

SRE and AI

Feature Description

Table of Contents

1 Introduction 1

2 Text-to-Speech (TTS) 1

3 Speech-to-Text (STT) 2

4 LLM Nodes in the Call Flow 2

5 Real-Time Voice AI with OpenAI Realtime over SIP 3

6 Real-Time vs Chained Processing 4

1 Introduction

SRE (Service Routing Engine) brings AI into live calls through two parts of the platform:

- **The media processor** drives an Asterisk media server. It plays prompts, records the caller and handles DTMF.
- **The service-logic engine** runs graphical call flows built from nodes. A node can call any HTTP API, open WebSockets and route calls.

AI can be used in two ways: speech and language services chained step by step inside a call flow, or a real-time voice agent such as OpenAI Realtime that talks to the caller directly while SRE keeps control of the call.

2 Text-to-Speech (TTS)

The **Play text-to-speech** node turns any text into audio and plays it on the call.

- **Provider:** Google Cloud Text-to-Speech, configured with a service-account credentials file.
- **Dynamic content:** the text and the language can come from variables in the call flow. For example, a prompt can read back an account balance or an LLM's answer.
- **SSML:** text wrapped in `<speak>...</speak>` is sent as SSML, which controls pauses, emphasis and pronunciation.
- **Voice settings:** language tag (e.g. `en-GB`, `fr-FR`, `nl-BE`) and voice gender (male, female or neutral).
- **Direction:** the audio can play to the caller or to the called party.

- **Caching:** a prompt that was already generated is reused instead of calling the cloud provider again, which saves cost and time.

3 Speech-to-Text (STT)

The **Speech to text** node records the caller and transcribes what they said.

- **Provider:** Google Cloud Speech-to-Text, configured with a service-account credentials file.
- **Recording controls:**
  - maximum duration
  - maximum silence (end of speech)
  - optional beep at the start
  - optional DTMF key to stop recording
- **Language:** set per node, and it can come from a variable, e.g. the language the caller picked earlier.
- **Result:** when the transcription finishes, the call flow gets an `stt_finished` event with the text in a variable. The flow can then branch on it, send it to an LLM or store it.

4 LLM Nodes in the Call Flow

The **AI language model completion** node sends a prompt to a large language model and stores the answer in a variable. It works in any call flow: voice calls, HTTP/REST requests, SIP routing and WebSocket events.

Supported providers:

Provider	How it connects
OpenAI	Native OpenAI SDK
Azure OpenAI	Native SDK; requires the resource endpoint, API version and deployment name
Anthropic (Claude)	Native Anthropic SDK
Google Gemini	Native Google GenAI SDK
Mistral AI	OpenAI-compatible API
DeepSeek	OpenAI-compatible API

Provider	How it connects
OpenRouter	OpenAI-compatible gateway to many models
Ollama (self-hosted)	OpenAI-compatible API on your own infrastructure, so no data leaves the network

The API base URL can also be overridden, which lets the node use any other OpenAI-compatible endpoint, such as a private gateway or a self-hosted model.

Prompting:

- A system prompt, plus any number of additional user or assistant messages.
- All prompts are templates, so they can include call data such as the calling number, an STT transcript or a CRM lookup result.

Settings: model name, temperature, max tokens and timeout.

Reliability and cost:

- The node has separate success and failure exits, so the call can fall back to a default route if the LLM is slow or unavailable.
- Responses can be cached, so identical prompts are not billed twice.

Typical uses:

- classifying a caller's intent from a transcript
- extracting a structured value (account number, reason for calling) as JSON for the flow to parse
- generating a reply to read back with TTS
- summarising a call for a ticket or notification

5 Real-Time Voice AI with OpenAI Realtime over SIP

OpenAI's Realtime API offers a SIP endpoint. A call can be connected straight to a speech-to-speech model that listens, reasons and answers with very low delay. SRE sets up this connection and stays in control of the call while the AI talks. The integration is built from standard SRE nodes:

1. **Route the call to OpenAI.** SRE routes the SIP call to the OpenAI SIP endpoint for your project, the same way it routes any other call.
2. **Receive and verify OpenAI's webhook.** OpenAI notifies SRE of the incoming call with a webhook. The SRE HTTP interface receives it, and the **Validate webhook signature** node checks the

signature (Standard Webhooks, which OpenAI uses) so that only genuine OpenAI requests are accepted.

3. **Accept and configure the session.** An HTTP request node calls OpenAI to accept the call. It sets the instructions, voice and tools, and can personalise them with data SRE already holds about the caller.
4. **Stream events back to SRE.** The **Connect WebSocket** node opens a WebSocket to the call's Realtime session through the SRE WebSocket processor.
    - Each event from OpenAI triggers the execution of a service logic. Events include transcripts, function/tool calls and end of response.
    - The event arrives as parsed JSON, together with the call identifier and any variables passed when the connection was opened.
5. **Act on the call.** The service logic can do anything a normal service logic can do:
    - Answer tool calls by looking up the internal database, a CRM or any API, and send the result back.
    - Change the session while it runs, for example its instructions.
    - Transfer the call to a human agent or another number, or hang up, when the AI or a business rule decides.
    - Close the session.

The result: OpenAI holds the conversation, and SRE supplies the business data, the routing decisions, the logging and the controls around it.

6 Real-Time vs Chained Processing

SRE supports both models, and they suit different needs.

Chained processing (step by step). Each step is a separate node, and the service logic runs them in order. For example:

1. Play a TTS greeting.
2. **Speech to text** captures what the caller says.
3. **AI completion** works out the intent, e.g. "billing question, account 12345".
4. The flow branches on the result: look up the account through a REST call, then read the answer back with TTS.
5. **Route** to the right queue or number, or hang up.

Real-time interaction (speech-to-speech). The caller talks directly to the OpenAI Realtime model over SIP. Events stream to SRE, which reacts to them.

	Chained (STT - LLM - TTS - route)	Real-time
Conversation	Turn by turn: the caller speaks, the system processes, then answers	Natural, fluid conversation; the caller can interrupt the AI
Delay	Several seconds per turn (recording, transcription, LLM, audio generation)	Sub-second answers
Control	Full and predictable: every step is visible and testable in the service logic	The model leads the conversation; SRE steps in through events and tool calls
Choice of vendor	Each piece chosen separately: Google STT/TTS plus any of the supported LLM providers, including self-hosted Ollama	Tied to OpenAI's Realtime models
Data privacy	Can be kept on-premises with self-hosted models	Audio is sent to OpenAI
Cost	Lower; billed per request, and TTS and LLM answers are cached	Higher; billed per minute of audio
Best for	Structured IVRs, intent-based routing, collecting a few data points, regulated environments	Virtual agents, open-ended customer service, natural conversation

The two can be combined. For example, a chained flow can qualify the caller first and then hand the call to a real-time agent. Or a real-time agent can call back into SRE through tools to look up data or route the call.