets and the different symbols like " -> " and the " \*\*Q:\*\* "

### ChatGPT said:

ChatGPT

The structure of **Objection Handling** in the example is carefully designed to provide clarity and organization for both the AI's decision-making and response-generation processes. Let me explain the purpose of each component and why symbols like [ ], ->, and \*\*Q:\*\* are used:

---

## Breakdown of the Structure

## 1. The Brackets [ ]

- **Purpose:** Encapsulates the condition or trigger that determines when this specific objection-handling rule applies.

### Example:

plaintext

Copy code

```
[ If 0 = "Not interested." ]
```

- - **What It Does:** Specifies that this rule is relevant only if the user's objection matches the exact condition (e.g., "Not interested").
- **Why It's Needed:**
  - Clearly separates conditions from actions.
  - Makes it easier to organize and navigate multiple objection-handling rules.

## 2. The Arrow ->

- **Purpose:** Indicates the flow from the condition (user's objection) to the corresponding response (AI's action).

### Example:

plaintext

Copy code

```
[ If 0 = "Not interested." ] -> **Q:** "No worries..."
```

- - **What It Does:** Links the trigger (user's statement) to the AI's response, showing the conversational flow.
- **Why It's Needed:**
  - Enhances readability by explicitly showing the cause-and-effect relationship.
  - Provides a clear visual separation between the condition and the response.

## 3. The Prefix \*\*Q:\*\*

- **Purpose:** Denotes the AI's response, distinguishing it from the condition or other elements.

### Example:

plaintext

Copy code

```
**Q:** "No worries, I hope I got this across correctly..."
```

-

- **What It Does:** Marks the text as the AI's direct verbal response to the objection.
  - **Why It's Needed:**
    - Clearly identifies the AI's part of the conversation.
    - Ensures that the response format is consistent across all objection-handling cases.
- 

## Symbol Usage in Context

### Full Example:

plaintext

Copy code

```
- [ If 0 = "We're ok with what we have." ] -> **Q:** "I hear you, here's what I can tell you... we've helped many other businesses like {{BusinessName}} in {{Location}} {{SuccessMetric}}. I'll have my account manager call you back, they can help you understand the program and then you can make an informed decision of whether you want to {{CoreBenefit}}. You'll be open at <opening_time>, right."
```

### How It Works Step-by-Step:

1. **Condition:** [ If 0 = "We're ok with what we have." ]
    - Specifies the trigger condition: The user's response includes "We're ok with what we have."
  2. **Transition (->):**
    - Connects the condition to the action, showing the logical flow.
  3. **Action/Response (\*\*Q:\*\*):**
    - The AI delivers a response specifically crafted for this objection.
- 

## Why This Structure Works

1. **Clarity and Modularity:**
  - Each objection-response pair is isolated and easy to identify.
  - This modularity allows you to add, remove, or modify individual rules without affecting others.
2. **Readable and Intuitive:**
  - Symbols like [ ], ->, and \*\*Q:\*\* create a consistent, logical structure that's easy to understand.

- Even without prior experience, the format is self-explanatory once you see how it operates.

### 3. **Dynamic and Flexible:**

The bracketed conditions can accommodate complex logic:

plaintext

Copy code

```
[ If 0 = "Not interested" OR "I'm busy" ] -> **Q:** "I hear you, but I promise this will be quick..."
```

- 
- Conditions can be expanded for broader applicability.

### 4. **Action-Oriented:**

- The -> symbol keeps the focus on actionable outcomes by directly linking conditions to responses.

### 5. **Separates Dialogue from Logic:**

- Encapsulates the AI's logic in brackets and arrows while reserving \*\*Q:\*\* for spoken output.

---

## When to Use This Structure

This approach works best when:

1. **There Are Multiple Potential Objections:**
  - E.g., "We already have a solution," "Not interested," "Call back later."
2. **Responses Are Customized:**
  - Each objection requires a tailored response that aligns with the AI's purpose and tone.
3. **Flexibility is Needed:**
  - The system allows easy updates or additions as new objections arise.

---

## Improving the Format

You could further enhance this structure by:

1. **Adding Conditions for Tone or Priority:**

Example:

plaintext

Copy code

```
- [ If 0 = "Not interested" AND Tone = "Neutral" ] -> **Q:** "I get that. Would a quick call at <opening_time> work better?"  
- [ If 0 = "Not interested" AND Tone = "Frustrated" ] -> **Q:** "I hear you, I won't take much of your time—just a quick question..."
```

○

## 2. Incorporating Non-Verbal Actions:

Example:

plaintext

Copy code

```
- [ If 0 = "Not interested." ] ->  
~Log this objection in the CRM~  
**Q:** "No worries, I hope I got this across correctly..."
```

○

---

Would you like me to help you write a few objection-handling rules for a specific use case, or explore advanced ways to structure this logic?

**You said:**

This is extremely valuable insights, thank you so much! I'm very happy with what I have learnt today. Without going into too much detail, are there any other prompting techniques that you feel are beneficial when creating voice agent system prompts?

**ChatGPT said:**

ChatGPT

I'm so glad you found this helpful! While we've covered a lot of ground with condition blocks, ~ instructions, variable blocks, and objection handling, there are a few additional prompting techniques that can elevate voice agent system prompts. These techniques are focused on improving **naturalness**, **adaptability**, and **user engagement**:

---

## 1. Context Persistence

- **What It Is:** Ensuring the AI remembers relevant details shared earlier in the conversation.
- **Why It's Useful:** Maintains continuity and avoids redundant questions.

**Example:**

plaintext

Copy code

```
"Just to confirm, you mentioned your business operates from  
{{OpeningTime}} to {{ClosingTime}}, right?"
```

```
~Refer to earlier data to avoid asking again~
```

- 
- 

## 2. Progressive Disclosure

- **What It Is:** Sharing information in manageable chunks to avoid overwhelming the user.
- **Why It's Useful:** Keeps conversations concise and engaging while maintaining clarity.

**Example:**

plaintext

Copy code

```
"We help businesses like {{BusinessName}} improve {{Metric}}. Would  
you like me to explain how?"
```

- 
- 

## 3. Tone and Emotion Control

- **What It Is:** Adjusting the agent's tone dynamically based on user input or sentiment.
- **Why It's Useful:** Creates a more empathetic and human-like interaction.

**Example:**

plaintext

Copy code

```
~If sentiment = "Frustrated," respond with a slower pace and  
apologetic tone~
```

```
"I completely understand, and I'll make this quick for you."
```

- 
- 

## 4. Intent Clarification

- **What It Is:** Asking targeted questions to ensure the agent correctly understands the user's needs or responses.

- **Why It's Useful:** Reduces misunderstandings and guides the conversation effectively.

**Example:**

plaintext

Copy code

"Just to clarify, are you saying you'd prefer a callback tomorrow morning?"

- 
- 

## 5. Multi-Step Tasks

- **What It Is:** Breaking down complex tasks into smaller, manageable steps.
- **Why It's Useful:** Helps the AI guide users through intricate processes without confusion.

**Example:**

plaintext

Copy code

"Let's book a call. First, what time works best for you tomorrow?"  
~After receiving time, confirm details and move to the next step~

- 
- 

## 6. Fallback Mechanisms

- **What It Is:** Establishing clear fallback actions for when the AI doesn't understand or the user provides ambiguous input.
- **Why It's Useful:** Prevents dead-ends and keeps the conversation productive.

**Example:**

plaintext

Copy code

~If response is unclear, rephrase the question with additional context~

"I didn't quite catch that—do you mean {{OptionA}} or {{OptionB}}?"

- 
- 

## 7. Goal-Reinforcement Statements

- **What It Is:** Reiterating the purpose of the interaction periodically to keep the user engaged.
- **Why It's Useful:** Aligns the conversation with the end goal and reduces distractions.

**Example:**

plaintext

Copy code

```
"The main reason I'm calling is to ensure {{BusinessName}} doesn't miss out on {{CoreBenefit}}."
```

- 
- 

## 8. Personalized Summaries

- **What It Is:** Summarizing the conversation with user-specific details.
- **Why It's Useful:** Reinforces the value of the interaction and ensures mutual understanding.

**Example:**

plaintext

Copy code

```
"Great, so I've scheduled a call for {{ScheduledTime}} to discuss how we can help {{BusinessName}} improve {{Metric}}."
```

- 
- 

## 9. Adaptive Question Framing

- **What It Is:** Framing questions based on the user's prior responses or tone.
- **Why It's Useful:** Increases relevance and encourages engagement.

**Example:**

plaintext

Copy code

```
~If user expressed skepticism earlier, frame the next question to address their doubts~
```

```
"Would it help if I explained how {{SimilarBusiness}} benefited from this?"
```

-

---

## 10. Closing with a Call-to-Action

- **What It Is:** Ending every interaction with a clear and actionable next step.
- **Why It's Useful:** Ensures the conversation moves toward the desired outcome.

### Example:

plaintext

Copy code

```
"Perfect, I'll text you a confirmation. You'll hear from our account manager at {{ScheduledTime}}!"
```

- 

---

## Combining Techniques

These techniques can and should be combined to create robust and natural voice agent prompts. For instance:

- Use **progressive disclosure** to keep users engaged while employing **context persistence** to maintain continuity.
- Pair **intent clarification** with **fallback mechanisms** to navigate ambiguity gracefully.

---

### You said:

Amazing thank you

### ChatGPT said:

ChatGPT

You're very welcome! I'm glad I could help. Feel free to reach out whenever you're ready to dive deeper or start crafting new prompts. Wishing you all the best with your project—it's shaping up to be something fantastic! 🚀 😊