REVERIE

Language Technologies

API REFERENCE DOCUMENT

For

REVERIE TTS API

Abstract

Reverie TTS API Reference Document

You can use our API to convert the text to speech. This document describes the available variables, commands, and interfaces that make up the Reverie TTS API.

This document will help the developers to implement Automated Voice Recognition (ASR) in their applications. The API is platform agnostic, which means any device that can record the speech can use this API.

The API is organized around REST. All requests should be made over SSL. All request and response bodies, including errors, are encoded in JSON.

Table of Contents

1. Overview

Reverie TTS (Text to Speech) is a solution that turns text into lifelike speech, allowing you to create applications that talk in multiple Indic languages and build comprehensive speech-enabled products.

The Reverie TTS service offers Neural Text-to-Speech voices, delivering innovative enhancements in speech quality through a state-of-art machine learning approaches. You can select the ideal voice and tone to build the natural and human-like speech-enabled applications in the market to enable the interactive customer experience.

1.1. Languages Supported

The Reverie TTS solution can understand popularly used two languages and offers service to all the domains like Telecom, Banking, Entertainment, etc.:

Languages	Language code
1. Hindi	hi
2. English	en

Note: *Our Research and Development team is continuously working to enable all the leading Indian languages on the text-to-speech platform.*

1.2. Key Features

Reverie TTS API has some powerful features that help you to serve your consumers effectively in their native language:

- **Faster than Real-time Speech Synthesis**
  An effective Speech-enabled application should deliver the conversational user experience accurately and swiftly. As soon as you pass the text to Reverie TTS, it returns the audio to your application as a stream so you can play the voices immediately.
- **Customize Speech Model**
  Tailor the text-to-speech solution to suit your requirements. The Reverie TTS will support lexicons and SSML tags, which allow you to manage the speech aspects like volume, pitch, speed rate, the pronunciation of words with context, etc.
- **High-Quality & Accurate Pronunciation**
  The Reverie TTS's neural network technology enables you to deliver high-quality voice output with a more accurate pronunciation of words. The solution provides a wide selection of lifelike sounding male and female voices.
- **Distribute Speech in Multiple Formats**
  You can create and save the speech files in multiple formats like WAV, OGG, MP3, and FLAC and deploy them to serve with apps or different applications.

2. Getting Started

The Reverie TTS API allows you to synthesize high-quality spoken audio in multiple formats. You'll always receive audio data or an error message in the same HTTP transaction.

2.1. Versioning

When we make backward-incompatible changes to the API, we release new dated versions. The current version is **v1.0.**

All requests will use your account API settings unless you override the API version. Events generated by API requests will always be structured according to your account API version.

2.2. How it works

The process flow to generate the text to audio file is:

1. Post the request by:
    a. Define the Language Code
    b. Enter text for which you want to generate the speech
    c. Define SSML Code if required
2. The Response is provided in audio format with sampling rate of 22.05KHz.

2.3. Sample Code

URL:

```
https://hackapi.reverieinc.com/tts
```

Request:

```
curl -X POST \
  https://hackapi.reverieinc.com/tts \
  -H 'Content-Type: application/json' \
  -d '{
"lang": "en",
"text": "<s>Waiting</s><break time=\"2s\"/><s>Finished</s>"
}'
```

The final error response:

```
{
    "error": "Invalid `lang` value: kn"
}
```

3. API Reference

3.1. HTTP Request URL

URL Elements:

```
https://hackapi.reverieinc.com/tts
```

3.2. Request Post Body

Element	Type	Is Mandatory?	Description
lang	string	Yes	The supported language code to speak the text in. To know the language code of the supported languages, refer section 1.1 Languages Supported above
text	string	Yes	The textual content for converting to speech
speed	Integer (seconds)	No	The speech rate of the generated audio file. Allows values: from 0.5 (slowest speed) up to 1.5 (fastest speed). ***Note:*** *By default, the speed value: 1 (normal speed).*
pitch	Integer (seconds)	No	Speaking pitch, in the range [-3.0, 3.0]. 3 means an increase of 3 semitones from the original pitch. -3 means a decrease of 3 semitones from the original pitch. ***Note:*** *By default, the pitch value: 0*
SSML	string	No	SSML is an XML-based markup language that provides text annotation for speech-synthesis applications.

3.3. Response

The response is an audio file.

- The audio format supported are:
  - WAV
  - OGG
  - MP3
  - FLAC
- The service always synthesizes audio with a sampling rate of 22.05KHz.

4. SSML (Speech Synthesis Markup Language)

SSML tags are used to customize the way a text-to-speech engine creates audio. The tags are used to add pauses, change emphasis, and change the pronunciation. Pronouncing numbers in cardinal format (example: 123 is spoken as One Hundred and Twenty Three).

In some cases, you may want additional control over the speech generated from the text in the response. For example, you may want a string of digits read back as a standard telephone number and not in cardinal form. The Reverie TTS provides control with Speech Synthesis Markup Language (SSML) support.

4.1. <speak>

The **<speak>** is the root element of an SSML document. When using SSML with the Reverie TTS, surround the text to be spoken with this tag.

By, default the API will add the **<speak>** tag if not defined in the request.

4.2. <p> & <s>

<p> represents a Paragraph. This tag is used to add a pause between paragraphs in the text.

Note: *By default, the paragraphs are separated by an 800ms break.*

<s> represents a Sentence. This tag is used to add a pause between lines or sentences in the text.

Note: *By default, sentences within a paragraph are separated by a 400ms break.*

Untagged text request is converted to the SSML format below:

```
<speak>
<p><s>This is a sentence</s></p>
</speak>
```

Note: *Recommend to use* **<p>** *and* **<s>** *tags to structure your requests, especially when using* **<break>**.

4.3. <break>

<break> represents a pause in the speech. Set the length of the pause with the time attribute.

Note: **<break>** *is currently not supported within a sentence. It can only exist before or between sentences or paragraphs.*
Note: *The break time between a sentence is 400ms*

Attribute	Possible Values
time	Duration of the pause. Include the unit with the time – **s** for seconds and **ms** for milliseconds Example: 1s for 1 second, 450ms for 450 milliseconds

Sample Code:

```
<s>Fetching results</s>
<break time="3s"/>
<s>Found 3 items</s>
```

4.4. <sub>

Pronounce the specified word or phrase as a different word or phrase. Specify the pronunciation to substitute with the **alias** attribute.

Attribute	Possible Values
alias	The word or phrase to speak in place of the tagged text.

Sample Code:

```
<sub alias="World Wide Web">www</sub>
```

4.5. <say-as>

The **<say-as>** tag will allow you to provide instructions for how a particular text must be spoken. Many of these features are automatically detected in speech by the TTS engine, but the say-as command allows you to mark them specifically.

Use the <say-as> tag with the **interpret-as** attribute to tell Reverie TTS how to speak certain characters, words, and numbers.

Attribute	Possible Values
Interpret-as	uses some available possibility values to speak the given text.

The following values are available with interpret-as:

4.5.1 Cardinal

Interprets the numerical text as a cardinal number, as in 1,234

Note: *By default, the API will interpret and pronounce the numbers in Cardinal format.*

Example:

1234 is spoken as One thousand two hundred and thirty-four

Sample Code:

```
<say-as interpret-as="cardinal">1234</say-as>
```

4.5.2 Ordinal

This value will interpret and speak the ordinal value for the given digit within the enclosed tag.

Example:

44 is interpreted as **Forty-Fourth**

2 is interpreted as **Second**

Sample Code:

```
<say-as interpret-as="ordinal">2</say-as>
```

4.5.3 Character

This value will speak the characters in a given text within the enclosed tag.

Example:

HELLO is interpreted, and each character is pronounced separately.

50WS is interpreted and spoked as Five Zero W S

Sample Code:

```
<say-as interpret-as="characters">50WS</say-as>
```

4.5.4 Digits

This value spells out the digits in a given number within the enclosed tag.

Example:

44 is interpreted as **Four Four**

1234 is interpreted as **One Two Three Four**

Sample Code:

```
<say-as interpret-as="digits">1234</say-as>
```

4.5.5 Date

This value will speak the date within the enclosed tag, using the format given in the associated format attribute. The format attribute is mandatory for use with the date value of interpret-as.

Note: *The digits length for each parameter is given below:*

1. *d: Date: 1 or 2 digits*
2. *m: Month: 1 or 2 digits*
3. *y: Year: 4 digits*

This gives a list of dates in all the various formats:

Date Format	How is the date Read?
dmy	day-month-year **Example:** 10-02-1990 = tenth February, nineteen ninety **Sample Code:** `<say-as interpret-as="date" format="dmy">10-2-1990</say-as>`

Date Format	How is the date Read?
mdy	month-day-year **Example:** 02-10-1990 = February tenth, nineteen ninety **Sample Code:** `<say-as interpret-as="date" format="mdy">2-10-1990</say-as>`
dm	day-month **Example:** 10-2 = Tenth February **Sample Code:** `<say-as interpret-as="date" format="dm">10-2</say-as>`
md	month-day **Example:** 2-10 = February Tenth **Sample Code:** `<say-as interpret-as="date" format="md">2-10/say-as>`

4.5.6 Currency

This value is used to control the synthesis of monetary quantities.

The currency formats supported and spoken by the API is:

Currency	Symbols Supported
Rupee & Paisa	₹, Rs.,रु **Example:** ₹10.50 : Ten rupees fifty paise **Sample Code:** `<say-as interpret-as="currency">₹10.50</say-as>`
Dollars & Cents	$ **Example:** $10.50 : Ten dollars fifty cents **Sample Code:** `<say-as interpret-as="currency">$10.50</say-as>`

Currency	Symbols Supported
Pounds & Pence	£ **Example:** £10.50 : Ten pounds fifty pence **Sample Code:** `<say-as interpret-as="currency">£10.50</say-as>`
Euros & Cents	€ **Example:** €10.50 : Ten euros fifty cents **Sample Code:** `<say-as interpret-as="currency">€10.50</say-as>`

4.5.7 URL

This value is used to control the synthesis of the website address.

By default, the API will interpret the website address and speak in the right format.

Note: *The API will pronounce symbols as present in a web address.*

Example:

reverieinc.com is spoken as **reverieinc dot com**

Sample Code:

```
<say-as interpret-as="url"> reverieinc.com</say-as>
```

4.6. API Default Behaviour

Reverie TTS automatically handles normal punctuation, such as pausing after a period or speaking a sentence ending in a question mark as a question. The text that is automatically interpreted and spoken by API are:

4.6.1 Numbers

The number is automatically detected and doesn't require a tag. By default, the number is pronounced in cardinal format. Use ordinal or digits tag to pronounce it in ordinal format (First) and digit form (one, two, three) respectively.

The API will automatically identify:

- Decimals. The numbers after the decimal point are considered as digits
- Negative Numbers: The "-" sign is read as minus

4.6.2 Symbols

The API will automatically interpret a set of symbols in the text and will speak accordingly. The symbols automatically detected are:

Symbols	Interpret as
.	• "dot" when present between two letters, • "point" in numbers, • sentence break if last character or followed by space
,	• Pause, Duration dependent on training data
@	• "at"
#	• "hash"
%	• "percent"
&	• "and"
✳	• "star"
-	• "minus" in numbers • "dash" in <say-as characters>
✚	• "plus"
=	• "equals"
°	• "degrees"
°C	• "degrees celsius"
°F	• "degrees fahrenheit"
/	• "slash"
\	• "backslash"

4.6.3 HANDLING ERRORS

Our API raises exceptions for many reasons, such as a failed connection, invalid parameters, authentication errors, and network unavailability. We provide more specific machine-readable messages with an error response so that users can react to errors more effectively. In the API response, if the **success=false**, then the **cause** will display the reason for the failure occurred. Refer section 4.6.4 API Final Output Cause below to view the list of causes and description for the cause

4.6.4 API Final Output Cause

Cause	Description
Ready	The connection between the client system and server is established, and you can pass the audio input.
EOF received	The input is provided and then stopped respectively by the user successfully.
Partial	The output received by the API is partial
Silence detected	After starting recording, the silence was recognized for more than 1 second, and the API has terminated the connection
Too many samples	The data input received length exceed 10 seconds
Timeout	The connection was established and left idle for 10 seconds
CSR limit exhausted	After the completion of the subscribed transcription limit
Language not supported	The entered language code is incorrect