# Vocalware API Reference

## Table of Contents

## Introduction

The Vocalware API enables you to use our cloud based Text-To-Speech service, to generate & play audio in real-time within your online application. By the term “online application” we refer to *any online program*, including: web pages, or native code apps on either desktop, server or mobile device. The only requirement is that your application has access to an internet connection fast enough to stream 48kbps audio.

The Vocalware API allows you to generate audio and control audio playback. **The API comes in two flavors: JavaScript/HTML5, and HTTP-REST** - so it can be easily incorporated into any application. Whether your application runs in-browser or standalone, on mobile, desktop or server - one of our API flavors will work for you.

The Vocalware API supports TTS in over 30 languages, with several voices available in each. The API allows you to specify the language and voice to use, as well as optional audio effects such as pitch, echo, etc. A list of languages and voices & their respective IDs to use in the API is available in Appendix A.

## Select the API Flavor to Use

The API comes in two flavors:

1. JavaScript/HTML5 - also referred to as the JavaScript API
2. HTTP-REST - also referred to as the HTTP API

To proceed, you should first identify the section of this document that refers to the API flavor you plan to use. The JavaScript API is covered by the first section of this guide. The HTTP API is covered by the second section.

How to select the API flavor that’s right for you? Here are several rules of thumb:

- in your web pages – use the JavaScript API (supports mobile browsers as well)
- in your standalone (out of browser) app, including mobile app - use the HTTP API
- on your server – use the HTTP API

Note: Whether mobile, desktop or server, the decision boils down to whether your intended use is within a web browser or not. If it is, use the JS API. If it is not – use the HTTP API.

## Additional Resources

If you have any questions, or run into difficulty when trying to use any of our APIs, please check out our support pages, where you will be able to access:

- Frequently Asked Questions covering a large number of issues.
- API examples for each of the API functions listed here, including full source code, covering both Javascript and HTTP APIs.
- Send a note to our support team.

## The JavaScript API

### *Introduction*

The JavaScript API operates by way of an invisible client side code object ("Agent"), that your web page loads & can then access via the API functions.

> ***Note***: This works transparently on both Desktop and Mobile browsers, as the client side Agent code automatically adapts to the client platform's capabilities.

The API supports TTS audio generation as well as playback control functionality. The interface consists of a set of client side JavaScript calls, and does not require you to make any call to our servers, as the client side API encapsulates all interaction with the Vocalware servers.

> ***Tip:*** The simplest way to handle playback of the generated audio, is to control it via the documented API functions. Using these high level functions makes direct access to the audio data unnecessary.

To get started with the JS API you need to:

a. Create a JS API object in your account's My APIs page. Copy the 'embed code' unique to your API object – and paste it into the BODY section of your page.
b. Specify in your 'Security Settings' page the domain (or several domains) in which this API is to operate.
c. Implement the vw_apiLoaded callback in your web page to receive notice that your API is ready.

**Important caveats / pitfalls to avoid -**

- Your embed code is specific to your account and for your protection will allow playback only from the domain(s) you specify. **Specifying a domain is mandatory. Your API will not function without it.**

- The “vw_apiLoaded” status callback is dispatched when the API is ready. It is therefore advisable to implement the “vw_apiLoaded” callback – and avoid calling any API function prior to receiving confirmation that loading has completed. **API functions work only after the API has completed loading.**

Note that your embed code must be placed within the BODY section of the HTML page, and will not work otherwise!  See additional detail below in “Using your Embed Code”.

In the next sections you will find instructions and code examples explaining how to use your embed code as well as a listing of the API function calls.

## *Programming for Mobile*

The JavaScript API operates on ‘desktop’ as well as mobile browsers (the term ‘desktop’ is used here to refer to non-mobile client side environments, such as desktop and laptop computers of all types). This means that you need not do anything special in order to support mobile functionality in your web pages when using the Vocalware JavaScript API. That said, there are a couple of differences between mobile and desktop that you should be aware of.

The JavaScript API is fully compatible with all major mobile browsers – and with two exceptions will function in the same way within mobile browsers as it will on desktop browsers. One exception is with the function ‘setPlayerVolume’ – which does not have any effect in some mobile browsers – but there is no harm in making the call.

**"Play-on-load":** Another important difference is that on mobile browsers, the first call to the API must always be user driven (e.g. user clicks on a button). This restriction prevents the web page from automatically speaking to the viewer unprompted. Trying to do so will not cause an error – but will simply not work.
This same restriction is being introduced into desktop browsers as well, where the implementation is not uniform, or permanent. Some browsers will allow play-on-load for a user who has visited a page before and had interacted with media on the page. The policies enacted by browsers in this regard are both evolving & undocumented, so there is no point in attempting to describe them. The main takeaways here should be:

- it's ok to try to play the audio as the page loads (verify that API has loaded first!)
- you should be aware that such playback may or may not be blocked, and will always be blocked on mobile.
- no need for special attention to coding for mobile browsers - the API operates the same across browsers and devices.

## *Using your Embed Code*

The embed code is a code segment unique to your account, or more specifically to an API Object within your account. Instructions and examples below explain how to incorporate the embed code.

### JavaScript Instructions

Locate the embed code in your account in the "My APIs" section, and copy it. Paste your embed code into your HTML page's BODY section. Needless to say that your page must have a BODY section to be able to fulfill this requirement… This instruction applies to mobile as well as non-mobile web pages.
The exact location within your HTML is not significant, though it is best not to include it within FORM brackets or other nested HTML structures.

Use the Javascript API functions defined below.

#### Error Handling

All errors are written to the console. `'sayText'` errors, if encountered, are also returned via callback (in addition to being written to the console. Please see documentation for `vh_apiError` for more details.

This API is designed to fail silently if an error is encountered, with the realization that it is not helpful to display or expose an error scenario to your web page visitors.

## *Playback Control Functions*

### sayText (txt,voice,lang,engine,[effect], [effLevel])

Real-time (dynamic) Text-To-Speech (TTS).

Note: This function will work only within a *licensed domain* for the account. Domain specific licensing is a security measure. If playback is attempted within a domain that is not specifically licensed for the account, this call will generate an error.

**Arguments:**

| | |
|---|---|
| `txt` | Required. String - The text to speak. Most languages are limited to 600 characters. The exceptions are Chinese & Japanese which are limited to 150 characters. A longer text string will be truncated. |

| | |
|---|---|
| `voice` | Required. Integer – Voice ID, as listed in Appendix A. |
| `lang` | Required. Integer – Language ID, as listed in Appendix A. |
| `engine` | Required. Integer – Voice Family ID. See languages and voices listed listed in Appendix A. |
| `effect` | Optional. Character. Audio effect – one of: |

- "D" – Duration levels: -3, -2, -1, 1, 2, 3
- "P" – Pitch levels: -3, -2, -1, 1, 2, 3
- "S" – Speed levels: -3, -2, -1, 1, 2, 3
- "R" – Robotic:
  - Bullhorn level: 3 (note: levels 1 and 2 are deprecated)
- "T" – Time:
  - Echo level: 1
  - Reverb level: 2
  - Flanger level: 3
  - Phase level: 4
- "W" – Whisper levels: 1, 2, 3

`effLevel` Optional. Integer. Effect level must be provided if effect is provided.

**Examples:**
```
sayText('Hello World',3,1,3)
sayText('Hello World',3,1,3,'S',-1)
```

---

## setPlayerVolume (level)

Set playback volume, or mute the audio.

**Arguments:**

`level` Required. Integer (0-10) – Default = 7.
a value from 0 to 10; 0 is equivalent to mute, 1 is softest, 10 is loudest.

**Example:**
```
setPlayerVolume(10)
```

Note:
- Setting the volume to 0, does not stop playback or the audio stream. It only affects the audio volume. To stop playback, use the function stopSpeech(). To pause playback, use the function freezeToggle().
- Calling this function has no effect on some mobile browsers.

---

## stopSpeech ()

Stop audio playback in progress. If audio is not currently playing, stopSpeech has no effect (i.e. it does not prevent speech that has not yet begun).

**Arguments:**

None.

**Example:**

```
stopSpeech()
```

---

## freezeToggle ()

Toggle between the pause and play states. If playback is in progress, it is paused. If playback is paused, it is resumed from the point it was paused.

**Arguments:**

None.

**Example:**

```
freezeToggle()
```

---

## setStatus (interruptMode,progressInterval,reserved1,reserved2)

Set several status values which govern various aspects of playback.

**Arguments:**

`interruptMode`

Required. Integer (0/1) – Default = 0.
If set to 0 consecutive audio playback function calls (sayText) are queued for consecutive playback.
If set to 1 current audio is interrupted when sayText is called.

`progressInterval`

Required. Non-negative Integer – Default = 0.
The audio progress interval value controls progress callbacks which take place during playback. The callback function

vw_audioProgress(percent_played)

is called during playback if the value of ‘progressInterval’ is non-zero. The non-zero value determines the frequency of the call.
The value must be an integer greater than or equal to 0. When greater than 0, the callback "vw_audioProgress(percent_played)" is triggered at the frequency specified by the number (in seconds). The callback returns the percent of the current audio that has played. Callbacks will continue for all subsequent audios played once this field is set. Set back to 0 for the callbacks to cease.

`reserved1`

Required. Integer. Set to 0.

`reserved2`

Required. Integer. Set to 0.

**Example:**

```
setStatus(1,0,0,0)
```

---

## *Status Callback Functions*

Vocalware Callback Functions enable coordination between playback and your page or application. To set them up, please follow these instructions.

### Setting up Vocalware Callback Functions in your HTML page:

Events during playback trigger calls to specific JavaScript functions in your page, if such functions exist. To take advantage of these calls you must **add the appropriate JavaScript functions to your page**. Note that you do not need to add callback functions which you do not intend to use.

### vw_apiLoaded (apiID)

Triggered when the API is fully loaded. Use this callback to verify API is ready, prior to making any function calls.

**Arguments:**

| | |
|---|---|
| `apiID` | The id of the api being loaded. |

**Example -**

```
function vw_apiLoaded(apiID){
      alert(“the API is loaded”);
      }
```

---

## vw_audioProgress (percentPlayed)

Called during playback, if and only if the ‘progressInterval’ status is set. vw_audioProgress is repeatedly called at regular intervals during playback. The intervals are determined according to the value of the ‘progressInterval’ status. See ‘setStatus’ API call for information about how to set this value.
This callback can be used to enable synchronization between playback and other events taking place at the same time. For example: highliting text segments, or visual elements on the page in coordination with speech playback.

**Arguments**

| | |
|---|---|
| `percentPlayed` | A value between 0 and 100 which indicated the proportion of audio already played. |

**Example -**

```
      function vw_audioProgress(percentPlayed) {
      }
```

---

## vw_talkStarted ()

Triggered when the audio playback begins.

**Example -**

```
      function vw_talkStarted(){
      }
```

---

## vw_talkEnded ()

Triggered when audio playback is done.

**Example -**

```
      function vw_talkEnded(){
      }
```

---

## vw_audioStarted ()

Triggered when audio playback begins. Unlike vw_talkStarted() this event is fired for each audio playback in a sequence.

**Example -**
```
      function vw_audioStarted(){
      }
```

---

## vw_audioEnded ()

Triggered when audio playback ends. Unlike talkEnded() this event is fired for each audio playback in a sequence.

**Example -**
```
      function vw_audioEnded(){
      }
```

---

## vw_apiError (errCode, errDesc)

Triggered when an error is encountered in when processing a sayText call. An error code and description are returned - as specified in Appendix C.

**Arguments**

| | |
|---|---|
| `errCode` | Integer. Error code is returned. See in-stream error codes in Appendix C. |
| `errDesc` | String. A Description of the error. |

**Example -**
```
      function vw_apiError(errCode,errDesc) {
            alert(“an error has occurred: ”+errCode+" "+errDesc);
      }
```

---

## The HTTP REST API

**Note:** This API supports both HTTP and HTTPS protocols. For simplicity only HTTP is documented. Where you see HTTP mentioned you may assume either HTTP or HTTPS can be used.

### *The HTTP GEN Request*

This HTTP request supports either GET or POST parameter passing. The syntax example below describes only the HTTP GET request.

**Syntax Example:**

```
http://www.vocalware.com/tts/gen.php?EID=2&LID=1&VID=2&TXT=Test+Message
&EXT=mp3&FX_TYPE=p&FX_LEVEL=1&ACC=8879&API=283475&SESSION=50cdbf7c4e5d11
7dcb2efff424e10054&HTTP_ERR=1&CS=ec1e9c6249980c208186bd64ce8ce6e1
```

**Note: BOLD parameters are required**

| Parameters | Description |
|---|---|
| **EID** | Engine Id. |
| **LID** | Language Id. |
| **VID** | Voice Id. |
| **TXT** | Text to be used for audio creation (URL Encoded) |
| EXT | 'mp3' or 'ogg'. Default is mp3 |
| FX_TYPE | Sound effect type. Default is empty (no effect) |
| FX_LEVEL | Sound effect level. Default is empty (no effect) |
| **ACC** | Account id |
| **API** | API id |
| SESSION | Used to verify the session (see Session Verification section) |
| HTTP_ERR | *Deprecated – do not provide, ignored if used.* |
| **CS** | Checksum – implemented as an md5 of all above parameter and your secret phrase |

**CS** = md5 (**EID** + **LID** + **VID** + **TXT** + EXT + FX_TYPE + FX_LEVEL + **ACC** + **API** + SESSION + HTTP_ERR + **SECRET PHRASE**)

**Return values:**
In case of success, mp3 or ogg binary stream is returned, and the HTTP header status "200". The returned audio data is single channel (mono), has a 22Khz sample rate and is encoded at a 48Kbps bitrate.

In case of failure, an an HTTP header error status code is returned, as well as an "in-stream" error code and message.
If the http header status code is different than 200, check the returned data for the In-Stream code and message for additional information.
See Appendix B: Error Handling & Codes for details.

## *Session Verification*

Session verification is an optional feature designed to protect your account. Here's how it works:

- When your application makes an HTTP GEN request, and if the checksum proves to be authentic, we call a predefined URL on your servers (the "Callback URL").
- You specify the Callback URL for us to use as part of your Vocalware account security settings.
- The call is an HTTP POST request, with two parameters – your account ID and the session ID you provided when making the GEN request.
- When we call you – you may authorize the session, or reject it.
- If the Callback URL for the account is not setup, or if the Session parameter is not provided, then no callback attempt is made.
- If the Callback URL for the account is not setup, checksum is calculated without the Session parameter even if present.
- Note: we cache your responses. Subsequent GEN calls that provide the same session ID will not always generate a callback.

Why use session verification?
If your GEN requests originate from your server, there is no need to setup session verification. But if you are making GEN requests from a client application (i.e. a web page) – then session verification is highly advisable to secure your account.

**Verification Syntax:**
POST request to account Callback URL.

| **Parameter** | **Description** |
|---|---|
| ACC | Account id |
| SESSION | Provided session id |

| **Return Values** | **Description** |
|---|---|
| 1 | SESSION is valid |
| 0 | Error – invalid session |

## *Example*

This example page demonstrates how to put together the HTTP GEN request:
http://www.vocalware.com/support/rest-api

## *The Create API Request*

This call enables you to programatically create new APIs in your account. This can be useful for setting up individual users or clients to be able to track their usage via Vocalware analytics. The effect of this call is the same as logging into your account and manually creating a new API.
This HTTP request supports either GET or POST parameter passing. The syntax example below describes only the HTTP GET request.

**Syntax Example:**

http://www.vocalware.com/tts/createapi.php?ACC=8879&APIName=abcd&APIType=h &CS=ec1e9c6249980c208186bd64ce8ce6e1

**Note: BOLD parameters are required**

| **Parameters** | **Description** |
|---|---|
| **ACC** | Account id |
| **APIName** | Name of new API to be created. String. Under 20 characters. |
| **APIType** | Type of new API to be created. "j" - Javascript; "h" - HTTP. |
| **CS** | Checksum – implemented as an md5 of all above parameter and you secret phrase |

**CS** = md5 (**ACC** + **APIName** + **APIType** + **SECRET PHRASE**)

**Return values:**
In case of success -
if Type = j - Embed code for new API is returned.
if Type = h - ID of new API is returned.

In case of failure, an error code and message are returned. See Appendix B: Error Codes for more information.

## *The Get Stream Balance Request*

This call enables you to retrieve the current stream balance from your Vocalware account. Note that the information is not real-time but updated about every 10 minutes, This HTTP request supports either GET or POST parameter passing. The syntax example below describes only the HTTP GET request.

**Syntax Example:**

```
http://www.vocalware.com/tts/getbalance.php?
ACC=8879&CS=ec1e9c6249980c208186bd64ce8ce6e1
```

**Note: BOLD parameters are required**

| **Parameters** | **Description** |
|---|---|
| **ACC** | Account id |
| **CS** | Checksum – implemented as an md5 of all above parameter and you secret phrase |

**CS** = md5 (**ACC** + **SECRET PHRASE**)

**Return values:**
In case of success -
The number of available streams in your account is returned (integer).

In case of failure, an error code and message are returned. See Appendix B: Error Codes sub-section for more information.

## *Generating the Checksum*

To calculate the checksum, concatenate all the parameters in the order they appear in this document and add your 'Secret Phrase', which you can find in the 'security settings' on your 'my APIs' page.

Apply the md5 one way function to the resulting string, to generate the checksum X, and append it to the parameter list as CS=X
The checksum is created in the following way:
**CS** = md5 (**EID** + **LID** + **VID** + **TXT** + EXT + FX_TYPE + FX_LEVEL + **ACC** + **API** + SESSION + HTTP_ERR + **SECRET PHRASE**)

**Note:**

- TXT value should not be encoded for checksum computation.
- Leading or trailing spaces should be trimmed from the TXT value

- Optional parameters are to be omitted when computing the checksum if missing, but included if present.

## Checksum Generation PHP Code Example:

```
//Set optional values to empty if not given.
$ext   = isset($_POST['EXT']) &&
in_array(trim(strtolower($_POST['EXT'])), array('mp3','swf')) ?
trim(strtolower($_POST['EXT'])) : '';

$fxType = isset($_POST['FX_TYPE']) && strlen($_POST['FX_TYPE']) > 0 ?
$_POST['FX_TYPE'] : '';

$fxLevel= isset($_POST['FX_LEVEL']) && strlen($_POST['FX_LEVEL']) > 0 ?
$_POST['FX_LEVEL'] : '';

$httpErr= isset($_POST['HTTP_ERR']) && strlen($_POST['HTTP_ERR']) > 0 ?
$_POST['HTTP_ERR'] : '';

//Construct parameters.
$get = 'EID='.$_POST['EID']
       .'&LID='.$_POST['LID']
       .'&VID='.$_POST['VID']
       .'&TXT='.urlencode($_POST['TXT'])
       .'&EXT='.$ext
       .'&FX_TYPE='.$fxType
       .'&FX_LEVEL='.$fxLevel
       .'&ACC='.$_POST['ACC']
       .'&API='.$_POST['API']
       .'&SESSION='.$_POST['SESSION']
       .'&HTTP_ERR='.$httpErr;

//Construct checksum
$CS = md5($_POST['EID'].$_POST['LID'].$_POST['VID'].$_POST['TXT']. $ext.
$fxType.$fxLevel.$_POST['ACC']. $_POST['API'].$_POST['SESSION'].
$httpErr.$_POST['SECRET']);

//Construct full URL
$url = 'http://www.vocalware.com/tts/gen.php?' . $get . '&CS=' . $CS;
```

## Appendix A: Languages and Voices

The following tables list Engine IDs, Language IDs and Voice IDs available for use with the Vocalware API.

| *Language* | *ID* |
|---|---|
| **Arabic** | 27 |
| **Basque** | 22 |
| **Catalan** | 5 |
| **Chinese** | 10 |
| **Czech** | 18 |
| **Danish** | 19 |
| **Dutch** | 11 |
| **English** | 1 |
| **Esperanto** | 31 |
| **Filipino** | 32 |
| **Finnish** | 23 |
| **French** | 4 |
| **Galician** | 15 |
| **German** | 3 |
| **Greek** | 8 |
| **Hindi** | 24 |
| **Hungarian** | 29 |
| **Indonesian** | 28 |
| **Italian** | 7 |
| **Japanese** | 12 |
| **Korean** | 13 |
| **Norwegian** | 20 |
| **Polish** | 14 |
| **Portuguese** | 6 |
| **Romanian** | 30 |
| **Russian** | 21 |
| **Slovak** | 37 |
| **Spanish** | 2 |
| **Swedish** | 9 |
| **Thai** | 26 |
| **Turkish** | 16 |
| **Ukrainian** | 40 |
| **Vietnamese** | 41 |

## Engine ID = 2

| Language | Lang. ID | Voice Name | Voice ID | Gender | Description |
|---|---|---|---|---|---|
| **English** | 1 | Susan | 1 | F | US |
| **English** | 1 | Dave | 2 | M | US |
| **English** | 1 | Elizabeth | 4 | F | UK |
| **English** | 1 | Simon | 5 | M | UK |
| **English** | 1 | Catherine | 6 | F | UK |
| **English** | 1 | Allison | 7 | F | US |
| **English** | 1 | Steven | 8 | M | US |
| **English** | 1 | Alan | 9 | M | Australian |
| **English** | 1 | Grace | 10 | F | Australian |
| **English** | 1 | Veena | 11 | F | Indian |
| **Spanish** | 2 | Carmen | 1 | F | Castilian |
| **Spanish** | 2 | Juan | 2 | M | Castilian |
| **Spanish** | 2 | Francisca | 3 | F | Chilean |
| **Spanish** | 2 | Diego | 4 | M | Argentine |
| **Spanish** | 2 | Esperanza | 5 | F | Mexican |
| **Spanish** | 2 | Jorge | 6 | M | Castilian |
| **Spanish** | 2 | Carlos | 7 | M | American |
| **Spanish** | 2 | Soledad | 8 | F | American |
| **Spanish** | 2 | Leonor | 9 | F | Castilian |
| **Spanish** | 2 | Ximena | 10 | F | American |
| **German** | 3 | Stefan | 2 | M | |
| **German** | 3 | Katrin | 3 | F | |
| **French** | 4 | Bernard | 2 | M | European |
| **French** | 4 | Jolie | 3 | F | European |
| **French** | 4 | Florence | 4 | F | European |
| **French** | 4 | Charlotte | 5 | F | Canadian |
| **French** | 4 | Olivier | 6 | M | Canadian |
| **Catalan** | 5 | Montserrat | 1 | F | |
| **Catalan** | 5 | Jordi | 2 | M | |
| **Catalan** | 5 | Empar | 3 | F | Valencian |
| **Portuguese** | 6 | Amalia | 2 | F | European |
| **Portuguese** | 6 | Eusebio | 3 | M | European |

| | | | | | |
|---|---|---|---|---|---|
| **Italian** | 7 | Paola | 1 | F | |
| **Italian** | 7 | Silvana | 2 | F | |
| **Italian** | 7 | Valentina | 3 | F | |
| **Italian** | 7 | Luca | 5 | M | |
| **Italian** | 7 | Marcello | 6 | M | |
| **Italian** | 7 | Roberto | 7 | M | |
| **Italian** | 7 | Matteo | 8 | M | |
| **Italian** | 7 | Giulia | 9 | F | |
| **Italian** | 7 | Federica | 10 | F | |
| **Greek** | 8 | Afroditi | 1 | F | |
| **Greek** | 8 | Nikos | 3 | M | |
| **Swedish** | 9 | Annika | 1 | F | |
| **Swedish** | 9 | Sven | 2 | M | |
| **Chinese** | 10 | Linlin | 1 | F | Mandarin |
| **Chinese** | 10 | Lisheng | 2 | F | Mandarin |
| **Dutch** | 11 | Willem | 1 | M | |
| **Dutch** | 11 | Saskia | 2 | F | |
| **Polish** | 14 | Zosia | 1 | F | |
| **Polish** | 14 | Krzysztof | 2 | M | |
| **Galician** | 15 | Carmela | 1 | F | |
| **Turkish** | 16 | Kerem | 1 | M | |
| **Turkish** | 16 | Zeynep | 2 | F | |
| **Turkish** | 16 | Selin | 3 | F | |
| **Danish** | 19 | Frida | 1 | F | |
| **Danish** | 19 | Magnus | 2 | M | |
| **Norwegian** | 20 | Vilde | 1 | F | |
| **Norwegian** | 20 | Henrik | 2 | M | |
| **Russian** | 21 | Olga | 1 | F | |
| **Russian** | 21 | Dmitri | 2 | M | |
| **Finnish** | 23 | Milla | 1 | F | |
| **Finnish** | 23 | Marko | 2 | M | |
| **Arabic** | 27 | Tarik | 1 | M | |
| **Arabic** | 27 | Laila | 2 | F | |
| **Romanian** | 30 | Ioana | 1 | F | |
| **Esperanto** | 31 | Ludoviko | 1 | M | |

## Engine ID = 4

| Language | Lang. ID | Voice Name | Voice ID | Gender | Description |
|---|---|---|---|---|---|
| **English** | 1 | Jill | 2 | F | US |
| **English** | 1 | Tom | 3 | M | US |
| **English** | 1 | Karen | 4 | F | Australian |
| **English** | 1 | Daniel | 5 | M | UK |
| **English** | 1 | Serena | 7 | F | UK |
| **English** | 1 | Moira | 8 | F | Irish |
| **English** | 1 | Sangeeta | 9 | F | Indian |
| **English** | 1 | Lee | 10 | M | Australian |
| **English** | 1 | Samantha | 11 | F | US |
| **English** | 1 | Fiona | 12 | F | Scottish |
| **English** | 1 | Tessa | 13 | F | South African |
| **Spanish** | 2 | Duardo | 1 | M | |
| **Spanish** | 2 | Monica | 3 | F | |
| **Spanish** | 2 | Paulina | 4 | F | Mexican |
| **Spanish** | 2 | Javier | 5 | M | Mexican |
| **German** | 3 | Steffi | 1 | F | |
| **German** | 3 | Yannick | 2 | M | |
| **German** | 3 | Anna | 3 | F | |
| **French** | 4 | Felix | 1 | M | Canadian |
| **French** | 4 | Julie | 2 | F | Canadian |
| **French** | 4 | Sebastien | 3 | M | European |
| **French** | 4 | Virginie | 4 | F | European |
| **French** | 4 | Thomas | 5 | M | European |
| **Catalan** | 5 | Nuria | 1 | F | |
| **Portuguese** | 6 | Raquel | 2 | F | Brazilian |
| **Portuguese** | 6 | Joana | 3 | F | European |
| **Italian** | 7 | Paolo | 1 | M | |
| **Italian** | 7 | Silvia | 2 | F | |
| **Greek** | 8 | Alexandros | 1 | M | |
| **Swedish** | 9 | Alva | 1 | M | |
| **Swedish** | 9 | Oskar | 3 | M | |
| **Chinese** | 10 | Sin-Ji | 1 | F | Cantonese |
| **Chinese** | 10 | Ya-Ling | 2 | F | Taiwanese Mandarin |
| **Chinese** | 10 | Ting-Ting | 4 | F | Mandarin |
| **Dutch** | 11 | Ellen | 1 | F | Belgian |

| | | | | | |
|---|---|---|---|---|---|
| **Dutch** | 11 | Clair | 2 | F | |
| **Dutch** | 11 | Xander | 4 | M | |
| **Japanese** | 12 | Kyoko | 1 | F | |
| **Korean** | 13 | Narae | 1 | F | |
| **Polish** | 14 | Agata | 1 | F | |
| **Turkish** | 16 | Aylin | 1 | F | |
| **Czech** | 18 | Zuzana | 1 | F | |
| **Danish** | 19 | Ida | 1 | F | |
| **Norwegian** | 20 | Stine | 2 | F | |
| **Russian** | 21 | Milena | 2 | F | |
| **Basque** | 22 | Arantxa | 1 | F | |
| **Finnish** | 23 | Mikko | 1 | M | |
| **Hindi** | 24 | Lekha | 1 | F | |
| **Thai** | 26 | Narisa | 1 | F | |
| **Arabic** | 27 | Maged | 1 | M | |
| **Indonesian** | 28 | Damayanti | 1 | F | |
| **Hungarian** | 29 | Eszter | 1 | F | |
| **Romanian** | 30 | Simona | 1 | F | |
| **Slovak** | 37 | Nadeja | 3 | F | |

## TTS Engine ID = 6  (Offbeat Voices)

| *Language* | *Lang. ID* | *Voice Name* | *Voice ID* | *Gender* | *Description* |
|---|---|---|---|---|---|
| **English** | 1 | Bigdude | 1 | M | |
| **English** | 1 | Giant | 2 | M | |
| **English** | 1 | Male | 3 | M | |
| **English** | 1 | Female | 4 | F | |
| **English** | 1 | Wee One | 5 | M | Child |
| **English** | 1 | Old Woman | 6 | F | |
| **English** | 1 | Robotoid | 7 | M | |
| **English** | 1 | Martian | 8 | M | |
| **English** | 1 | Munchkin | 9 | M | |
| **English** | 1 | Colossus | 10 | M | |
| **English** | 1 | Mellow Yellow I | 11 | F | |
| **English** | 1 | Mellow Yellow II | 12 | M | |
| **English** | 1 | Crisper | 13 | M | |

| | | | | |
|---|---|---|---|---|
| **English** | 1 | Fast Fred | 15 | M |
| **English** | 1 | Troll | 16 | M |
| **English** | 1 | Nerd | 17 | M |
| **English** | 1 | Milk Toast | 18 | M |
| **English** | 1 | Tipsy | 19 | M |
| **English** | 1 | Choirboy I | 20 | M |
| **English** | 1 | Choirboy II | 21 | M |

## Engine ID = 7

| *Language* | *Lang. ID* | *Voice Name* | *Voice ID* | *Gender* | *Description* |
|---|---|---|---|---|---|
| **English** | 1 | Olivia | 1 | F | UK |
| **English** | 1 | Oliver | 2 | M | UK |
| **English** | 1 | Matilda | 3 | F | Australian |
| **English** | 1 | Lakshmi | 5 | F | Indian |
| **English** | 1 | Prashant | 6 | M | Indian |
| **English** | 1 | Brenda | 7 | F | US |
| **German** | 3 | Hilda | 1 | F | |
| **German** | 3 | Heinz | 2 | M | |
| **French** | 4 | Beatrice | 1 | F | |
| **French** | 4 | Antoine | 2 | M | |
| **French** | 4 | Leonie | 3 | F | Canadian |
| **French** | 4 | Gaspard | 4 | M | Canadian |
| **Portuguese** | 6 | Ana | 1 | F | Brasilian |
| **Portuguese** | 6 | Leonor | 3 | F | |
| **Portuguese** | 6 | Tiago | 4 | M | |
| **Italian** | 7 | Bianca | 1 | F | |
| **Italian** | 7 | Alessandro | 2 | M | |
| **Greek** | 8 | Eleni | 1 | F | |
| **Greek** | 8 | Giorgos | 2 | M | |
| **Swedish** | 9 | Astrid | 1 | F | |
| **Swedish** | 9 | Gustav | 2 | M | |
| **Chinese** | 10 | Chia-ling | 1 | F | Taiwanese |
| **Chinese** | 10 | Chia-hao | 2 | M | Taiwanese |
| **Chinese** | 10 | Yan | 3 | F | HK Cantonese |
| **Chinese** | 10 | Chan | 4 | M | HK Cantonese |
| **Dutch** | 11 | Famke | 1 | F | |

| | | | | | |
|---|---|---|---|---|---|
| **Dutch** | 11 | Dirk | 2 | M | |
| **Japanese** | 12 | Himari | 1 | F | |
| **Japanese** | 12 | Kaito | 2 | M | |
| **Polish** | 14 | Danota | 1 | F | |
| **Polish** | 14 | Wojciech | 2 | M | |
| **Turkish** | 16 | Zehra | 1 | F | |
| **Turkish** | 16 | Eymen | 2 | M | |
| **Czech** | 18 | Pavla | 1 | F | |
| **Danish** | 19 | Dagny | 1 | F | |
| **Danish** | 19 | Erik | 2 | M | |
| **Norwegian** | 20 | Dagrun | 1 | F | |
| **Norwegian** | 20 | Lars | 2 | M | |
| **Finnish** | 23 | Sanna | 1 | F | |
| **Hindi** | 24 | Swathi | 1 | F | |
| **Hindi** | 24 | Karan | 2 | M | |
| **Arabic** | 27 | Amina | 1 | F | |
| **Arabic** | 27 | Jamal | 2 | M | |
| **Indonesian** | 28 | Putri | 1 | F | |
| **Indonesian** | 28 | Bintang | 2 | M | |
| **Hungarian** | 29 | Flora | 1 | F | |
| **Hungarian** | 29 | Laszlo | 2 | M | ** |
| **Filipino** | 32 | Mayumi | 1 | F | |
| **Filipino** | 32 | Datu | 2 | M | |
| **Slovak** | 37 | Eliska | 1 | F | |
| **Ukrainian** | 40 | Vira | 1 | F | |
| **Vietnamese** | 41 | Nguyet | 1 | F | |
| **Vietnamese** | 41 | Phuong | 2 | M | |

## Appendix B: Error Handling & Codes

Two types of error codes are used:

1. "In Stream" error codes are always returned in case of failure. The error code begins with the string "Error".
2. *(HTTP REST API only)* "HTTP Header" status codes are always returned. If normal completion, "200" is returned.
   If the http header status code is different than 200, check the returned data for the In-Stream code and message for additional information.

The following error codes are used (*Note: some of the listed error codes are only relevant to the HTTP REST API):*

| **In-Stream Error Codes (always returned in case of error)** | | |
|---|---|---|
| **Error Code** | **Message** | **More Info** |
| 100 | No data found in request. | Missing all request parameters. |
| 101 | Missing Required Parameter | Missing EID |
| 102 | Missing Required Parameter | Missing LID |
| 103 | Missing Required Parameter | Missing VID |
| 104 | Missing Required Parameter | Missing TXT |
| 105 | Missing Required Parameter | Missing ACC |
| 106 | Missing Required Parameter | Missing API |
| 107 | Missing Required Parameter | Missing CS |
| 108 | Missing Required Parameter | Missing APIName |
| 109 | Missing Required Parameter | Missing APIType |
| 201 | Unknown account id | ACC failed verification |
| 202 | Invalid session | SESSION failed verification |
| 203 | Invalid checksum | Checksum failure |
| 204 | Authorization failure | Verification failed (General) |
| 205 | Inactive account | Inactive account |
| 206 | Invalid API | API ID not assigned to account |
| 300 | General error | General error |
| 301 | Too many TTS Requests | Too many TTS Requests |
| 302 | TTS Failed | TTS Failed |
| 303 | Too many errors | Detailed explanation below* |
| 304 | Web Server Error | Error description may vary |
| 400 | APS Failed | Unspecified server processing error |

| **HTTP Header Status Codes** *(HTTP REST API only)* | |
|---|---|
| **Error Code** | **Description** |
| 200 | Successful TTS request |
| 400 | Bad (malformed) request. Modify the request before re- |

| | |
|---|---|
| | submitting. |
| 401 | Unauthorized request. |
| 503 | The server is temporarily unable to fulfill the request. OK to re-submit. |

* Too many errors (Error Code 303)

Certain invalid inputs are impossible to process and will generate failure. Examples include providing input text in a language other than the specified language (e.g. Japanese input for an English voice), or providing long meaningless strings of characters (e.g. "aaaaaaaaaaaaaaaaaaaaaaaa" ).

Individual requests of this type may fail with (In-Stream) error code 400 or another code, and do not affect service. But if an account generates an ongoing torrent of invalid input, the account will be automatically suspended. Suspension will continue for 30 min at a time, until the problem is corrected. While suspended, both valid and invalid inputs will be blocked, and error code 303 returned.

If you receive this error, check your application and examine the syntax of your requests. A typical error to look for (as stated above) would be providing an incorrect language ID for the intended input language. Once the problem is corrected on your end, service should be automatically restored within 30 minutes.