Getting Started

Install

```
npm install @overshoot/sdk
```

Get an API Key

Get your API key at Overshoot Platform.

Basic Usage

```
import { RealtimeVision } from '@overshoot/s

const vision = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api
  apiKey: 'your-api-key',
  prompt: 'Read any visible text',
  onResult: (result) => {
    console.log(result.result)
  }
})

await vision.start()
await vision.stop()
```

Using a Video File

On This Page

```
const vision = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api
  apiKey: 'your-api-key',
  prompt: 'Describe what you see',
  source: { type: 'video', file: videoFile }
  onResult: (result) => {
    console.log(result.result)
  }
})
```

What can we improve?

Results arrive continuously as the video plays. Each result describes what happened in a short window of video.

Last updated on January 17, 2026

Video Input Source

The Overshoot SDK supports two video inputs:

- **Camera (default)**: uses the camera of the device running the webapp.
- **Video file**: plays a video file in the browser and streams its frames.

If you don't set anything, the SDK uses the back camera when available (`cameraFacing: 'environment'`).

```
// Use the camera (back camera on mobile devices)
const vision = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api
  apiKey: 'your-api-key',
  prompt: 'Read any visible text',
  source: { type: 'camera', cameraFacing: 'u
})

// Use a video file instead
const visionFromFile = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api
  apiKey: 'your-api-key',
  prompt: 'Describe what you see',
  source: { type: 'video', file: videoFile }
})
```

When using the camera, the browser will ask the user for permission the first time. The SDK handles the rest (creating a stream, wiring it to the AI, and sending frames).

Last updated on January 17, 2026

Overshoot Documentation

Configuration

Most apps only need a few options:

- `apiUrl` : where to send requests (usually `https://cluster1.overshoot.ai/api/v0.2`)
- `apiKey` : your secret key from Overshoot
- `prompt` : what you want the AI to do
- `source` : where the video comes from (`camera` by default, or a `video` file)

```
const vision = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api/v0.2',
  apiKey: 'your-api-key',
  prompt: 'Read any visible text',
  source: { type: 'camera', cameraFacing: 'environment' }
})
```

Updating the Prompt

You can change the prompt while the stream is running. This is useful when you want to ask different questions about the video without restarting.

```
// Change what the AI looks for mid-stream
vision.updatePrompt('Count the number of people')
```

The next result will use the new prompt.

Processing Parameters

These control how much of the video gets analyzed. Most apps can skip this section and use the defaults.

- `clip_length_seconds` : how long each window is (default: 1

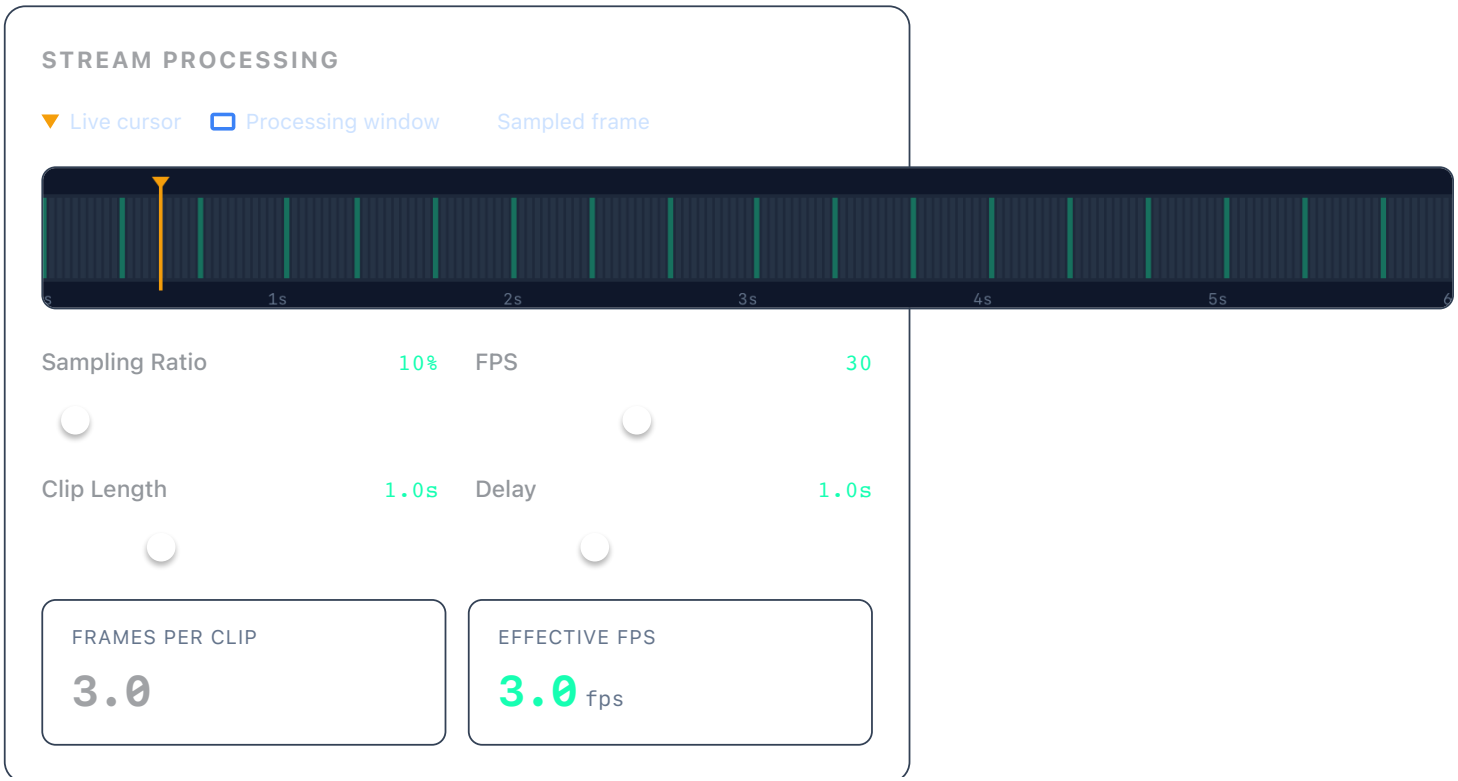
second). Longer windows give the AI more context but take longer to process.

- `delay_seconds` : how often you get a new result (default: 1 second). Smaller delays mean more frequent results.
- `fps` : max frames per second to capture (default: 30)
- `sampling_ratio` : what fraction of frames to actually send to the AI (default: 0.1 = 10%). Lower values are faster and cheaper.

```
const vision = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api/v0.2',
  apiKey: 'your-api-key',
  prompt: 'Read any visible text',
  processing: {
    clip_length_seconds: 1,
    delay_seconds: 1,
    fps: 30,
    sampling_ratio: 0.1
  }
})
```

Processing Visualization

Play with the sliders below to see how these parameters affect frame sampling.



Last updated on January 17, 2026

Overshoot Documentation

Output

Every time the AI finishes analyzing a window of video, it calls your `onResult` callback.

```
onResult: (result) => {
  console.log(result.result)
  console.log(result.inference_latency_ms)
  console.log(result.total_latency_ms)
}
```

Structured Output

By default, `result.result` is plain text. If you want JSON, pass an `outputSchema`:

```
const vision = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api
  apiKey: 'your-api-key',
  prompt: 'Count the people in the frame',
  outputSchema: {
    type: 'object',
    properties: {
      count: { type: 'number' }
    }
  },
  onResult: (result) => {
    const data = JSON.parse(result.result)
    console.log('People count:', data.count)
  }
})
```

Last updated on January 17, 2026

Overshoot Documentation

Models

Overshoot runs vision models optimized for low-latency real-time inference.

Available Models

- `Qwen/Qwen3-VL-30B-A3B-Instruct` (default)
- `Qwen/Qwen3-VL-8B-Instruct`
- `OpenGVLab/InternVL3_5-30B-A3B`

```
const vision = new RealtimeVision({
  apiUrl: 'https://cluster1.overshoot.ai/api
  apiKey: 'your-api-key',
  prompt: 'Read any visible text',
  model: 'Qwen/Qwen3-VL-30B-A3B-Instruct'
})
```

If you don't specify a model, the SDK uses the default.

Last updated on January 17, 2026