Getting Started

API Quick Start

Copy page

Integrate Odyssey-2 Pro into your application in 5 minutes.

Create a developer account & get an API key

Sign up and receive an API key at developer.odyssey.ml.

Install

npm yarn pnpm

```
npm install @odysseyml/odyssey
```

npm package version `"@odysseyml/odyssey": "^1.0.0" required`

Image-to-Video Requirements

For image-to-video generation:

- **Max size**: 25MB
- **Supported formats**: JPEG, PNG, WebP, GIF, BMP, HEIC, HEIF, AVIF
- **Resolution**: Images are automatically resized to 1280×704 (landscape) or 704×1280 (portrait)

```
  <style>
    body { font-family: system-ui; max-width: 600px; margin: 2rem auto; }
    video { width: 100%; background: #000; }
    input[type="text"] { flex: 1; padding: 0.5rem; }
    button { padding: 0.5rem 1rem; }
    .row { display: flex; gap: 0.5rem; margin-top: 0.5rem; }
  </style>
</head>
<body>
  <video id="video" autoplay playsinline muted></video>
  <p id="status">Loading...</p>
  <div class="row">
    <input id="image" type="file" accept="image/*" />
  </div>
  <div class="row">
    <input id="prompt" type="text" placeholder="Enter prompt..." />
    <button id="send">Send</button>
    <button id="end" disabled>End</button>
  </div>
  <script type="module">
    import { Odyssey } from 'https://esm.sh/@odysseyml/odyssey';

    const client = new Odyssey({ apiKey: 'ody_your_api_key_here' });
    const status = document.getElementById('status');
    const prompt = document.getElementById('prompt');
    const imageInput = document.getElementById('image');
    const endBtn = document.getElementById('end');
    let isStreaming = false;

    window.addEventListener('beforeunload', () => client.disconnect());

    status.textContent = 'Connecting...';
    const mediaStream = await client.connect();
    document.getElementById('video').srcObject = mediaStream;
    status.textContent = 'Connected';
```

```
        if (isStreaming) {
          await client.interact({ prompt: text });
        } else {
          const image = imageInput.files[0]; // Get selected image
          status.textContent = 'Starting...';
          await client.startStream({ prompt: text, image });
          isStreaming = true;
          endBtn.disabled = false;
          status.textContent = 'Streaming';
        }
      };

      endBtn.onclick = async () => {
        await client.endStream();
        isStreaming = false;
        endBtn.disabled = true;
        status.textContent = 'Connected';
      };
    </script>
  </body>
  </html>
```

Callback Style

⚠ Always ensure `disconnect()` is called when done (via page unload handlers or component cleanup). Stale connections count towards your concurrent session limit (max 1), which will block new connections until they time out. If `disconnect()` is not called, connections are automatically cleared after 40 seconds on the server side.

JavaScript

```
                    >
```

```
const mediaStream = await client.connect();
document.querySelector('video').srcObject = mediaStream;

// Get image from file input
const imageFile = document.querySelector('input[type="file"]').files[0];

await client.startStream({ prompt: 'A cat', image: imageFile });
await client.interact({ prompt: 'Pet the cat' });
await client.endStream();
client.disconnect();
```

Callback Style

React

```
                    >

const client = new Odyssey({ apiKey: 'ody_your_api_key_here' });

function App() {
  const videoRef = useRef<HTMLVideoElement>(null);
  const [prompt, setPrompt] = useState('');
  const [image, setImage] = useState<File | null>(null);
  const [status, setStatus] = useState('Disconnected');
  const [isConnected, setIsConnected] = useState(false);
  const [isStreaming, setIsStreaming] = useState(false);

  useEffect(() => {
    return () => client.disconnect();
  }, []);

  const handleConnect = async () => {
    setStatus('Connecting...');
    const mediaStream = await client.connect();
    if (videoRef.current) videoRef.current.srcObject = mediaStream;
    setIsConnected(true);
    setStatus('Connected');
  };

  const handleSend = async () => {
    const text = prompt;
    setPrompt('');

    if (isStreaming) {
      await client.interact({ prompt: text });
    } else {
      setStatus('Starting...');
      await client.startStream({ prompt: text, image: image || undefined });
      setIsStreaming(true);
      setStatus('Streaming');
    }
  };
```

```
                >

        <button onClick={handleConnect} disabled={isConnected}>Connect</button>
        <input type="file" accept="image/*" onChange={(e) => setImage(e.target.file
        <input value={prompt} onChange={(e) => setPrompt(e.target.value)} placehold
        <button onClick={handleSend} disabled={!isConnected}>Send</button>
        <button onClick={() => client.endStream()} disabled={!isStreaming}>End</but
      </div>
    );
  }
```

Callback Style

Hook Style (useOdyssey)

Next.js

›

```
const client = new Odyssey({ apiKey: 'ody_your_api_key_here' });

export default function OdysseyDemo() {
  const videoRef = useRef<HTMLVideoElement>(null);
  const [prompt, setPrompt] = useState('');
  const [image, setImage] = useState<File | null>(null);
  const [isConnected, setIsConnected] = useState(false);
  const [isStreaming, setIsStreaming] = useState(false);

  useEffect(() => () => client.disconnect(), []);

  const handleConnect = async () => {
    const mediaStream = await client.connect();
    if (videoRef.current) videoRef.current.srcObject = mediaStream;
    setIsConnected(true);
  };

  const handleSend = async () => {
    const text = prompt;
    setPrompt('');
    if (isStreaming) {
      await client.interact({ prompt: text });
    } else {
      await client.startStream({ prompt: text, image: image || undefined });
      setIsStreaming(true);
    }
  };

  return (
    <div>
      <video ref={videoRef} autoPlay playsInline muted style={{ width: '100%', ba
      <button onClick={handleConnect} disabled={isConnected}>Connect</button>
      <input type="file" accept="image/*" onChange={(e) => setImage(e.target.file
      <input value={prompt} onChange={(e) => setPrompt(e.target.value)} placehold
```

›

Callback Style

Hook Style (useOdyssey)

Next.js App Router requires the `'use client'` directive for components using React hooks and browser APIs like video elements.

Python

```
                ›
```

```python
async def main():
    client = Odyssey(api_key="ody_your_api_key_here")

    try:
        await client.connect(
            on_video_frame=lambda frame: print(f"Frame: {frame.width}x{frame.heig
            on_stream_started=lambda stream_id: print(f"Ready: {stream_id}"),
        )
        # Start with an image file
        await client.start_stream(
            prompt="A cat",
            portrait=False,
            image_path="/path/to/image.jpg"
        )
        await client.interact("Pet the cat")
        await client.end_stream()
    except OdysseyAuthError:
        print("Invalid API key")
    except OdysseyConnectionError as e:
        print(f"Connection failed: {e}")
    finally:
        await client.disconnect()

asyncio.run(main())
```

Next Steps

JavaScript Python

Odyssey-2 Pro Overview

Interaction Tips