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, Flash widget, 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 three flavors: JavaScript/HTML5, ActionScript (Flash) 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 20 languages, with several voices available in most. The API allows you to specify the language and voice to use, as well as optional audio effects such as pitch, echo, etc.

Select the API Flavor to Use

The API comes in three flavors:

1. JavaScript/HTML5 - also referred to as the JavaScript API
2. ActionScript - both AS2 and AS3 are supported
3. 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 and ActionScript APIs are similar, and are both 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 Flash app - use the ActionScript API
- 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 either JS or AS APIs. 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, including full source code, covering all three APIs and each of the API functions listed here.

The JavaScript & ActionScript APIs

Introduction

The API function calls for JavaScript and ActionScript are identical in syntax and functionality. Both JavaScript and ActionScript APIs operate by way of an invisible client side code object ("Agent"), that your web page or Flash application 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 or ActionScript 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 in most cases. However, if you need low level access to the audio stream data, the API allows ActionScript data direct access through the "vw_audioStarted" callback (available only in AS3).

To get started with either JS or AS APIs you need to:

a. Create an AS or 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 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 loading confirmation. **API functions work only after the API has completed loading.**

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 transparently on 'desktop' as well as mobile browsers (the word 'desktop' is used throughout to refer to non-mobile client side environments, which include desktop and laptop computers of all types). This means that you should not need to 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 one exception will function in the same way within mobile browsers as it will on desktop browsers. The only 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.

Another important difference to note is that on some mobile browsers (notably Safari on iOS), the first call to the API must be user driven (i.e. user clicks on a button). This limitation prevents the page from automatically speaking to the viewer unprompted. Trying to do so will not cause an error – but will simply not work.

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".

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

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.

ActionScript 3 Instructions

1. Run Adobe Flash CS3 or higher.
2. Click Layer 1
3. If you do not see the Actions > Frame window label in the middle left of the screen, click Actions.
4. Add the following block of code into the Actions Frame window.

```
Security.allowDomain("content.oddcast.com");
var ldr:Loader;
var req:URLRequest;

var vw_player:MovieClip;
var _example_ui:MovieClip;

req = new URLRequest("EMBED_CODE");
ldr = new Loader();
ldr.contentLoaderInfo.addEventListener(Event.COMPLETE,completeHan
dler);
ldr.load(req);
addChild(ldr);

function completeHandler($ev:Event):void
{
   trace("EXAMPLE --- COMPLETE HANDLER "+$ev.target);
   vw_player = MovieClip(ldr.content);
}
```

5. Copy your embed code and Paste where you see EMBED_CODE above.

Note: In the HTML file make sure that allowScriptAccess' is set to 'always'.

6. Declare event listeners in the completeHandler function if call back functions will be used, for example:

```
vw_player.addEventListener("vw_apiLoaded", vw_apiLoaded);
```

ActionScript 2 Instructions

1. Run Adobe Flash MX or higher.
2. From the top level menu, choose Insert->New Symbol->Ok
3. Click on Scene 1
4. The Scene window appears as a blank white rectangle in the center of the Adobe Flash screen.

5. Drag Symbol 1 from the Library window to the upper left corner of the Scene window. Replace with instance_name.
6. Click Layer 1
7. If you do not see the Actions > Frame window label in the middle left of the screen, click Actions.
8. Copy your embed code and Paste into the Actions Frame window.
9. The line below should appear in the Actions for the first Frame of the first Scene of your movie:
```
System.security.allowDomain("vhost.oddcast.com", "vhss-a.oddcast.com", "vhss-c.oddcast.com", "vhss-d.oddcast.com");
```
10. The line below should appear in the Actions for the Scene. Replace instance_name with the name of the instance name you entered for the symbol you inserted, and replace EMBED_CODE with your embed code.
```
instance_name.loadMovie("EMBED_CODE");
```
11. Declare all callback functions that you wish to use, for example:
```
function vw_apiLoaded(){
        //any commands that should be triggered here;
}
```

Playback Control Functions

All functions are available to both JS and AS APIs. The syntax of the functions is the same in both.

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 900 characters. The exceptions are Chinese & Japanese which are limited to 225 characters. A longer text string will be truncated.
`voice`	Required. Integer – Voice ID, as listed in Appendix.
`lang`	Required. Integer – Language ID, as listed in Appendix.
`engine`	Required. Integer – Voice Family ID. See languages and voices listed listed in Appendix.
`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',1,1,1)
sayText('Hello World',1,1,1,'S',-2)
```

---

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

Callback Functions enable coordination between playback and your page or application.

Callback functions are supported in both Flash movies (ActionScript) and HTML pages (JavaScript). The syntax of the functions is the same, though the method of setting them up is different - please see below.

Embedding in an 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.

Embedding in a Flash movie:

ActionScript 2

Events during playback trigger calls to specific ActionScript functions in your movie, if such functions exist. To take advantage of these calls you must **add the callback functions within your movie at the _parent level**. Note that you do not need to add callback functions which you do not intend to use.

ActionScript 3

To receive the status callbacks you need to register an event listener for each callback function. Here's an example of loading the content and registering as a listener for the "vw_talkStarted" event:

```
loader:Loader = new Loader();
```

```
loader.loaderContentInfo.addEventListener(Event.COMPLETE,
setListeners);
loader.load( /* your AS3 embed code here */ );
function setListeners():void
{
MovieClip(loader.content).addEventListener("vw_talkStarted",talkS
tartedHandler);
function talkStartedHandler():void{ trace("talk started"); }
}
```

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 - JavaScript & ActionScript2

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

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vw_apiLoaded",ap
iLoadedHandler);
function apiLoadedHandler(event:*):void{
     trace("api loaded. Id="+ event.data);
}
```

---

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 - JavaScript & ActionScript2

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

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vw_audioProgress",audi
oProgHandler);
function audioProgHandler(event:*):void{
      trace("percent played: "+ event.data.percent);
}
```

---

vw_talkStarted ()

Triggered when the audio playback begins.

Example - JavaScript & ActionScript2

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

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vw_talkStarted",talkS
tartedHandler);
function talkStartedHandler(event:*):void{
      trace("talk started");
}
```

---

vw_talkEnded ()

Triggered when audio playback is done.

Example - JavaScript & ActionScript2

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

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vw_talkEnded",talkEnde
dHandler);
function talkEndedHandler ():void{
      trace("talk ended");
}
```

---

vw_audioStarted ()

Triggered when audio playback begins. Unlike vw_talkStarted() this event is fired for each audio playback in a sequence. In ActionScript3, the event contains a "data" property which provides direct references to the Sound object (event.data.sound) and the SoundChannel (event.data.sound_channel) to allow advanced control for as3 developers.

Example - JavaScript & ActionScript2

```
function vw_audioStarted(){
}
```

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vw_audioStarted",audio
StartedHandler);

function audioStartedHandler(event:*):void{
      var sound:Sound = event.data.sound;
      var sound_channel:SoundChannel = event.data.sound_channel;
      trace("audio started");
}
```

---

vw_audioEnded ()

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

Example - JavaScript & ActionScript2

```
function vw_audioEnded(){
}
```

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vw_audioEnded",audEnde
dHandler);
function audEndedHandler ():void{
      trace("audio ended");
}
```

The HTTP REST API

The HTTP GEN Request

This HTTP request supports both GET & 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=50cdbf7c4e5d117dcb2efff424e10054&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 (Encoded)
EXT	swf or mp3. 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	Optionally use HTTP header status codes to return success or error. Values: 0 – do not use HTTP codes (default) 1 – use HTTP codes
CS	Checksum – implemented as an md5 of all above parameter and secret word

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 swf binary stream is returned. The audio data is single channel (mono), has a 22Khz sample rate and is encoded at a 48Kbps bitrate.

In case of failure, two types of error codes are used:

1. “In Stream” error codes are always returned in case of failure. The Code is returned within the audio binary data, and begins with the string “Error”.
2. “HTTP Header” status codes are only returned if HTTP_ERR parameter is set to 1. Otherwise, code “200” will always be returned. By default, this parameter is set to 0 (for backward compatibility).

The following are the error codes returned for each type:

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
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
400	APS Failed	APS Failed

HTTP Header Status Codes (returned only if requested)	
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.

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

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;
```

Audio Timing Meta-Data

To coordinate audio playback within your application (i.e. to display captions etc.) you may want to take advantage of timing data Vocalware stores in an ID3 tag. The tag includes multiple bits of information – the following explanation will focus on the text and timing.

In the ID3 tag look for "timed_phonemes" then look for the letter "W" (uppercase). each W denotes a "Word" and is followed by four comma separated symbols:
1. start time (in miliseconds)
2. end time
3. amplitude
4. the text

Example: An audio generated from the text "one two three test"

```
audio_duration = "1.708";
date = "20130827_15:10:40.088";
host = "ODDAPS003";
kbps = "48";
khz = "22050";
lip_string =
"f0=1&f1=5&f2=7&f3=6&f4=1&f5=1&f6=12&f7=13&f8=13&f9=6&f10=15&f11=15&f12=1
5&f13=9&f14=9&f15=6&f16=6&f17=0&f18=0&f19=0&f20=0&f21=0&nofudge=1&lipvers
ion=2&ok=1";
timed_phonemes = "P,0,46,51,x    S,46,1336,71,.    G,46,256,78,8
W,46,256,78,one    P,46,106,86,w    P,106,186,95,^    P,186,256,54,n
G,256,476,71,8    W,256,476,71,two    P,256,336,67,t    P,336,476,74,U
G,476,736,77,8    W,476,736,77,three    P,476,546,66,T
P,546,576,87,R    P,576,736,80,E    W,736,1336,64,test
P,736,796,63,t    P,796,1026,90,e    P,1026,1196,64,s
P,1196,1336,24,t    P,1336,1686,0,x";
```

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
Catalan	5
Chinese	10
Danish	19
Dutch	11
English	1
Esperanto	31
Finnish	23
French	4
Galician	15
German	3
Greek	8
Italian	7
Japanese	12
Korean	13
Norwegian	20
Polish	14
Portuguese	6
Romanian	30
Russian	21
Spanish	2
Swedish	9
Turkish	16

Engine ID = 2

Language	*Lang. ID*	*Voice Name*	*Voice ID*	*Gender*	*Description*	*Expressive Cues**
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	Gabriela	1	F	Brasilian	√
Portuguese	6	Amalia	2	F	European	√
Portuguese	6	Eusebio	3	M	European	√
Portuguese	6	Fernanda	4	F	Brazilian	√
Portuguese	6	Felipe	5	M	Brazilian	√
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		√

* *Expressive Cues* are a set of special tags which you may use in your text to specify distinct non-verbal expressions, such as laughing, crying, sighing, coughing, etc. Expressive Cues can be used only with a subset of voices, as indicated above. For a complete list of Expressive Cue tags see separate documentation.

Engine ID = 3

Language	*Lang. ID*	*Voice Name*	*Voice ID*	*Gender*	*Description*
English	1	Kate	1	F	US
English	1	Paul	2	M	US
English	1	Julie	3	F	US
English	1	Bridget	4	F	UK
English	1	Hugh	5	M	UK
English	1	Ashley	6	F	US
English	1	James	7	M	US

Spanish	2	Violeta	1	F	Mexican
Spanish	2	Francisco	2	M	Mexican
French	4	Chloe	1	F	Canadian
Chinese	10	Lily	1	F	Mandarin
Chinese	10	Hui	3	F	Mandarin
Chinese	10	Liang	4	M	Mandarin
Japanese	12	Show	2	M	
Japanese	12	Misaki	3	F	
Korean	13	Yumi	1	F	
Korean	13	Junwoo	2	M	

Appendix B: Expressive Cues

Expressive Cues are a set of special tags which you may use in your text to specify distinct non-verbal expressions, such as laughing, crying, sighing, coughing, etc. Expressive Cues can only be used with a subset of our voices, as indicated in Appendix A.

Expressive Cues tags are to be placed directly in your text. Example - clearing throat sound:

_Throat_01 you may want to consider checking out our specials!

NOTE: You must prepend all Expressive Cues with a '\' character before using them in API functions. For example:

sayText("Hello World _Laugh",5,1,2);

Following is a list of Expressive Cue tags supported by our voices.

* The list of voices is incomplete – additional details to be added as they become available. For now, if using a voice not on the list below, but that supports Expressive Cues, please try to use the cues listed below for voices of the same language, as there is a large overlap.

Dutch: Saskia

_Ah _Ah_01 \Ah_02 \Ah_03
_Aha _Aha_01 _Aha_02
_Bleah _Bleah_01 _Bleah_02
_Breath _Breath_01
_Click
_Cough _Cough_01
_Eh _Eh_01 _Eh_02
_Ehm _Ehm_01 _Ehm_02
_He _He_01 _He_02
_Heh _Heh_01
_Hm _Hm_01
_Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04
_Mmm _Mmm_01 _Mmm_02
_Nah
_Oef _Oef_01
_Oei
_Oeps
_Oh _Oh_01 _Oh_02 _Oh_03
_Oho
_Oohw _Oohw_01 _Oohw_02
_Prrf _Prrf_01 _Prrf_02

_Sniff _Sniff_01 _Sniff_02
_Swallow
_Throat _Throat_01 _Throat_02
_Tss _Tss_01
_Uh _Uh_01 _Uh_02
_Whistle _Whistle_01
_Wow _Wow_01
_Yawn _Yawn_01

Dutch: Willem

_Ah _Ah_01
_Aha _Aha_01 _Aha_02
_Bleah _Bleah_01 _Bleah_02
_Breath _Breath_01 _Breath_02 _Breath_03
_Click
_Cough _Cough_01 _Cough_02
_Eh _Eh_01
_Ehm _Ehm_01
_He _He_01 _He_02
_Heh _Heh_01
_Hm _Hm_01
_Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04 _Laugh_05
_Mmm _Mmm_01 _Mmm_02
_Oef
_Oeps
_Oh _Oh_01 _Oh_02
_Oho _Oho_01
_Smack _Smack_01
_Sniff _Sniff_01
_Swallow _Swallow_01
_Throat _Throat_01
_Tss _Tss_01
_Uh _Uh_01
_Whistle _Whistle_01 _Whistle_02
_Wow _Wow_01
_Yawn

English (UK): Simon

_Ah _Ah_01 _Ah_02
_Aha
_Click _Click_01 _Click_02
_Cough _Cough_01 _Cough_02
_Cry _Cry_01
_Doh

_Duh
_Eh
_Eugh
_Hiccup _Hiccup_01 _Hiccup_02
_Hm _Hm_01
_Hurrah
_Kiss _Kiss_01 _Kiss_02 _Kiss_03
_Laugh _Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04 _Laugh_05
_Mmm _Mmm_01
_Oh _Oh_01 _Oh_02
_Oho
_Ooh _Ooh_01
_Oops
_Pain _Pain_01 _Pain_02
_Phoarr
_Raspberry _Raspberry_01 _Raspberry_02
_Sigh _Sigh_01
_Sneeze _Sneeze_01 _Sneeze_02
_Sniff _Sniff_01 _Sniff_02
_Snore _Snore_01
_Sshhh
_Swallow
_Throat _Throat_01 _Throat_02 _Throat_03
_Tuttut
_Uh _Uh_01
_Uhuh _Uhuh_01 _Uhuh_02 _Uhuh_03
_Um
_Whistle _Whistle_01 _Whistle_02 _Whistle_03 _Whistle_04 _Whistle_05
_Woh _Woh_01 _Woh_02
_Wow
_Yawn _Yawn_01
_Yuck
_Yummy

French: Jolie

_Aaah
_Aah
_Ah
_Aie _Aie_01
_Bah
_Ben
_Berk
_Bleah
_Bof
_Breath _Breath_01 _Breath_02 _Breath_03 _Breath_04

_Chut _Chut_01
_Click _Click_01 _Click_02
_Cough _Cough_01 _Cough_02
_Ehe
_Ehi
_Ehm
_Euhh
_Heho _Heho_01
_Hep _Hep_01
_HmHm
_HuHu
_Hum
_Laugh _Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04 _Laugh_05 _Laugh_06 _Laugh_07 _Laugh_08 _Laugh_09
_Mmm
_Mmum
_Oh
_Oho _Oho_01
_Ooh
_Ops
_Ouf
_Pfuit
_Prrr
_Ptt_01 _Ptt_02
_Rrrr
_Smack _Smack_01 _Smack_02 _Smack_03 _Smack_04
_Sniff _Sniff_01 _Sniff_02 _Sniff_03 _Sniff_04
_Swallow _Swallow_01 _Swallow_02
_TChut
_Throat _Throat_01
_Toh
_Tt
_Tttt
_Uff
_Uh _Uh_01
_Wao _Wao_01
_Whistle _Whistle_01 _Whistle_02 _Whistle_03 _Whistle_04 _Whistle_05 _Whistle_06 _Whistle_07
_Wuuh
_Yawn _Yawn_01 _Yawn_02 _Yawn_03 _Yawn_04

German: Katrin

_Ah _Aha
_Aha
_Bleah _Bleah_01 _Bleah_02 _Bleah_03 _Bleah_04

_Breath _Breath_01 _Breath_02 _Breath_03
_Click _Click_01 _Click_02 _Click_03 _Click_04 _Click_05
_Cough _Cough_01 _Cough_02
_Eh _Ehm _Ehm_01
_Ehm _Ehm_01
_Hey _Hey_01
_Kiss _Kiss_01 _Kiss_02
_Laugh _Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04 _Laugh_05
_Mhm _Mhm_01 _Mhm_02
_Mmm _Mmm_01 _Mmm_02 _Mmm_03 _Mmm_04
_Oh _Oh_01 _Oh_02 _Oh_03 _Oh_05
_Pff _Pff_01
_Puh _Puh_01
_Sniff _Sniff_01 _Sniff_02
_Swallow _Swallow_01 _Swallow_02
_Throat
_Tss _Tss_01
_Ups _Ups_01
_Whistle _Whistle_01
_Wow _Wow_01
_Yawn _Yawn_01 _Yawn_02

German: Stefan

_Ah _Ah_01 _Aha _Aha_01 _Aha_02 _Aha_04 _Aha_05 _Ahja _Ahja_01 _Ahja_02
_Aha _Aha_01 _Aha_02 _Aha_04 _Aha_05
_Ahja _Ahja_01 _Ahja_02
_Bleah _Bleah_01 _Bleah_02 _Bleah_03 _Bleah_04
_Breath _Breath_01 _Breath_02 _Breath_03 _Breath_04 _Breath_05 _Breath_06 _Breath_07 _Breath_08 _Breath_09
_Click _Click_01 _Click_02 _Click_03 _Click_04 _Click_05
_Cough _Cough_01 _Cough_02 _Cough_03 _Cough_04 _Cough_05 _Cough_06 _Cough_07
_Eh _Eh_01 _Eh_02 _Eh_03 _Eh_04 _Eh_05
_Ehm _Ehm_01 _Ehm_02 _Ehm_03 _Ehm_04 _Ehm_05 _Ehm_06 _Ehm_07 _Ehm_08
_He _Hee
_Hee
_Laugh _Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04
_Mhm _Mhm_01 _Mhm_02 _Mhm_03 _Mhm_04 _Mhm_05 _Mhm_06 _Mhm_07
_Mmm _Mmm_01
_Oh _Oho
_Prrr _Prrr_01
_Puffpant
_Sniff _Sniff_01 _Sniff_02 _Sniff_03 _Sniff_04 _Sniff_05 _Sniff_06 _Sniff_07
_Swallow _Swallow_01 _Swallow_02

_Toh
_Uuu
_Whistle _Whistle_01 _Whistle_02 _Whistle_03 _Whistle_04
_Wow _Wow_1 _Wow_2
_Yawn _Yawn_01

Italian: Giulia

_Ah_01 _Ah_02 _Ah_03 _Aha _Ahahah
_Aha _Ahahah
_Ahahah
_Bah
_Breath _Breath_01 _Breath_02 _Breath_03 _Breath_04
_Click _Click_01 _Click_02
_Cough _Cough_01 _Cough_02 _Cough_03 _Cough_04
_Di'
_Eh _Ehe
_Laugh _Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04
_Mah
_Mhm _Mhm_01 _Mhm_02 _Mhm_03 _Mhm_04
_Mmm _Mmm_01 _Mmm_02 _Mmm_03
_Oh _Oho
_Smack _Smack_01 _Smack_02 _Smack_03 _Smack_04
_Swallow
_Throat _Throat_01 _Throat_02 _Throat_03
_Toh

Italian: Luca

_Aagh
_Acci _Acci_01
_Ah _Ah_01
_Arf _Arf_01
_Argh _Argh_01 _Argh_02 _Argh_03 _Argh_04
_Azz _Azz_01
_Bah _Bah_01 _Bah_02 _Bah_03
_Bau _Bau_01 _Bau_02 _Bau_03
_Beh _Beh_01 _Beh_02 _Beh_03 _Beh_04 _Beh_05 _Beh_06 _Beh_07 _Beh_08 _Beh_09
_Bleah _Bleah_01 _Bleah_02
_Boh _Boh_01 _Boh_02
_Breath _Breath_01 _Breath_02 _Breath_03 _Breath_04 _Breath_05 _Breath_06 _Breath_07 _Breath_08 _Breath_09 _Breath_10 _Breath_11 _Breath_12
_Buh _Buh_01
_Buuu _Buuu_01 _Buuu_02
_Click _Click_01 _Click_02 _Click_03 _Click_04 _Click_05 _Click_06

_Cough _Cough_01 _Cough_02 _Cough_03 _Cough_04 _Cough_05

_Cry _Cry_01 _Cry_02 _Cry_03 _Cry_04 _Cry_05 _Cry_06 _Cry_07 _Cry_08 _Cry_09 _Cry_10

_Deh _Deh_01 _Deh_02 _Deh_03 _Deh_04 _Deh_05

_Di' _Di'_01 _Di'_02

_Eh _Eh_01 _Eh_02 _Eh_03 _Eh_04 _Eh_05 _Eh_06 _Eh_07 _Eh_08

_Ellalla'

_Embe' _Embe'_01 _Embe'_02 _Embe'_03 _Embe'_04 _Embe'_05

_Gasp

_Gnam _Gnam_01 _Gnam_02 _Gnam_03

_Grrr

_Hah

_Hahah _Hahah_01 _Hahah_02 _Hahah_03 _Hahah_04 _Hahah_05 _Hahah_06 _Hahah_07

_Heh

_Hehe _Hehe_01 _Hehe_02 _Hehe_03 _Hehe_04 _Hehe_05 _Hehe_06 _Hehe_07 _Hehe_08 _Hehe_09 _Hehe_10 _Hehe_11 _Hehe_12

_Hei _Hei_01 _Hei_02 _Hei_03 _Hei_04 _Hei_05 _Hei_06 _Hei_07

_Huhuh _Huhuh_01 _Huhuh_02 _Huhuh_03

_Laugh _Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04 _Laugh_05 _Laugh_06 _Laugh_07 _Laugh_08 _Laugh_09 _Laugh_10 _Laugh_11 _Laugh_12 _Laugh_13 _Laugh_14 _Laugh_15 _Laugh_16 _Laugh_17

_Mah _Mah_01 _Mah_02 _Mah_03 _Mah_04 _Mah_05 _Mah_06

_Mhm _Mhm_01

_Miao _Miao_01 _Miao_02 _Miao_03 _Miao_04

_Miii _Miii_01 _Miii_02

_Mizz _Mizz_01 _Mizz_02

_Mmm _Mmm_01 _Mmm_02 _Mmm_03 _Mmm_04 _Mmm_05

_Oh _Oh_01 _Oh_02 _Oh_03 _Oh_04

_Ohi _Ohi_01 _Ohi_02 _Ohi_03 _Ohi_04 _Ohi_05 _Ohi_06 _Ohi_07

_Oho _Oho_01 _Oho_02

_Ohoh _Ohoh_01 _Ohoh_02 _Ohoh_03 _Ohoh_04 _Ohoh_05

_Ops _Ops_01 _Ops_02 _Ops_03 _Ops_04

_Pf _Pf_01 _Pf_02 _Pf_03

_Prrr _Prrr_01 _Prrr_02 _Prrr_03 _Prrr_04 _Prrr_05 _Prrr_06 _Prrr_07

_Roar _Roar_01

_Shhh _Shhh_01

_Shht _Shht_01 _Shht_02 _Shht_03

_Smack _Smack_01 _Smack_02

_Ssss

_Ssst _Ssst_01

_Swallow _Swallow_01 _Swallow_02

_Throat _Throat_01 _Throat_02 _Throat_03 _Throat_04 _Throat_05 _Throat_06 _Throat_07 _Throat_08 _Throat_09 _Throat_10 _Throat_11 _Throat_12 _Throat_13

_Tie' _Tie'_01 _Tie'_02 _Tie'_03 _Tie'_04 _Tie'_05 _Tie'_06

_To' _To'_01 _To'_02

_Toh _Toh_01 _Toh_02 _Toh_03 _Toh_04 _Toh_05 _Toh_06 _Toh_07

_Uffa
_Ufff _Ufff_01 _Ufff_02 _Ufff_03
_Uh _Uh_01 _Uh_02 _Uh_03 _Uh_04
_Uhuh _Uhuh_01 _Uhuh_02
_Ups _Ups_01 _Ups_02 _Ups_03 _Ups_04 _Ups_05
_Uuuu _Uuuu_01 _Uuuu_02
_Whistle _Whistle_01 _Whistle_02 _Whistle_03 _Whistle_04 _Whistle_05
_Whistle_06 _Whistle_07 _Whistle_08 _Whistle_09 _Whistle_10 _Whistle_11
_Whistle_12 _Whistle_13
_Wow _Wow_01 _Wow_02 _Wow_03 _Wow_04 _Wow_05
_Yawn _Yawn_01 _Yawn_02
_Yeah _Yeah_01 _Yeah_02 _Yeah_03 _Yeah_04 _Yeah_05
_Yeee _Yeee_01 _Yeee_02 _Yeee_03
_Yo _Yo_01 _Yo_02 _Yo_03
_Yuhu _Yuhu_01 _Yuhu_02 _Yuhu_03 _Yuhu_04
_Yuppi _Yuppi_01 _Yuppi_02

Italian: Paola

_Acci _Acci_01
_Argh
_Atciu'
_Azz
_Bah
_Bau _Bau_01
_Beh _Beh_01
_Bleah
_Boh
_Breath _Breath_01
_Buh _Buh_01
_Buuu
_Cough _Cough_01 _Cough_02
_Cry _Cry_01
_Deh
_Di'
_Eh _Ehm
_Embe'
_Gasp
_Gnam
_Grrr
_Hah
_Haha _Haha_01
_Heh _Hehe
_Hei
_Hihi
_Hoho

_Huhu
_Laugh _Laugh_01 _Laugh_02 _Laugh_03 _Laugh_04 _Laugh_05 _Laugh_06
_Mah _Mah_01
_Mhm
_Miao _Miao_01
_Miii
_Mmm _Mmm_01
_Oh _Oho
_Oi
_Ops _Ops_01
_Pf
_Prrr _Prrr_01 _Prrr_02 _Prrr_03
_Shhh
_Shht
_Smack _Smack_01
_Sniff _Sniff_01
_Swallow _Swallow_01 _Swallow_02
_Throat _Throat_01 _Throat_02 _Throat_03 _Throat_04
_Toh _Toh_01
_Ue'
_Uff
_Whistle _Whistle_01 _Whistle_02 _Whistle_03
_Wow
_Yawn _Yawn_01 _Yawn_02 _Yawn_03
_Yeah
_Yeee
_Yo
_Yuhu
_Yuppi