Table of Contents

Introduction

The SitePal / Avatar Studio playback environment supports an API that allows you to control character speech and other runtime attributes by making JavaScript function calls from your web page. The API enables communication between your web page and the embedded Scene or Show.

> Note: SitePal users do not have access to Shows, only to Scenes. Show-specific functions which apply only to Avatar Studio are clearly marked as such.

Embedding in your web page

To use this API in your web page you must add your embed code to the page. In your account, click on the "Publish" button for the Scene or Show you wish to embed. Select the “Embed” option, copy your embed code and add it to the BODY section of your page, where you wish your character to appear.

Note: Angular and React users see special instructions on our support page.

Important caveats / pitfalls to avoid -

- API function calls may not work as expected until your embedded Scene or Show is fully loaded. It is therefore advisable to implement the 'vh_sceneLoaded' callback – and not call any API function before this callback is received. For more information please review Callback Functions documentation
- For your protection certain API functions may only work when your page is loaded from a domain you authorize. Such domains are called *Licensed Domains,* Specifically:
    - 'sayText' and 'sayAIResponse' will only work under a Licensed Domain.
    - If you turn ON 'Secure Playback' for your account, your Scene will not load except under a Licensed Domain. Note: default state is OFF.
    - Wildcards are supported in the domain name prefix. So for example, you may specify “*.mycompany.com” to cover all subdomains.
    - 'localhost' and '127.0.0.1' are always authorized and do not need to be specifically declared.
    - Add/edit your Licensed Domains & Secure Playback setting in your *Account Info* page.

Error Handling

A Licensed Domain infraction is the only case where an alert message will appear in your page. Otherwise, the API is designed to fail silently, with error messages written to the console.

Examples & Additional Reference Material

Before you get started you may want to check out our comprehensive technical examples, which cover the use of all functions in this API, as well as multiple advanced scenarios – please see reference links in Appendix A.

Note: recently added functions or details are highlited in yellow for your convenience.

Play on Load (Autoplay)

On modern web browsers, audio playback in a web page must be preceded by user interaction with the page (e.g. user clicks on a button, or touches the screen). This user action “activates” the web page which will then allow audio playback. This restriction prevents a web page from playing audio unprompted. Attempting to do so using this API will not cause a problem – but may or may not work, depending on the browser & the circumstances.

On desktop browsers, the restriction is not absolute. Some browsers will allow play-on-load for a user who has visited your page before and interacted with media on the page. The policies enacted by browsers in this regard are both evolving & undocumented.

The main takeaways therefore should be:

- It is ok to try to initiate speech as soon as the page loads (verify that API has loaded first! - see 'caveats' above)
- You should be aware that such play-on-load attempts may be blocked by the browser, and will always be blocked on mobile browsers.
- It makes sense to design your web page / application to not rely on play-on-load. You should always provide another way for the user to initiate the verbal interaction with your speaking character, in case play-on-load is blocked.

Programming for Mobile

This API is fully compatible with all major browsers on both desktop and mobile (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 differently in order to support mobile browsers in your web pages when using this API.

One issue to pay attention to is responsive design - use the "dynamicResize" API function to adjust character dimensions on responsive web pages.

> *Note: the function 'setPlayerVolume' does not have any effect in some mobile browsers – but there is no harm in making the call.*

Embedding in React or Angular

React and Angular JS framesowrks have become quite popular and can be a great way to develop your web site or web application. Because of the unique way in which web pages are loaded in these frameworks, we've made some adjustments to make our embed code compatible with them.

To embed your SitePal character in React or Angular, you will need to set your embed code "context" parameter to 1, and follow the instructions provided on our support page. See "Embed in React & Angular" section here: www.sitepal.com/support.

Belated Loading of Your Scene or Show

In some cases it may be useful or even necessary to embed your Scene or Show in your web page, without loading it when the page loads. For example you may want to display your character in a pop-up or other UI element that is not immediately displayed, but is technically part of the same page.

To accomplish this, include your embed code on your page, in the appropriate place, but set the embed code 'load' parameter to 0 (see Embed Code specification below). Your Scene will not load and nothing will be displayed, but the embed code will be lodged in your page waiting for instructions. To load your Scene or Show, use the 'loadSceneByID' API function, and its counterpart 'unloadScene' to achieve the opposite effect. API examples on our support page demonstrate this functionality.

Portals & Embedding Multiple Characters on a Page

Yes, you may embed multiple characters in a page. But how do you address an API call to a specific character when there are several on the page? To do so we will introduce the concept of 'Portal'. A Portal is what we call the embed code placed in your page, separately from the Scene embedded in it. As you will

see (by browsing through this document and our API examples) it is possible to load a different Scene (or Show) to replace a Scene in your page. That being the case, a name was needed for the embed code, that would best describe its function. Enter: “Portal”.

By using the function -

selectPortal();

you can direct all subsequent API calls to the indicated Portal. If your page contains only one Scene, you can safely ignore all this.

Please see the documentation for ‘selectPortal’ in this document for details. We also put together two API examples demonstrating how to use ‘selectPortal’ to implement a conversation between two characters on your page. Check them out on our support page, in the ‘Advanced API Examples’ section.

About the Embed Code

The embed code function is not part of this API per se, as it is not designed to be called directly or manipulated. Don’t. Furthermore, the syntax of the embed code and the meaning & values of parameters may change in future (though it will always be backwards compatible).

For these reasons, it is not necessary for you to understand the details of the embed code syntax in most cases, but we provide the following as a matter of record.

Note: the parameters ‘load’ and ‘context’ are the only parameters you may need to set manually.

```
AC_Vhost_Embed(
accountID,          // your account ID
height,         // Scene or Show height
width,          // Scene or Show width
bgcolor,        // RGB hex value of embed background color
firstslide,     // Scene to be loaded first (Studio only)
controls,       // display controls: 0 – never; 1 – as needed; 2 - always
ss,             // Show ID in Studio; Scene ID in SitePal.
sl,             // Slide ID (Studio only)
load,           // load the Scene or show: 0 – do not load; 1 – load;
context,        // 0 – Standard web page; 1 – React or Angular JS Framework
embedId,        // unique string identifies this embed instance
version,        // always use 0
os)             // overlay string – only provided with overlay embed,
                // missing from in-stream embed.
```

Function Reference

Animation Control Functions

followCursor(mode)

Available for: ☑Studio ☑SitePal

Turn “follow cursor” to the OFF, ON IN BOX, or ON IN PAGE state. If OFF, the character’s gaze ignores cursor movement. If ON IN BOX, the character’s head and eyes follow the cursor within the embed rectangle. If ON IN PAGE, the character’s gaze follows the cursor in the entire page, including areas outside the embed rectangle.

Arguments:

`mode` Required, Numeric (0/1/2):

0: follow cursor is set to OFF.

1: follow cursor is set to ON IN BOX

2: follow cursor is set to ON IN PAGE.

Example:

```
followCursor(1)
```

---

freezeToggle ()

Available for: ☑Studio ☑SitePal

Toggle between the frozen and normal states. When frozen – all character movement stops. If the character is speaking, speech is paused. Unfrozen - character wakes up. If the character was previously paused in mid-speech, speech resumes from that point.

Arguments:

None.

Example:

```
freezeToggle()
```

---

recenter()

Available for: ☑Studio ☑SitePal

Cause the character to set its gaze to the default, centered position.

Arguments:

None.

Example:

```
recenter()
```

---

setGaze(degrees, duration, [amplitude])

Available for: ☑Studio ☑SitePal

Set the direction & amplitude of the character's head and eye movement.

This call will cause the character to divert the orientation of its gaze to the specified direction, and maintain the new orientation for the specified period of time. The orientation will naturally shift towards the center (default) position when the specified time is up, or when/if the character is requested to speak.

The optional *amplitude* parameter governs the “intensity” of the head & eye movement.

Arguments:

`degrees`	Required. Numeric. 0-360 (0 deg.=top, 90 deg.=right, etc.)
`duration`	Required. Numeric. In Seconds.
`amplitude`	Optional. Numeric. In percent. 0-100. Default = 100.

Example:

```
setGaze(90,6);
```

---

setFacialExpression(expression, amplitude, duration)

Available for: ☑Studio ☑SitePal

Note: this call is only supported for 3D characters. If called for a 2D character, it has no effect. (See related function is3D).

Set the facial expression animation for a character. setFacialExpression calls do not queue, but interrupt. If a call is made while a previous call’s duration is still in effect, the first expression transforms into the second expression immediately.

Arguments:

expression Required, Text string

"None"	neutral - the default expression. Other parameters are ignored.
"ClosedSmile"	happy (closed mouth smile)
"OpenSmile"	very happy (open mouth smile)
"Sad"	sad
"Angry"	angry
"Fear"	afraid
"Disgust"	disgusted
"Surprise"	surprised
"Thinking"	thinking
"Blush"	embarrassed (blush)
"LeftWink"	wink with left eye
"RightWink"	wink with right eye
"Blink"	blink with both eyes
"Scream"	mouth wide open for screem.

amplitude Numeric. Range: 0, 1.0

The extent to which the expression should be applied. Using higher values then 1.0 is not blocked, and might be useful in some cases - but can lead to unexpected results. Feel free to experiment.

duration Required if expression!="None", Integer.

Time in seconds. Use -1 for indefinite duration.

Example:

```
setFacialExpression("OpenSmile", 0.8, 5);
                                    // sets expression to OpenSmile for 5 seconds
                                    // at 80% amplitude
```

---

clearExpressionList( )

Available for: ☑Studio ☑SitePal

Note: This function is only supported for 3D characters. If called for 2D character it has no effect.

Clear all expressions.

Example:

```
clearExpressionList();
```

---

setIdleMovement(frequency, [amplitude])

Available for: ☑Studio ☑SitePal

Note: This function is fully supported for 3D characters, and only partly supported for 2D characters. If called for a 2D character, frequency is interpreted as follows: 0 – turn OFF idle movement; non-zero – turn ON idle movement. Amplitude is ignored.

Characters that are not engaged in speaking, following the cursor, or gazing (via the *setGaze* api) randomly look around by default. This function enables users to set the frequency and intensity of the character's movement when not otherwise engaged.

Arguments:

`frequency`	Required. Numeric. The frequency with which the character performs idle time head movement. Values are 0 to 100. Default is 50. Use 0 to turn off Idle movement.
`amplitude`	Optional. Numeric. The distance from center the character sets its random gaze. Values are 1 to 100. Default is 50.

Example:

```
setIdleMovement(20,100);
```

---

setSpeechMovement(amplitude)

Available for: ☑Studio ☑SitePal

Note: This function is only supported for 3D characters.
Characters perform random head movements during speech. This function enables users to set the intensity of the character's movement when speaking or disable the movement altogether.

Arguments:

`amplitude`	Numeric. The intensity with which the character performs head movements while speaking. Values are 0 to 100. Default is 50.

Example:

```
setSpeechMovement(100);
```

---

setBlinking(frequency)

Available for: ☑Studio ☑SitePal

Note: This function is only supported for 3D characters.
This function enables users to set the frequency of the character's eye blinking, or disable blinking altogether.

Arguments:

`frequency`	Required. Numeric. The blinking frequency. Lower value means less frequent blinking. Values are 0 to 100, corresponding to an average blink interval of 0.5 sec to 10 sec. The default value is 75, corresponding to an average blink interval of 3 sec. Use 0 to turn off blinking.

Example:

```
setBlinking(50);
```

Speech Functions

loadAudio(name)

Available for: ☑Studio ☑SitePal

Preload a specific audio track by name. Calling loadAudio in advance can reduce the loading time when the audio is played. Calling loadAudio a second time, while audio is loading or after audio has been loaded has no effect.

Implement the vh_audioLoaded() event callback to be notified when the audio track is done loading.

Use the sayAudio() function to play the audio.

Arguments:

`name`	Required. String. The name of the audio track from the account.

Example:

```
loadAudio('audioname')
```

---

loadText(txt,voice,lang,engine,[effect], [effLevel])

Available for: ☑Studio ☑SitePal

Preload a specific Text To Speech audio. Calling loadText in advance can reduce the loading time when the audio is played. Calling loadText a second time, while audio is loading or after audio has been loaded has no effect.

Implement the vh_ttsLoaded() event callback to be notified when the audio track is done loading.

Use the sayText() function to play the audio.

Arguments:

`txt`	Required. String - The text to speak. Text is limited to 900 characters. (225 characters in Chinese & Japanese). Longer text will be truncated.
`voice`	Required. Integer – Voice ID, as listed in Appendix B.
`lang`	Required. Integer – Language ID, as listed in Appendix B.
`engine`	Required. Integer – Voice Family ID. See languages and voices listed in Appendix B.
`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.

Example:

```
loadText('Hello World',1,1,1)
loadText('Hello World',1,1,1,'D',3)
```

---

sayAudio(name, [startTime])

Available for: ☑Studio ☑SitePal

Play a specific audio track by name.

Arguments:

`AudioTrackName`	Required. String. The logical name of the audio as specified within the account.
`startTime`	Optional. Floating. The offset, in seconds, from the beginning of the audio from which to start audio playback.

Example:

```
sayAudio('audio name',1.9)
```

---

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

Available for: ☑Studio ☑SitePal

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

For detailed step by step instructions, please review the Guidelines for Using the TTS API in the support section.

Note: This function is available only to a Silver plan or higher account and will work only within a specified licensed domain for the account. Domain specific licensing is a security measure. If the account is not TTS enabled, or the Scene is used within a domain that you have not added to your

licensed domains, then this call will generate an alert. To edit your licensed domains please login to your account and select Account Info from the main menu bar.

Arguments:

`txt` Required. String - The text to speak. Text is limited to 900 characters. (225 characters in Chinese & Japanese). Longer text will be truncated.

`voice` Required. Integer – Voice ID, as listed in Appendix B.

`lang` Required. Integer – Language ID, as listed in Appendix B.

`engine` Required. Integer – Voice Family ID. See languages and voices listed in Appendix B.

`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)
```

---

sayAIResponse (txt,voice,lang,engine,[botID],[effect], [effLevel])

Available for: ☑Studio ☑SitePal

For detailed step by step instructions on getting started with AI using this function, please review Implementing Your AI Agent in the Support section.

An Artificial Intelligence knowledge base provides a real time text and audio response to a text "question". The Audio is generated & spoken automatically via a Text To Speech engine according to the selected voice, language and engine. The response text is provided via the Event function 'vh_aiResponse()'.

The Artificial Intelligence knowledge base is based on the extensive A.L.I.C.E. knowledge base, which includes over 23,000 data entries. Your knowledge base can be edited and customized in the AI Management Center (AIMC) – click on AIMC from the main menu within your account.

Note:

This function is available only to a Gold plan or higher account and will work only within a licensed domain for the account. Domain specific licensing is a security measure. If the call is used in a domain that you have not added to your licensed domains, this call will generate an alert. To edit your licensed domains please login to your account and select Account Info from the main menu bar.

Arguments:

`txt` Required. String - The text to speak. Text is limited to 900 characters. (225 characters in Chinese & Japanese). Longer text will be truncated.

`voice` Required. Integer – Voice ID, as listed in Appendix B.

`lang` Required. Integer – Language ID, as listed in Appendix B.

`engine` Required. Integer – Voice Family ID. See languages and voices listed in Appendix B.

`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.

Example:

```
sayAIResponse('Sing me a song',2,1,2)
sayAIResponse('Sing me a song',2,1,2,,'P',-1)
```

---

saySilent (seconds)

Available for: ☑Studio ☑SitePal

Speech is visually simulated: No audio is downloaded, no streams are consumed and no interaction with the server is performed.

This function call may be useful when you want to call attention to the character but do not want use audio to do so. Most pertinent example is use in ad banners, where (in some cases) the use of audio may only be permitted after mouse rollover.

saySilent is always in 'InterruptMode' ON, meaning that any function call which invokes actual speech will interrupt simulated speech. saySilent calls cannot be queued.

Arguments:

`Seconds`	length of time desired for simulated speech, in seconds

Example:

```
saySilent(10)
```

---

setPlayerVolume (level)

Available for: ☑Studio ☑SitePal

Note: This function has no effect on some mobile browsers.
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 the speech (lip movement continues) or stop the audio stream. It affects only the volume . To stop the speech, use the function stopSpeech().

---

stopSpeech ()

Available for: ☑Studio ☑SitePal

Stop the speech of a currently speaking character. If the character is not currently speaking, stopSpeech has no effect (i.e. it does not prevent speech that has not yet begun).

Arguments:

None.

Example:

```
stopSpeech()
```

---

replay (ignorelimit)

Available for: ☑Studio ☑SitePal

Plays or replays current Scene, from the start.

If `interruptMode` is ON, ongoing playback (if any) is interrupted, and immediately begins again. If `interruptMode` is OFF, playback is queued. See setStatus to learn about `interruptMode`.

Playback limit is a Scene attribute which can be set in the Scene Options dialog. Set the `ignoreLimit` parameter to override this Scene option.

Arguments:

`ignoreLimit`	Optional. Boolean default value is 0. If 1 then Scene ignores playback limit status.

Example:

```
replay()
replay(1)
```

Scene Attributes

getSceneAttributes()

Available for: ☑Studio ☑SitePal

Retrieve several key Scene attributes. Function returns an object.

Arguments:

None.

Return Values:

An object with following named values is returned:

sceneID	Scene's database ID
bgName	Logical name of background assigned to Scene if any.
audioName	Logical name of audio assigned to Scene if any.
modelID	Database ID of model assigned to Scene if any.

Example:

```
var attr = getSceneAttributes();
var sceneID      = attr.sceneID;
var bgName       = attr.bgName;
var audioName    = attr.audioName;
var modelID      = attr.modelID;
```

---

setBackground (bgName)

Available for: ☑Studio ☑SitePal

Scene Background is modified. Change is not persistent.

Arguments:

bgName	Required. String. The logical name of the background as specified within the account. If an empty string is provided, the background is cleared.

Example:

```
setBackground('background name');
setBackground('');                    // clear the bg
```

---

setBackgroundColor (bgColor)

Available for: ☑Studio ☑SitePal

Scene background color is modified. Change is not persistent.

Note: If the Scene has a background image, the background color, which is displayed behind the image, may not be visible, except while loading.

Arguments:

`bgColor` Required. String. Hexadecimal RGB color representation.

Example:

```
setBackgroundColor('0000AA');
```

---

setColor (part,color)

Available for: ☑Studio ☑SitePal

Dynamically modify the color of the specified character "part".

Colors are applied to the grayscale baseline of the specific character's design. Therefore, the effect of a color on the specified area may not exactly match its color value, as you are seeing the effect of its application to a non-white surface. For the same reason, results may differ when the same color is applied to different characters.

Arguments:

`part` Required. String. The character part to color.

One of: 'eyes', 'hair', 'make-up', 'mouth', 'skin'

`color` Required. String. Hexadecimal RGB color representation.

Example:

```
setColor('eyes','0000AA')
```

---

setStatus (interruptMode,progressInterval,gazeSpeed,displayControls)

Available for: ☑Studio ☑SitePal

This function is used to 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 and sayAudio) are queued for consecutive playback.

If set to 1 current audio is interrupted when sayAudio or sayText are called.

`progressInterval`

Required. Non-negative Integer – Default = 0.

The audio progress interval value controls progress callbacks which take place during playback. The callback function

vh_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 "vh_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.

`gazeSpeed`

Required. Integer (0/1/2) – Default = 0.

Controls the reaction speed of the character when responding to setGaze function calls.

0 - slow
1 - medium
2 - fast

`displayControls`

Required. Integer (0/1/2) – Default = 1.

Controls the display of playback controls.

0 – Never. This can be useful if you want to create your own controls.
1 – As needed. Controls are shown when cursor rolls over the Scene (or Scene is touched).
2 – Always.

Example:

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

---

dynamicResize (width, height)

Available for: ☑Studio ☑SitePal

The dimensions of the embedded Show or Scene are dynamically modified without reloading the character. This can be used to support responsive design. Change is not persistent - if the page is reloaded, the embedded Show or Scene will load as originally embedded.

Maintaining the original aspect ratio is not required. If you would like to retain the relative position of the character within the Scene frame, you should retain the aspect ratio. Otherwise, character will be re-positioned as best possible.

Tip: To control the dimensions of the embedded Show or Scene when page is initially loaded, you could set the width and height *before* the page loads, using a back end programming language such as Java or php.

```
<scripttype="text/
javascript">AC_VHost_Embed(accountid,height,width,'',1,1, showsceneid,
0,1,0,'94fb29b9a4767343f36dd16fc8c0f81a',0);</script>
```

Arguments:

`width`	Required. Integer. The new width.
`height`	Required. Integer. The new height.

Example:

```
dynamicResize(300,200)
```

---

is3D ()

Available for: ☑Studio ☑SitePal

Is the character in the current Scene a 3D character?

Boolean function – returns true if 3D character is used, false otherwise.

Arguments:

None.

Example:

```
is3D()
```

Embed Overlay Functions

The following functions apply only to Scenes and Shows that are embedded as an overlay on top of the page. If any of these functions is called for a Scene or Show which is embedded in-line within page content, the function call will have no effect.

overlayOpen (mode, play)

Available for: ☑Studio ☑SitePal

Scene or Show is opened, or toggled between minimize and maximize display mode. If mode is 'max' then play parameter governs the playback behavior of the Scene/Show when opened.

Arguments:

`mode` `'max' - open Scene/Show in maximized mode or toggle to maximize mode.` If the Scene is already maximized, this call has no effect.

`'min' - open Scene/Show in minimized mode or toggle to minimize mode.` The effect is equivalent to a click on the minimize button. If the Scene is already minimized, this call has no effect.

`play` Optional. Integer (0/1/2) – Default = 2.

Relevant only for 'max' mode. If mode is 'min' then parameter is ignored.

If set to 0, playback does not start. Settings ignored

If set to 1, playback immediately starts. Settings ignored.

If set to 2, playback may start depending on the values of the "playback limit" and "play on load" settings.

Examples:

```
overlayOpen('max',1)
overlayOpen('max')
overlayOpen('min')
```

---

overlayClose ()

Available for: ☑Studio ☑SitePal

Scene or Show is closed. The effect is equivalent to a click on the close button. If Scene is already closed, this call has no effect.

Arguments:

`None.`

Example:

```
overlayClose()
```

Navigation Flow Functions – Avatar Studio

gotoNextScene ()

Available for: ☑Studio ☐SitePal

The current Scene is interrupted, and the next Scene (according to the preset show flow) immediately begins. This has the same effect as pressing the player's 'Next' button.

Arguments:

None

Example:

```
gotoNextScene()
```

---

gotoPrevScene ()

Available for: ☑Studio ☐SitePal

The current Scene is interrupted, and the previous Scene immediately begins. This has the same effect as pressing the player's 'Previous' button.

Arguments:

None

Example:

```
gotoPrevScene()
```

---

gotoScene (sceneRange)

Available for: ☑Studio ☐SitePal

The current Scene is interrupted, and the specified scene immediately begins.

Arguments:

`sceneRange`	Required. String Indicates the next Scene to follow the currently playing Scene. Can be either: - Index of specific Scene - Range of consecutive scenes to randomly choose from. The format is hyphen delimited.

Example

`gotoScene('3')` – goto a specific Scene

`gotoScene('4-7')` – goto a randomly selected scene from the set: 4,5,6,7

---

preloadNextScene ()

Available for: ☑Studio ☐SitePal

Preloads the assets of the next scene in a show. Upon successful preloading of a scene the callback vh_scenePreloaded() will be invoked. If there is no next scene the call is ignored. Subsequent calls to this function or preloadScene while a scene is loading will be ignored.

Arguments:

None

Example:

```
preloadNextScene()
```

---

preloadScene (sceneIndex)

Available for: ☑Studio ☐SitePal

Preloads the assets of the specified scene. Upon successful preloading of a scene the callback vh_scenePreloaded() will be invoked. If there is no scene with the specified index number the call is ignored. Subsequent calls to this function or preloadNextScene while a scene is loading will be ignored.

Arguments:

`sceneNumber` Required. Integer

The index of the scene to preload

Example

`preloadScene(3)` – preload a specific Scene

Manage Embeded Scenes/Shows in Page

Note: The word "Scene", as used in this section, should be understood to refer to the 'Embedded Entity' be it a Scene or a Show.

loadSceneByID (sceneID, slideID)

Available for: ☑Studio ☑SitePal

Replace the embedded Scene with another. The current Scene is interrupted, and the specified Scene is loaded instead.

Arguments:

`sceneID`	*Optional.* ID the Scene or Show to load. If not provided, the Scene which is specified in the embed code is loaded. In your embed code, the Scene or Show ID is parameter #7. To find the Scene ID for the Scene you want to load, select the Publish in Web Page option for that Scene, and locate the value used as the 7th parameter. See "About the Embed Code" for more information. Note: sceneID must belong to your account.
`slideID`	*Optional.* Studio only. ID of the Slide you would like to load first. If missing, first Slide is loaded. In your embed code, the Slide ID is the 8th parameter. Note: slideID must belong to the specified Scene.

Example

`loadSceneByID()` – load Scene specified in the portal's embed code

`loadSceneByID(459284)` – load specified Scene, first Scene to be displayed

---

unloadScene()

Available for: ☑Studio ☑SitePal

Unload the embedded Show or Scene. This removes the Scene from the page, and retains only the embed portal. Another (or the same) Show or Scene may be subsequently loaded into the portal.

Arguments:

none.

Example

```
unloadScene()
```

selectPortal (portal)

Available for: ☑Studio ☑SitePal

Notes:

- "SitePal Conversation" in Advanced Examples page in the Support section demonstrates how this function is used.
- Using this function requires embed functions v.4. To use this feature, locate this text in your embed code:
  - vhost_embed_functions_v2.php (for full body scenes, "v3")

    and modify to:
  - vhost_embed_functions_v4.php

A "portal" is what we call the space on the page created by the embed code, to be used for loading a Scene into. Typically the portal loads and displays a Scene which was originally specified when the embed code was created. But it is possible to load another Scene into it, to replace a previously loaded Scene. Hence the name: portal.

When multiple portals are embedded on the same page, only one can accept API calls at any given moment. This call is used to select the embed portal that accepts API calls, and thus allow the API to target a specific portal.

To use this call, you must collect the embed portal reference when the page is loaded – by assigning the values returned by the "Embed" function as follows:

```
var alice = AC_Vhost_Embed(…
var bob = AC_Vhost_Embed(…
```

The variables 'alice' and 'bob' in this example will contain the reference values for the two portals. Use these values to specify the currently active portal, and thus target your API functions. There is no limit to the number of portals you can embed on a single page.

Arguments:

`portal` Required. Portal reference retrieved from Embed function.

Example:

```
selectPortal(alice)
```

Status Callback Functions

Callback Functions can help improve coordination between the embedded Scene or Show and your page / application.

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 using callback functions is optional; There is no need to add callback functions which you do not intend to use.

Note: API functions work only after Scene or Show has completed loading

Keep in mind that certain aspects of the API may not function in a predictable manner until the "vh_sceneLoaded" status callback has been called/dispatched. It is therefore advisable to always implement the "vh_sceneLoaded" callback & check that it has been called before calling any API function.

vh_aiResponse (responseText, portal)

Available for: ☑Studio ☑SitePal

Triggered when an AI Response is returned, this call returns the text that is generated by the AI knowledge base in response to the function call 'sayAIResponse'.

Arguments:

`responseText`	Response text
`portal`	Reference to the portal that generated the call.

Example -

```
function vh_aiResponse(responseText, portal){
}
```

---

vh_audioLoaded (audioName, portal)

Available for: ☑Studio ☑SitePal

Triggered when an audio preload is done, and returns the name of the audio that was provided as input to 'loadAudio()'.

Arguments:

`audioName`	Loaded audio name
`portal`	Reference to the portal that generated the call.

Example -

```
function vh_audioLoaded(audioName,portal){
}
```

---

vh_ttsLoaded (audioText, portal)

Available for: ☑Studio ☑SitePal

Triggered when a Text-To-Speech audio preload is done and returns the text that was provided as input to 'loadText()'.

Arguments:

`audioText`	Loaded text to be spoken
`portal`	Reference to the portal that generated the call.

Example -

```
function vh_ttsLoaded(audioText, portal){
}
```

---

vh_audioProgress (percentPlayed, portal)

Available for: ☑Studio ☑SitePal

Called during playback, if and only if the 'progressInterval' status is set.

vh_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 status.

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.
`portal`	Reference to the portal that generated the call.

Example -

```
function vh_audioProgress(percentPlayed, portal) {
}
```

---

vh_portalReady (portal)

Available for: ☑Studio ☑SitePal

Triggered when the embed code is fully loaded, and before Scene is to be loaded. Use this callback to verify that the portal is ready to accept loadScene calls. No other API calls can be made at this time.

Note: when this callback is received, your Scene has not yet loaded. You must wait for the vh_sceneLoaded callback before calling any API function which affects the Scene.

Arguments:

`portal`	Reference to the portal that generated the call.

Example -

```
function vh_portalReady(portal){
      alert("the embed code has loaded");
}
```

---

vh_sceneLoaded (sceneIndex, portal)

Available for: ☑Studio ☑SitePal

Triggered when the Scene is fully loaded & displayed, just before the audio starts playing. Use this callback to verify Scene is ready to accept API calls.

Arguments:

`sceneIndex`	For Studio accounts: The index of the loaded Scene in the Show. Undefined for SitePal accounts.
`portal`	Reference to the portal that generated the call.

Example -

```
function vh_sceneLoaded(sceneIndex, portal){
      alert("scene has loaded");
}
```

---

vh_scenePreloaded (sceneIndex, portal)

Available for: ☑Studio ☐SitePal

Triggered after a successful call to preloadScene or preloadNextScene. The assets of the scene are loaded to memory. Subsequent display of the specified scene should be immediate.

Arguments:

`sceneIndex`	For Studio accounts: The index of the loaded Scene in the Show.
`portal`	Reference to the portal that generated the call.

Example -

```
function vh_scenePreloaded(sceneIndex, portal){
      alert("the scene is preloaded. index: "+ sceneIndex);
}
```

---

vh_talkStarted (portal)

Available for: ☑Studio ☑SitePal

Triggered when the character starts talking. When several audios are played in sequence, this callback will be dispatched at the start of the sequence.

Arguments:

`portal`	Reference to the portal that generated the call.

Example -

```
function vh_talkStarted(portal){
}
```

---

vh_talkEnded (portal)

Available for: ☑Studio ☑SitePal

Triggered when the character is done talking. When several audios are played in sequence, this callback will be dispatched at the end of the sequence.

Arguments:

`portal`	Reference to the portal that generated the call.

Example -

```
function vh_talkEnded(portal){
}
```

---

vh_audioStarted (portal)

Available for: ☑Studio ☑SitePal

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

Arguments:

`portal`	Reference to the portal that generated the call.

Example -

```
function vh_audioStarted(portal){
}
```

---

vh_audioEnded (portal)

Available for: ☑Studio ☑SitePal

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

Arguments:

`portal`	Reference to the portal that generated the call.

Example -

```
function vh_audioEnded(portal){
}
```

---

vh_playPause (status, portal)

Available for: ☑Studio ☑SitePal

Triggered when the play/pause button is pressed. This enables synchronization

Arguments:

`status`	0=paused; 1=playing.
`portal`	Reference to the portal that generated the call.

Example - JavaScript

```
function vh_playPause(status, portal){
      alert("play/pause button pressed. status: "+ status);
}
```

Appendix A: API Examples

We've put together a collection of technical examples that demonstrate how to use every one of our API functions and callbacks.

As you review these examples, feel free to view their source code, and to copy that source code for use in your own test pages as you get started. You will of course need to replace the embed code in the example with your own embed code from your own account - if you want to have control over character & speech.

Note: a common problem when getting started is not setting up you own "licensed domain" - which is required for certain functions to work. This feature is provided for your security. To enter your domain(s) select "Account Info" from the main menu bar. You can add your domain(s) at the bottom of the page.

Our API support examples can be found here:

Using the Client API - **Technical Examples**

Additional reference material can be found in our support section.

Appendix B: Text to Speech Languages and Voices

The following tables list Engine IDs, Language IDs and Voice IDs to be used in API calls.

Tip: to select your TTS voice, you may want to preview available voices here – www.ttsdemo.com

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

TTS 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	

TTS 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
English	1	Beth	8	F	US
Spanish	2	Violeta	1	F	Mexican
Spanish	2	Francisco	2	M	Mexican
Spanish	2	Gloria	3	F	Mexican
Spanish	2	Lola	4	F	Castilian
Spanish	2	Manuel	5	M	Castilian

German	3	Lena	1	F	
German	3	Tim	2	M	
French	4	Chloe	1	F	Canadian
French	4	Leo	2	M	Canadian
French	4	Roxane	3	F	European
French	4	Louis	4	M	European
Portuguese	6	Helena	1	F	Brazilian
Portuguese	6	Rafael	2	M	Brazilian
Italian	7	Elisa	1	F	
Italian	7	Roberto	2	M	
Chinese	10	Lily	1	F	Mandarin
Chinese	10	Hui	3	F	Mandarin
Chinese	10	Liang	4	M	Mandarin
Chinese	10	Qiang	5	M	Mandarin
Chinese	10	Kaho	6	M	HK Cantonese
Chinese	10	Kayan	7	F	HK Cantonese
Chinese	10	Yafang	8	F	Taiwanese
Japanese	12	Show	2	M	
Japanese	12	Misaki	3	F	
Japanese	12	Sayaka	4	F	
Japanese	12	Hikari	5	F	
Japanese	12	Haruka	6	F	
Japanese	12	Ryo	7	M	
Japanese	12	Takeru	8	M	
Korean	13	Yumi	1	F	
Korean	13	Junwoo	2	M	
Korean	13	Hyeryun	4	F	
Korean	13	Jimin	5	F	
Korean	13	Sena	6	F	
Korean	13	Dayoung	7	F	
Korean	13	Hayuna	8	F	
Korean	13	Yura	9	F	
Korean	13	Jihun	10	M	
Thai	26	Sarawut	1	M	
Thai	26	Somsi	2	F	

TTS 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 = 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	Portugal
Portuguese	6	Tiago	4	M	Portugal
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 C: SSML Tags for Text to Speech

The Speech Synthesis Markup Language (SSML) is an XML based language used to represent instructions to Text-To-Speech engines when processing input text. SSML Tags are inserted within the actual text to be processed, and are subsequently interpreted by the TTS engine to affect the manner in which voice audio is generated.

Using SSML Tags in your input text is not necessary, but allows you to achieve more precise control over the manner in which the text is spoken.

The syntaxt for SSML is an emerging standard, governed by the W3C. The specification for SSML 1.0 has only recently been finalized (see SSML Specification for more information). It should therefore come as no surprise that support for SSML is not yet fully or uniformly implemented.

We have reviewed what we consider the most relevant tags, and verified their implementation and functionality within the available TTS Engines. The following list summarizes our findings. For each of the listed tags, we note the support status per each of the TTS Engines (a.k.a Voice Family) #2 and #3 (Loquendo and Neospeech). Note that where specific languages are mentioned, this means that other languages for that TTS Engine have been reviewed and are not supported. This list will be updated from time to time.

Note: TTS Engine #4 does not support SSML tags. Please select only voices from Engines #2 and #3 for use with SSML.

Additional SSML tags, which are part of the SSML Specification but not listed here, might be useful for your purposes. Please feel free to experiment and come to your own conclusion regarding the suitability of unlisted tags.

> *Note that SSML tag interpretation is case sensitive, and the case of opening and closing tags must match!*
>
> *Examples:*
>
> ```
> <Prosody volume="loud"> very loud </prosody> Wrong
> <prosody volume="loud"> very loud </prosody> Correct
> <Prosody volume="loud"> very loud </Prosody> Correct
> ```

Structure Elements

Break

The **Break** tag instructs the TTS engine to insert a pause in the synthesized text in one of three ways.

Loquendo: **Partial support.**

Loquendo does not support the "size" attribute of the <break /> element, only the "time" attribute.

Neospeech: Supported

Syntax: `<BREAK/>`

Example: `Time for a pause <Break/> Okay, keep going.`

Inserts a brief break after the word "pause".

Syntax: `<BREAK Size="none | small | medium | large"/>`

Example: `No time for a pause <Break size="none"/> Keep going.`

Inserts no break after the word "pause".

Example: `Time for a pause <Break size="medium"/> Okay, keep going.`

Inserts a brief silence, the equivalent of the silence following a sentence, after the word "pause".

Example: `Time for a pause <Break size="large"/> Keep going.`

Inserts only the default break after the word "pause".

Example: `Time for a pause <Break size="medium"/> Okay, keep going.`

Inserts the equivalent of a paragraph break of silence after the word "pause".

Syntax: `<BREAK time=" duration "/>`

Example: `Break for 100 milliseconds <Break time="100ms"/> Okay, keep going.`

Inserts 100 milliseconds of silence after the word "milliseconds".

Example: `Break for 3 seconds <Break time="3s"/> Okay, keep going.`

Inserts 3 seconds of silence after the word "seconds".

Paragraph

The **PARAGRAPH** tag tells the TTS engine to change the prosody to reflect the end of a paragraph, regardless of the surrounding punctuation.

Syntax: `<PARAGRAPH> text </PARAGRAPH>`

`<P> text </P>`

Loquendo: Supported

Neospeech: Supported

Example: `<Paragraph> This example has only one sentence in the paragraph </Paragraph>`

Example: **<**`P`**>** `The paragraph tag can be abbreviated as just the letter P.` **<**`/P`**>**

The TTS engine changes the prosody to reflect the paragraph boundaries.

Sentence

The **SENTENCE** tag tells the TTS engine to change the prosody to reflect the end of a sentence, regardless of the surrounding punctuation.

Syntax: `<SENTENCE> text </SENTENCE>`

`<S> text </S>`

Loquendo: Supported

Neospeech: Supported

Example: `<Sentence> This text is a sentence. </Sentence>`

Example: `<S> The sentence tag can be abbreviated as just the letter S. </S>`

The TTS engine changes the prosody to reflect the sentence boundaries.

Prosody Elements

Volume

The **Volume** attribute of the **Prosody** tag allows the application to change the volume of the TTS voice. Note that this does not change the volume of the output device, but it does raise or lower the volume of the text spoken within the context of the tag.

Syntax: `<PROSODY VOLUME=" level "> text </PROSODY>`

where *level* is a value from 0.0 to 200.0. A value of 100 is the voice's default volume, a value of 0 changes the volume to 0 and a value of 200 doubles the volume. The volume changes linearly.

Syntax: `<PROSODY VOLUME=" silent | soft | medium | loud "> text </PROSODY>`

Sets the absolute volume to the specified level.

Loquendo: Supported

Neospeech: Supported

Example:
```
This is the default volume
<prosody volume="silent"> silence </prosody>
<prosody volume="soft"> Now I'm whispering </prosody>
<prosody volume="120"> a little louder </prosody>
<prosody volume="medium"> medium volume</prosody>
<prosody volume="loud"> very loud </prosody>
```

Rate

The **RATE** attribute of the **Prosody** tag changes the rate at which the text is spoken. You can specify either the absolute rate or a relative change in the current speaking rate.

Syntax: `<PROSODY RATE="x-fast | fast | medium | slow | x-slow | default"> text </PROSODY>`

Syntax: **`<PROSODY RATE="`***`relativeChange`***`">`** *text* **`</PROSODY>`**

changes the speaking rate which is expressed in Words Per Minute (WPM) or in percentage terms. *relativeChange* is a floating point number that is added to or subtracted from the current rate. A "+" or "-" sign must precede the number. If a percent sign follows then the change is interpreted as a percentage change..

Loquendo: Supported

Neospeech: Supported

Example:
```
This is the default speed
<prosody rate=”slow”> this is speaking slowly
   <prosody rate=”fast”> this is speaking fast </prosody>
    back to slow
</prosody>
back to the default rate
```

Example:
```
This is the default speed
<prosody rate="-50%">
    this is 50% slower
    <prosody rate="+50%"> this is 50% faster </prosody>
    back to 50% slower
</prosody>
back to the default rate
```

Pitch

The **PITCH** attribute of the **Prosody** tag changes the pitch at which the text is spoken. You can specify either the absolute pitch or a relative change in the current speaking pitch.

Syntax: `<PROSODY PITCH=”x-high | high | medium | low | x-low | default”> text </PROSODY>`

Syntax: `<PROSODY PITCH=”relativeChange”> text </PROSODY>`

relativeChange is an floating point number, expressed as a percentage that is added to or subtracted from to the current pitch. A “+” or “-“ sign must precede the number, and the percent sign must follow.

Loquendo: Supported

Neospeech: Supported

Example: `<prosody pitch="+12.5%"> Higher pitch sentence </prosody>`
Example: `<prosody pitch="high"> High pitch sentence </prosody>`

The Voice Element

The **Voice** tag enables control the voice of the TTS speaker from the input text. You can use this feature to change voices, e.g. you might use different voices to speak different sections of an email message or carry on a conversation between two different voices. You can even use different languages within the same sentence.

Note: this can only work when switching voices within the same voice family.

Select a voice by specifying one of the following attributes:

Gender, **Name**.

It is best to specify the speaker by **Name**, in which case the Gender attribute is unnecessary.

Syntax:
```
<VOICE
    Gender="male | female | neutral"
    Name= voicename
</VOICE>
```

Loquendo: **Supported by some voices**

Neospeech: **Supported**

Example:
```
<voice name="Bridget"> This is Bridget, <Voice
Name="Violeta"> Hola, me llamo Violeta,</Voice> and this
is Bridget again. </voice>
```

This string is pronounced in Bridget's voice "This is Bridget", then in Violeta's voice in Spanish, "Hola, me llamo Violeta", then in Bridget's voice, "This is Bridget again".