# Text To Speech

POST https://api.sarvam.ai/text-to-speech

Content-Type: application/json

This is the model to convert text into spoken audio.

The output is a wave file encoded as a base64 string.

Reference: https://docs.sarvam.ai/api-reference-docs/text-to-speech/convert

## OpenAPI Specification

```yaml

openapi: 3.1.1

info:

title: Text to Speech

version: endpoint\_textToSpeech.convert

paths:

/text-to-speech:

post:

operationId: convert

summary: Text to Speech

description: |-

This is the model to convert text into spoken audio.

The output is a wave file encoded as a base64 string.

tags:

- - subpackage\_textToSpeech

parameters:

- name: api-subscription-key

in: header

required: true

schema:

type: string

responses:

'200':

description: Successful Response

content:

application/json:

schema:

$ref: '#/components/schemas/Sarvam\_Model\_API\_TextToSpeechResponse'

'400':

description: Bad Request

content: {}

'403':

description: Forbidden

content: {}

'422':

description: Unprocessable Entity

content: {}

'429':

description: Quota Exceeded

content: {}

'500':

description: Internal Server Error

content: {}

requestBody:

content:

application/json:

schema:

$ref: '#/components/schemas/Sarvam\_Model\_API\_TextToSpeechRequest'

components:

schemas:

SarvamModelApiTextToSpeechRequestText:

oneOf:

- type: string

Sarvam\_Model\_API\_TextToSpeechLanguage:

type: string

enum:

- value: bn-IN

- value: en-IN

- value: gu-IN

- value: hi-IN

- value: kn-IN

- value: ml-IN

- value: mr-IN

- value: od-IN

- value: pa-IN

- value: ta-IN

- value: te-IN

Sarvam\_Model\_API\_TextToSpeechSpeaker:

type: string

enum:

- value: anushka

- value: abhilash

- value: manisha

- value: vidya

- value: arya

- value: karun

- value: hitesh

Sarvam\_Model\_API\_SpeechSampleRate:

type: string

enum:

- value: '8000'

- value: '16000'

- value: '22050'

- value: '24000'

Sarvam\_Model\_API\_TextToSpeechModel:

type: string

enum:

- value: bulbul:v2

TextToSpeechOutputAudioCodec:

type: string

enum:

- value: mp3

- value: linear16

- value: mulaw

- value: alaw

- value: opus

- value: flac

- value: aac

- value: wav

Sarvam\_Model\_API\_TextToSpeechRequest:

type: object

properties:

text:

$ref: '#/components/schemas/SarvamModelApiTextToSpeechRequestText'

target\_language\_code:

$ref: '#/components/schemas/Sarvam\_Model\_API\_TextToSpeechLanguage'

speaker:

oneOf:

- $ref: '#/components/schemas/Sarvam\_Model\_API\_TextToSpeechSpeaker'

- type: 'null'

pitch:

type:

- number

- 'null'

format: double

pace:

type:

- number

- 'null'

format: double

loudness:

type:

- number

- 'null'

format: double

speech\_sample\_rate:

oneOf:

- $ref: '#/components/schemas/Sarvam\_Model\_API\_SpeechSampleRate'

- type: 'null'

enable\_preprocessing:

type: boolean

model:

$ref: '#/components/schemas/Sarvam\_Model\_API\_TextToSpeechModel'

output\_audio\_codec:

oneOf:

- $ref: '#/components/schemas/TextToSpeechOutputAudioCodec'

- type: 'null'

required:

- text

- target\_language\_code

Sarvam\_Model\_API\_TextToSpeechResponse:

type: object

properties:

request\_id:

type:

- string

- 'null'

audios:

type: array

items:

type: string

required:

- request\_id

- audios

```

## SDK Code Examples

```python

from sarvamai import SarvamAI

client = SarvamAI(

api\_subscription\_key="YOUR\_API\_SUBSCRIPTION\_KEY",

)

client.text\_to\_speech.convert(

text="text",

target\_language\_code="bn-IN",

)

```

```typescript

import { SarvamAIClient } from "sarvamai";

const client = new SarvamAIClient({ apiSubscriptionKey: "YOUR\_API\_SUBSCRIPTION\_KEY" });

await client.textToSpeech.convert({

text: "text",

target\_language\_code: "bn-IN"

});

```

```go

package main

import (

"fmt"

"strings"

"net/http"

"io"

)

func main() {

url := "https://api.sarvam.ai/text-to-speech"

payload := strings.NewReader("{\n \"text\": \"foo\",\n \"target\_language\_code\": \"bn-IN\"\n}")

req, \_ := http.NewRequest("POST", url, payload)

req.Header.Add("api-subscription-key", "<apiKey>")

req.Header.Add("Content-Type", "application/json")

res, \_ := http.DefaultClient.Do(req)

defer res.Body.Close()

body, \_ := io.ReadAll(res.Body)

fmt.Println(res)

fmt.Println(string(body))

}

```

```ruby

require 'uri'

require 'net/http'

url = URI("https://api.sarvam.ai/text-to-speech")

http = Net::HTTP.new(url.host, url.port)

http.use\_ssl = true

request = Net::HTTP::Post.new(url)

request["api-subscription-key"] = '<apiKey>'

request["Content-Type"] = 'application/json'

request.body = "{\n \"text\": \"foo\",\n \"target\_language\_code\": \"bn-IN\"\n}"

response = http.request(request)

puts response.read\_body

```

```java

HttpResponse<String> response = Unirest.post("https://api.sarvam.ai/text-to-speech")

.header("api-subscription-key", "<apiKey>")

.header("Content-Type", "application/json")

.body("{\n \"text\": \"foo\",\n \"target\_language\_code\": \"bn-IN\"\n}")

.asString();

```

```php

<?php

$client = new \GuzzleHttp\Client();

$response = $client->request('POST', 'https://api.sarvam.ai/text-to-speech', [

'body' => '{

"text": "foo",

"target\_language\_code": "bn-IN"

}',

'headers' => [

'Content-Type' => 'application/json',

'api-subscription-key' => '<apiKey>',

],

]);

echo $response->getBody();

```

```csharp

var client = new RestClient("https://api.sarvam.ai/text-to-speech");

var request = new RestRequest(Method.POST);

request.AddHeader("api-subscription-key", "<apiKey>");

request.AddHeader("Content-Type", "application/json");

request.AddParameter("application/json", "{\n \"text\": \"foo\",\n \"target\_language\_code\": \"bn-IN\"\n}", ParameterType.RequestBody);

IRestResponse response = client.Execute(request);

```

```swift

import Foundation

let headers = [

"api-subscription-key": "<apiKey>",

"Content-Type": "application/json"

]

let parameters = [

"text": "foo",

"target\_language\_code": "bn-IN"

] as [String : Any]

let postData = JSONSerialization.data(withJSONObject: parameters, options: [])

let request = NSMutableURLRequest(url: NSURL(string: "https://api.sarvam.ai/text-to-speech")! as URL,

cachePolicy: .useProtocolCachePolicy,

timeoutInterval: 10.0)

request.httpMethod = "POST"

request.allHTTPHeaderFields = headers

request.httpBody = postData as Data

let session = URLSession.shared

let dataTask = session.dataTask(with: request as URLRequest, completionHandler: { (data, response, error) -> Void in

if (error != nil) {

print(error as Any)

} else {

let httpResponse = response as? HTTPURLResponse

print(httpResponse)

}

})

dataTask.resume()

```