Oral Cancer Patient Dialogue System API Reference

API Overview

- **Base URL**: `http://[server-address]:8000`
- **API Version**: v1
- **Content-Type**: `application/json` (unless otherwise specified)
- **Authentication**: None (reserved for future implementation)

API Visualization

To help you better understand the API workflow and system architecture, we provide a series of visualization diagrams.
Please refer to the API Visualization Document for more details.

Endpoints

GET /api/characters

Retrieve all available patient characters.

Request

```
GET /api/characters HTTP/1.1
Host: [server-address]:8000
```

Response

`200 OK`

```
{
  "characters": {
    "1": {
      "name": "Patient 1",
      "age": 55,
      "background": "Oral cancer patient...",
      "persona": "...",
      "goal": "..."
    },
    "2": {
      "name": "Patient 2",
      "age": 60,
      "background": "...",
      "persona": "...",
      "goal": "..."
```

```
    }
  }
}
```

POST /api/dialogue/text

Send text to engage in dialogue.

Request

```
POST /api/dialogue/text HTTP/1.1
Host: [server-address]:8000
Content-Type: application/json

{
  "text": "How are you feeling today?",
  "character_id": "1",
  "session_id": null,
  "response_format": "text"
}
```

Parameter	Type	Required	Description
`text`	string	Yes	Text input from the healthcare provider
`character_id`	string	Yes	Patient character ID
`session_id`	string	No	Existing session ID; if null, a new session will be created
`response_format`	string	No	Response format, options: "text" (default) or "audio"

Usage Scenarios

This endpoint supports the following two usage scenarios:

1. **Text Input, Text Response** (default): Send text, receive text responses in JSON format
2. **Text Input, Audio Response**: Send text, receive audio response in WAV format

Response

When `response_format` is "text" (default):

`200 OK`
Content-Type: application/json

```
{
  "status": "success",
  "responses": [
    "Good morning, doctor. My right cheek wound is still swollen, and there's still yellow discharge.",
    "Doctor, good morning. The swelling hasn't improved, and the discharge is still there.",
    "Morning, doctor. The wound on my cheek doesn't feel good, the swelling and discharge are still present."
  ],
  "state": "NORMAL",
  "dialogue_context": "Doctor's rounds",
  "session_id": "d8334f87-7d85-4d84-8101-6461e230c501"
}
```


When `response_format` is "audio":

`200 OK`
Content-Type: audio/wav
X-Session-ID: [session ID]

[binary audio data]

Error Responses

`400 Bad Request`


```
{
  "detail": "text parameter is required"
}
```


`400 Bad Request`


```
{
  "detail": "character_id is required to create a new session"
}
```

POST /api/dialogue/audio

Engage in dialogue through audio.

Request

```
POST /api/dialogue/audio HTTP/1.1
Host: [server-address]:8000
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="audio_file"; filename="recording.wav"
Content-Type: audio/wav
```

```
[binary audio data]
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="character_id"

1
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="session_id"

d8334f87-7d85-4d84-8101-6461e230c501
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="response_format"

text
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

Parameter	Type	Required	Description
`audio_file`	file	Yes	Audio file (WAV format)
`character_id`	string	Yes	Patient character ID
`session_id`	string	No	Existing session ID; if empty, a new session will be created
`response_format`	string	No	Response format, options: "text" (default) or "audio"

Usage Scenarios

This endpoint supports the following two usage scenarios:

1. **Audio Input, Text Response** (default): Upload audio, receive text responses in JSON format
2. **Audio Input, Audio Response**: Upload audio, receive audio response in WAV format

For scenario 2 (Audio Input, Audio Response), add the `response_format=audio` parameter to your request:

```
POST /api/dialogue/audio HTTP/1.1
Host: [server-address]:8000
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="audio_file"; filename="recording.wav"
Content-Type: audio/wav

[binary audio data]
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="character_id"

1
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="session_id"

d8334f87-7d85-4d84-8101-6461e230c501
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="response_format"
```

```
audio
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

This will return an audio file (WAV format) as the response, instead of JSON.

Response

When `response_format` is "text" (default):

`200 OK`
Content-Type: application/json

```
{
  "status": "success",
  "responses": [
    "Good morning, doctor. My right cheek wound is still swollen, and there's still yellow discharge."
  ],
  "state": "NORMAL",
  "dialogue_context": "Doctor's rounds",
  "session_id": "d8334f87-7d85-4d84-8101-6461e230c501"
}
```

When `response_format` is "audio":

`200 OK`
Content-Type: audio/wav
X-Session-ID: [session ID]

[binary audio data]

Error Responses

`400 Bad Request`

```
{
  "detail": "Audio processing failed: [error details]"
}
```

GET /api/health

Check API service health status.

Request

```
GET /api/health HTTP/1.1
Host: [server-address]:8000
```

Response

`200 OK`

```
{
  "status": "ok",
  "active_sessions": 5
}
```

Status Codes

Status Code	Description
200	Success
400	Bad Request - Invalid parameters
404	Not Found - The requested resource was not found
500	Internal Server Error

Dialogue States

The `state` field in the dialogue response may have the following values:

State	Description
`NORMAL`	Normal dialogue state
`CONFUSED`	Patient feels confused
`ANXIOUS`	Patient feels anxious
`ANGRY`	Patient feels angry

Input/Output Combinations

The following table summarizes the available input and output combinations:

Input Type	Output Type	API Endpoint	Parameter Settings
Text	Text (JSON)	`/api/dialogue/text`	`response_format="text"` (default)
Text	Audio (WAV)	`/api/dialogue/text`	`response_format="audio"`
Audio	Text (JSON)	`/api/dialogue/audio`	`response_format="text"` (default)
Audio	Audio (WAV)	`/api/dialogue/audio`	`response_format="audio"`

Client Code Examples

Python

```python
import requests
import json

# Text dialogue (getting text response)
def text_dialogue(text, character_id, session_id=None):
    url = "http://localhost:8000/api/dialogue/text"
    data = {
        "text": text,
        "character_id": character_id,
        "session_id": session_id,
        "response_format": "text"  # explicitly specify text response
    }
    response = requests.post(url, json=data)
    return response.json()

# Text dialogue (getting audio response)
def text_dialogue_with_audio(text, character_id, session_id=None):
    url = "http://localhost:8000/api/dialogue/text"
    data = {
        "text": text,
        "character_id": character_id,
        "session_id": session_id,
        "response_format": "audio"  # request audio response
    }
    response = requests.post(url, json=data)

    # Check response type
    if response.headers.get('Content-Type') == 'audio/wav':
        # Get session ID from header
        session_id = response.headers.get('X-Session-ID')

        # Save audio
        audio_filename = f"response_{hash(text)}.wav"
        with open(audio_filename, "wb") as f:
            f.write(response.content)

        return {
            "status": "success",
            "audio_file": audio_filename,
            "session_id": session_id
        }
    else:
        # If not audio response, might be an error
        return response.json()

# Audio dialogue (getting text or audio response)
def audio_dialogue(audio_file_path, character_id, session_id=None, response_format="text"):
    url = "http://localhost:8000/api/dialogue/audio"
```

```
    with open(audio_file_path, "rb") as audio_file:
        files = {"audio_file": audio_file}
        data = {
            "character_id": character_id,
            "response_format": response_format
        }

        if session_id:
            data["session_id"] = session_id

        response = requests.post(url, files=files, data=data)

        # If requesting audio response
        if response_format == "audio" and response.headers.get('Content-Type') ==
'audio/wav':
            # Get session ID from header
            session_id = response.headers.get('X-Session-ID')

            # Save audio
            audio_filename = f"response_audio_{hash(audio_file_path)}.wav"
            with open(audio_filename, "wb") as f:
                f.write(response.content)

            return {
                "status": "success",
                "audio_file": audio_filename,
                "session_id": session_id
            }
        else:
            # Text response or error
            return response.json()

# Usage example: Text input, text response
text_result = text_dialogue("How are you feeling today?", "1")
session_id = text_result["session_id"]
print(f"Patient response: {text_result['responses'][0]}")

# Usage example: Text input, audio response
audio_response = text_dialogue_with_audio("Tell me about your wound", "1", session_id)
if "audio_file" in audio_response:
    print(f"Audio response saved to: {audio_response['audio_file']}")
    session_id = audio_response["session_id"]

# Usage example: Audio input, text response
audio_text_result = audio_dialogue("recording.wav", "1", session_id)
print(f"Patient response: {audio_text_result['responses'][0]}")

# Usage example: Audio input, audio response
audio_audio_result = audio_dialogue("recording.wav", "1", session_id,
response_format="audio")
if "audio_file" in audio_audio_result:
    print(f"Audio response saved to: {audio_audio_result['audio_file']}")
```

```
    # You can play the audio or process it further
```

JavaScript

```javascript
// Text dialogue (text response)
async function textDialogue(text, characterId, sessionId = null) {
  const response = await fetch("http://localhost:8000/api/dialogue/text", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({
      text: text,
      character_id: characterId,
      session_id: sessionId,
      response_format: "text"
    })
  });
  return await response.json();
}

// Text dialogue (audio response)
async function textDialogueWithAudio(text, characterId, sessionId = null) {
  const response = await fetch("http://localhost:8000/api/dialogue/text", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({
      text: text,
      character_id: characterId,
      session_id: sessionId,
      response_format: "audio"
    })
  });

  // Check response type
  if (response.headers.get('Content-Type') === 'audio/wav') {
    // Get session ID
    const sessionId = response.headers.get('X-Session-ID');

    // Create audio URL and play
    const blob = await response.blob();
    const audioUrl = URL.createObjectURL(blob);

    // Play audio
    const audio = new Audio(audioUrl);
    audio.play();

    return {
      status: "success",
      audioUrl: audioUrl,
      sessionId: sessionId
    };
  } else {
    // Handle error
```

```
    return await response.json();
  }
}

// Audio dialogue (text or audio response)
async function audioDialogue(audioFile, characterId, sessionId = null, responseFormat =
"text") {
  const formData = new FormData();
  formData.append("audio_file", audioFile);
  formData.append("character_id", characterId);
  formData.append("response_format", responseFormat);

  if (sessionId) {
    formData.append("session_id", sessionId);
  }

  const response = await fetch("http://localhost:8000/api/dialogue/audio", {
    method: "POST",
    body: formData
  });

  // If requesting audio response
  if (responseFormat === "audio" && response.headers.get('Content-Type') === 'audio/wav') {
    // Get session ID
    const sessionId = response.headers.get('X-Session-ID');

    // Create audio URL and play
    const blob = await response.blob();
    const audioUrl = URL.createObjectURL(blob);

    // Play audio
    const audio = new Audio(audioUrl);
    audio.play();

    return {
      status: "success",
      audioUrl: audioUrl,
      sessionId: sessionId
    };
  } else {
    // Text response or error
    return await response.json();
  }
}

// Usage example
async function dialogueExample() {
  // Text input, text response
  const textResult = await textDialogue("How are you feeling today?", "1");
  console.log(`Patient response: ${textResult.responses[0]}`);

  // Use returned session ID to continue the dialogue
  const sessionId = textResult.session_id;
```

```
  // Text input, audio response
  const audioResponse = await textDialogueWithAudio("Tell me about your wound", "1",
sessionId);
  if (audioResponse.status === "success") {
    console.log(`Playing audio response, session ID: ${audioResponse.sessionId}`);
  }

  // Get audio file from input element
  const audioInput = document.getElementById('audioInput');
  if (audioInput && audioInput.files.length > 0) {
    // Audio input, text response
    const audioTextResult = await audioDialogue(audioInput.files[0], "1", sessionId);
    console.log(`Patient response: ${audioTextResult.responses[0]}`);

    // Audio input, audio response
    const audioAudioResult = await audioDialogue(audioInput.files[0], "1", sessionId,
"audio");
    if (audioAudioResult.status === "success") {
      console.log(`Playing audio response, audio URL: ${audioAudioResult.audioUrl}`);
    }
  }
}
```

Session Lifecycle

1. A new session is created on the first request (when no `session_id` is provided or it's `null`)
2. Use the returned `session_id` for subsequent requests to maintain dialogue continuity
3. Sessions automatically terminate after 1 hour of inactivity

Limitations and Recommendations

- Audio files must be in WAV format, with a recommended sampling rate of 16kHz or higher
- Each audio segment should not exceed 60 seconds
- Consecutive requests from the same user should use the same `session_id`
- When the `state` is `CONFUSED`, simplify your question or try a different expression
- When receiving audio responses, remember to extract the session ID from the `X-Session-ID` HTTP header