SharyX Voice Agent

Technical Documentation

v1.0.0 • Production Ready • MIT Licensed

1. Overview

SharyX Voice Agent is a modular ecosystem for building ultra-low-latency AI voice assistants. It enables real-time communication across both web and telephony platforms, and is designed with scalability and flexibility as core principles.

The SDK provides developers with the tools needed to build intelligent voice applications quickly, supporting both browser-based and phone-based interactions — making it suitable for modern AI-driven communication systems.

Package	sharyx-voice-agent (npm)
License	MIT — open source, free for commercial use
Repository	github.com/sharyx-repo/sharyx-voice-agent
Status	v1.0.0 — Production Ready

2. Architecture Overview

The system follows a clean and extensible 3-layer architecture. Each layer has a clearly defined responsibility; this separation of concerns improves maintainability and allows independent development and scaling of components.

Layer	Component	Responsibility
1 — Adapters	Transport Layer	Handles input/output — web browser, telephony, simulation
2 — Pipeline	Core Engine	Orchestrates STT → LLM → TTS processing flow

3 — Providers	AI Services	Pluggable AI backends: OpenAI, Anthropic, Deepgram, etc.

3. Core Engine — VoiceAgent & Pipeline

The core engine is the heart of the SharyX Voice Agent SDK. It is responsible for managing real-time voice interactions by orchestrating audio input, AI reasoning, and audio output. It consists of two main components: VoiceAgent and Pipeline.

3.1 VoiceAgent Class

The VoiceAgent class manages adapters and controls the session lifecycle. It registers adapters, starts the server, and delegates processing to the pipeline. It acts as the central controller of the system.

- Registers and manages transport adapters
- Controls session start, pause, and teardown lifecycle
- Delegates audio processing to the Pipeline
- Exposes event hooks for monitoring and customization

3.2 Pipeline Class

The Pipeline is responsible for handling the full voice interaction flow. It processes audio input, converts speech to text, generates AI responses, and converts them back to audio — ensuring real-time streaming and low-latency communication.

Data flow through the Pipeline:

```
Audio Input  →  STT (Speech-to-Text)
           →  LLM (AI Reasoning & Response)
           →  TTS (Text-to-Speech)
           →  Audio Output
```

4. Code Examples

4.1 Agent Setup

Create an agent with your preferred STT provider and system prompt:

```
import { createAgent } from 'sharyx-voice-agent';

const agent = createAgent({
  apiKey: process.env.OPENAI_API_KEY,
  stt: { provider: 'deepgram', apiKey: 'YOUR_KEY' },
  systemPrompt: 'You are a helpful assistant.'
});
```

4.2 Pipeline Flow (Simplified)

The pipeline listens for audio events and processes them end-to-end:

```
// Audio → STT → LLM → TTS → Audio

transport.on('audio', (data) => {
  sttStream.send(audioBuffer);
});

sttStream.on('transcript', async (text) => {
  const response = await llm.generate(text);
  const audio    = await tts.speak(response);
  transport.sendAudio(audio);
});
```

5. Installation & Quick Start

5.1 Install via npm

```
npm install sharyx-voice-agent
```

5.2 Scaffold a New Project with CLI

Use the scaffolding CLI to generate a ready-to-use project structure instantly:

```
npx create-sharyx-agent my-voice-app
```

The CLI generates a pre-configured folder with templates including TypeScript, JavaScript, and sample voice workflows.

5.3 Development Workflow

- Create your project using the CLI command above
- Navigate into the generated directory
- Install dependencies with npm install
- Run the development server with npm run dev
- Customize adapters, providers, and system prompt
- Deploy to your production environment

6. Feature Reference

Feature	Description
Core SDK	Robust VoiceAgent class handling real-time conversation, streaming, and custom event hooks.
Simulation Mode	Test voice flows instantly without hardware — ideal for rapid prototyping and CI/CD pipelines.
LLM Cloud	Direct integration with OpenAI, Anthropic, or local models. Includes built-in prompt and memory management.
CLI Tooling	Scaffold full project structures in seconds with npx create-sharyx-agent and pre-built templates.
Workflow Engine	Define complex conversational paths and fallbacks. Stream transcripts in real time to any interface.
Monorepo Ready	Clean exports, TypeScript definitions, and MIT Licensed. Built for developers by developers.

7. Package & Repository Structure

7.1 Monorepo Overview

The GitHub repository contains two main packages, both MIT licensed and published to npm.

sharyx-voice-agent	create-sharyx-agent
The engine: VoiceAgent class, createAgent factory, real-time conversation core.	CLI tool: project generator, templates, and quickstart with best practices.

7.2 Local Development & Contribution

```
git clone https://github.com/sharyx-repo/sharyx-voice-agent.git
cd sharyx-voice-agent && npm install && npm run build
cd ./create-sharyx-agent && npm install && npm run build
```

8. Simulation Mode

The SDK includes a full simulation environment to test conversational flows without any hardware — ideal for debugging and integration tests. Enable simulation by passing configuration options at agent startup:

```
agent.enableSimulation({
  latency: 200,
  responses: 'mock'
});
```

- No hardware or microphone required
- Configurable mock latency to simulate real-world conditions
- Supports custom mock response payloads
- Compatible with CI/CD pipelines and automated testing

sharyx-voice-agent — MIT Licensed

npm package • GitHub repository • Full documentation