AllTalk-TTS Documentation

1 Overview

The AllTalk Text-to-Speech (TTS) Generation API facilitates the generation of speech from text inputs using various configuration options. This API supports different voices.

2 Endpoints

2.1 Ready Endpoint

Purpose: Check if the Text-to-Speech (TTS) service is ready to accept requests.

- **URL**: `http://127.0.0.1:7851/api/ready`
- **Method**: GET

```
curl -X GET "http://127.0.0.1:7851/api/ready"
```

Response: Ready

Note: When hosted on Azure or AWS, replace the localhost URL with your server's URL and include your API key in the headers.

```
curl -X GET "http://your-server-url/api/ready" -H "Authorization:␣Bearer␣
   your_api_key"
```

2.2 Voices List Endpoint

Purpose: Retrieve a list of available voices for generating speech.

- **URL**: `http://127.0.0.1:7851/api/voices`
- **Method**: GET

```
curl -X GET "http://127.0.0.1:7851/api/voices"
```

JSON Response:

```
{
    "voices": ["voice1.wav", "voice2.wav", "voice3.wav"]
}
```

Note: For a cloud-hosted service, include the API key as shown below.

```
curl -X GET "http://your-server-url/api/voices" -H "Authorization:␣Bearer
   ␣your_api_key"
```

2.3 Current Settings Endpoint

Purpose: Retrieve the current settings and available models for generating speech.

- **URL**: `http://127.0.0.1:7851/api/currentsettings`
- **Method**: GET

```
curl -X GET "http://127.0.0.1:7851/api/currentsettings"
```

JSON Response:

```
{
    "models_available": [
        {"name": "Coqui", "model_name": "API TTS"},
        {"name": "Coqui", "model_name": "API Local"},
        {"name": "Coqui", "model_name": "XTTSv2 Local"}
    ],
    "current_model_loaded": "XTTSv2 Local",
    "deepspeed_available": true,
    "deepspeed_status": true,
    "low_vram_status": true,
    "finetuned_model": false
}
```

Note: For cloud services, modify the request as follows.

```
curl -X GET "http://your-server-url/api/currentsettings" -H "
    Authorization: Bearer your_api_key"
```

2.4 Preview Voice Endpoint

Purpose: Generate a preview of a specified voice with hardcoded settings.

- **URL**: `http://127.0.0.1:7851/api/previewvoice/`
- **Method**: POST
- **Content-Type**: application/x-www-form-urlencoded

```
curl -X POST "http://127.0.0.1:7851/api/previewvoice/" -F "voice=
    female_01.wav"
```

JSON Response:

```
{
    "status": "generate-success",
    "output_file_path": "/path/to/outputs/api_preview_voice.wav",
    "output_file_url": "http://127.0.0.1:7851/audio/api_preview_voice.wav
        "
}
```

Note: For cloud services, include the API key.

```
curl -X POST "http://your-server-url/api/previewvoice/" -F "voice=
   female_01.wav" -H "Authorization:␣Bearer␣your_api_key"
```

2.5 Switching Model Endpoint

Purpose: Switch between different TTS models.

- **URL**: `http://127.0.0.1:7851/api/reload`
- **Method**: POST

```
curl -X POST "http://127.0.0.1:7851/api/reload?tts_method=API\␣Local"
curl -X POST "http://127.0.0.1:7851/api/reload?tts_method=API\␣TTS"
curl -X POST "http://127.0.0.1:7851/api/reload?tts_method=XTTSv2\␣Local"
curl -X POST "http://127.0.0.1:7851/api/reload?tts_method=XTTSv2\␣FT"
```

JSON Response:

```
{
    "status": "model-success"
}
```

Note: For cloud services, include the API key.

```
curl -X POST "http://your-server-url/api/reload?tts_method=API\␣Local" -H
    "Authorization:␣Bearer␣your_api_key"
```

2.6 Switch DeepSpeed Endpoint

Purpose: Enable or disable DeepSpeed mode.

- **URL**: `http://127.0.0.1:7851/api/deepspeed`
- **Method**: POST

```
curl -X POST "http://127.0.0.1:7851/api/deepspeed?new_deepspeed_value=
   True"
```

JSON Response:

```
{
    "status": "deepspeed-success"
}
```

Note: For cloud services, include the API key.

```
curl -X POST "http://your-server-url/api/deepspeed?new_deepspeed_value=
   True" -H "Authorization:␣Bearer␣your_api_key"
```

2.7 Switching Low VRAM Endpoint

Purpose: Enable or disable Low VRAM mode.

- **URL**: `http://127.0.0.1:7851/api/lowvramsetting`
- **Method**: POST

```
curl -X POST "http://127.0.0.1:7851/api/lowvramsetting?new_low_vram_value
   =True"
```

JSON Response:

```
{
    "status": "lowvram-success"
}
```

Note: For cloud services, include the API key.

```
curl -X POST "http://your-server-url/api/lowvramsetting?
   new_low_vram_value=True" -H "Authorization:␣Bearer␣your_api_key"
```

2.8 TTS Generation Endpoint (Standard Generation)

Purpose: Generate TTS from text input. Supports character voices only.

- **URL**: `http://127.0.0.1:7851/api/tts-generate`
- **Method**: POST
- **Content-Type**: application/x-www-form-urlencoded

Example Command Lines:
Character Speech:

```
curl -X POST "http://127.0.0.1:7851/api/tts-generate" -d "text_input=All␣
   of␣this␣is␣text␣spoken␣by␣the␣character." -d "text_filtering=standard"
    -d "character_voice_gen=female_01.wav" -d "language=en" -d "
   output_file_name=myoutputfile" -d "output_file_timestamp=true" -d "
   autoplay=true" -d "autoplay_volume=0.8"
```

Request Parameters:

- `text_input`: The text you want the TTS engine to produce.
- `text_filtering`: Filter for text. Options: `none`, `standard`, `html`.
- `character_voice_gen`: The WAV file name for the character's voice.
- `language`: Choose the language for TTS. Options: `ar`, `zh-cn`, `cs`, `nl`, `en`, `fr`, `de`, `hi`, `hu`, `it`, `ja`, `ko`, `pl`, `pt`, `ru`, `es`, `tr` (according to standard language conventions).
- `output_file_name`: The name of the output file (excluding the `.wav` extension).

- `output_file_timestamp`: Add a timestamp to the output file name. Options: `true`, `false`.
- `autoplay`: Enable or disable playing the generated TTS to your standard sound output device at the time of TTS generation. Options: `true`, `false`.
- `autoplay_volume`: Set the autoplay volume. Should be between 0.1 and 1.0.

JSON Response:

```
{
    "status": "generate-success",
    "output_file_path": "C:\\text-generation-webui\\extensions\\
       alltalk_tts\\outputs\\myoutputfile_1704141936.wav",
    "output_file_url": "http://127.0.0.1:7851/audio/
       myoutputfile_1704141936.wav",
    "output_cache_url": "http://127.0.0.1:7851/audiocache/
       myoutputfile_1704141936.wav"
}
```

Note: For cloud services, modify the request as follows.

```
curl -X POST "http://your-server-url/api/tts-generate" -d "text_input=All
   ␣of␣this␣is␣text␣spoken␣by␣the␣character." -d "text_filtering=standard
   " -d "character_voice_gen=female_01.wav" -d "language=en" -d "
   output_file_name=myoutputfile" -d "output_file_timestamp=true" -d "
   autoplay=true" -d "autoplay_volume=0.8" -H "Authorization:␣Bearer␣
   your_api_key"
```

2.9 TTS Generation Endpoint (Streaming Generation)

Purpose: Generate TTS from text input as an audio stream.

- **URL**: `http://localhost:7851/api/tts-generate-streaming`
- **Method**: POST
- **Content-Type**: application/x-www-form-urlencoded

Example (JavaScript):

```
const text = "Here␣is␣some␣text";
const voice = "female_01.wav";
const language = "en";
const outputFile = "stream_output.wav";
const encodedText = encodeURIComponent(text);
const streamingUrl = `http://localhost:7851/api/tts-generate-streaming?
   text=${encodedText}&voice=${voice}&language=${language}&output_file=${
   outputFile}`;
const audioElement = new Audio(streamingUrl);
audioElement.play();
```

Request Parameters:

- `text`: The actual text you want to convert to speech. Must be URL-encoded.
- `voice`: The WAV file name for the voice.
- `language`: The language for TTS. Options: `en`, `fr`, `de`, etc.
- `output_file`: The name of the output file where the audio will be streamed.

Note: For cloud services, modify the URL and include the API key in the headers.

```
const streamingUrl = `http://your-server-url/api/tts-generate-streaming?
    text=${encodedText}&voice=${voice}&language=${language}&output_file=${
    outputFile}`;
const audioElement = new Audio(streamingUrl);
audioElement.play();
```

3 Scenarios and Usage

3.1 Scenario 1: Checking Service Availability

Use: To verify if the TTS service is up and running before making any requests.

- Endpoint: `/api/ready`

3.2 Scenario 2: Retrieving Available Voices

Use: To get a list of available voices to choose from for generating speech.

- Endpoint: `/api/voices`

3.3 Scenario 3: Checking Current Settings

Use: To see which model is currently loaded and other settings like DeepSpeed and Low VRAM status.

- Endpoint: `/api/currentsettings`

3.4 Scenario 4: Previewing a Voice

Use: To generate a preview of a specific voice.

- Endpoint: `/api/previewvoice`

3.5 Scenario 5: Switching TTS Models

Use: To switch between different TTS models available on the server.

- Endpoint: `/api/reload`

3.6 Scenario 6: Enabling/Disabling DeepSpeed

Use: To enable or disable the DeepSpeed mode for performance optimization.

- Endpoint: `/api/deepspeed`

3.7 Scenario 7: Enabling/Disabling Low VRAM Mode

Use: To enable or disable the Low VRAM mode.

- Endpoint: `/api/lowvramsetting`

3.8 Scenario 8: Generating Standard TTS

Use: To generate a TTS output with character voices only, saved as a file.

- Endpoint: `/api/tts-generate`

3.9 Scenario 9: Generating Streaming TTS

Use: To generate a TTS output as an audio stream.

- Endpoint: `/api/tts-generate-streaming`

4 Summary of Endpoints

- **Ready Endpoint**: `/api/ready`
- **Voices List Endpoint**: `/api/voices`
- **Current Settings Endpoint**: `/api/currentsettings`
- **Preview Voice Endpoint**: `/api/previewvoice`
- **Switching Model Endpoint**: `/api/reload`
- **Switch DeepSpeed Endpoint**: `/api/deepspeed`
- **Switching Low VRAM Endpoint**: `/api/lowvramsetting`
- **TTS Generation Endpoint (Standard)**: `/api/tts-generate`
- **TTS Generation Endpoint (Streaming)**: `/api/tts-generate-streaming`

For cloud hosting scenarios, always ensure to replace localhost URLs with your server's URL and include the necessary API keys in the request headers.