



Table of Contents

Introduction	4
Function Reference	5
Facial Control Functions	5
followCursor(mode)	5
freezeToggle ()	5
recenter()	5
setGaze(degrees, duration, [magnitude])	6
setFacialExpression(expressionId, duration, amplitude)	6
setIdleMovement(frequency, amplitude)	7
setSpeechMovement(amplitude)	7
Speech Functions	7
loadAudio(name)	7
loadText(txt,voice,lang,engine,[effect], [effLevel])	8
sayAudio(name, [startTime])	9
sayText (txt,voice,lang,engine,[effect], [effLevel])	9
sayAIResponse (txt,voice,lang,engine,[botID],[effect], [effLevel])	10
saySilent (seconds)	11
setPlayerVolume (level)	11
stopSpeech ()	11
replay (ignorelimit)	12
Speech Functions for Exported Scenes	13
sayAudioExported(name, accountId, [startTime])	13
sayTextExported(txt,voice,lang,engine,accountId,[effect], [effLevel])	13
sayAIResponseExported (txt, voice, lang, engine, accountId, [botID],[effect], [effLevel])	14
Scene Attributes	15
setBackground (bgName)	15
setColor (part,color)	15
setLink (href,target)	16
setStatus (interruptMode,progressInterval,gazeSpeed)	16
is3D ()	17
Embed Overlay Functions	18
overlayOpen (mode, play)	18
overlayClose ()	18
Navigation Flow Functions	19
gotoNextScene ()	19
gotoPrevScene ()	19
gotoScene (sceneRange)	19
loadScene (sceneIndex)	19
loadShow (showIndex)	20
preloadNextScene ()	20
preloadScene (sceneIndex)	20
setNextSceneIndex (sceneRange)	21

Status Callback Functions	22
vh_aiResponse ()	22
vh_audioLoaded (audioName)	23
vh_ttsLoaded (text)	23
vh_audioProgress (percentPlayed)	23
vh_sceneLoaded (sceneIndex)	24
vh_scenePreloaded (sceneIndex)	24
vh_talkStarted ()	25
vh_talkEnded ()	25
vh_audioStarted ()	25
vh_audioEnded ()	26
Appendix A: API Examples	27
Appendix B: Text to Speech Languages and Voices	28
Appendix C: SSML Tags for Text to Speech	34
Say-as Elements	34
Currency	34
Date:MD	35
Date:MDY	35
Date:Y	35
Number	35
Number:Decimal	35
Number:Fraction	36
Telephone	36
Time	36
Structure Elements	36
Break	36
Paragraph	37
Sentence	37
Prosody Elements	38
Volume	38
Rate	38
Pitch	39
The Voice Element	39
Appendix D: Loquendo Expressive Cues	40

Introduction

The SitePal/Studio Player provides an API (Application Programming Interface) that allows a JavaScript or Flash ActionScript programmer to affect different runtime attributes by making function calls from the hosting HTML page or Flash movie. The API enables communication between the host environment and the embedded VHost Scene or Show.

API functions allow you to interface with Shows and Scenes embedded in Flash movies and HTML pages. Unless otherwise noted, the API functions are identical for both JavaScript and ActionScript environments. Where the usage or parameters differs, this is clearly noted.

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

API functions work only after Scene or Show has completed loading

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 implement the “vh_sceneLoaded” callback – and not call any API functions until the “vh_sceneLoaded” callback has been invoked.

For more detailed information please review the [Status Callback Functions](#) section.

Embedding in an HTML page:

To use the functions in an HTML page you must add the ‘JavaScript API Stub Code’ to the HTML page in which the VHost object is embedded. The JavaScript API Stub Code is available in the ‘Embed’ window. Copy-paste the JavaScript stub code from the ‘Embed’ window into the HTML page.

When calling an API function from within a different frame or window, be sure to reference the page where the JavaScript API Stub Code is embedded.

Note: Basic knowledge of JavaScript is assumed.

Embedding in a Flash movie:

Embed a Show or a Scene in a Flash movie by using the ‘loadMovie’ ActionScript function. Once you load the VHost object into a movie instance, any call to a VHost API function should be formatted as: instance.function() , where “instance” is the name of the object into which the VHost object was loaded.

Note: Basic knowledge of Flash ActionScript is assumed.

Examples & Additional Reference Material

Before you get started – you may want to check out our comprehensive source code examples – please see reference links in [Appendix A](#).

Function Reference

Facial 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 remains fixed, and 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 flash 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 pause and play states. Pause – character falls asleep. If the character is speaking, speech is paused. Play - 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 VHost character to set its gaze to the default position.

Arguments:

None.

Example:

```
recenter()
```

setGaze(degrees, duration, [magnitude])

Available for: Studio SitePal

Set the direction & magnitude of the VHost character head and eye movement.

This call will cause the VHost 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 VHost is requested to speak.

The optional *magnitude* 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.
magnitude	Optional. Numeric. In percent. 0-100. Default = 100.

Example:

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

setFacialExpression(expressionId, duration, amplitude)

Available for: Studio SitePal

Set the facial expression animation for a character. If “0” is passed no duration is required. setExpression 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.

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

Arguments:

expressionId	Required, Numeric
	0 – neutral (stop all expressions. duration is not relevant if 0 is passed)
	1 – happy (closed mouth smile)
	2 – very happy (open mouth smile)
	3 – sad
	4 – angry
	5 – afraid
	6 – disgusted
	7 – surprised
	8 – thinking
	12 – embarrassed (blush)
duration	Required if expressionID!=0, Numeric.
	Time in seconds. Use -1 for indefinite duration.
amplitude	Optional, Numeric. Range: 0, 100; default = 100

The extent to which the expression should be applied

Example:

```
setExpression(3, 6); // sets expression to sad for 6 seconds at max amplitude
```

setIdleMovement(frequency, amplitude)

Available for: Studio SitePal

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

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

Arguments:

<code>frequency</code>	Optional. 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.
<code>amplitude</code>	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

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.

Note: This function is supported only for 3D characters.

Arguments:

<code>amplitude</code>	Numeric. The intensity with which the character performs head movements while speaking. Values are 0 to 100. Default is 50.
------------------------	---

Example:

```
setSpeechMovement (100) ;
```

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. Most languages are limited to 600 characters for SitePal, 900 characters in Studio. The exceptions are: Chinese & Japanese which are limited to 150 characters in SitePal, 225 characters in Studio. A longer text string 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 listed in Appendix B .
effect	Optional. Character. Audio effect – one of: <ul style="list-style-type: none">• “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:<ul style="list-style-type: none">◦ Bullhorn level: 3 (note: levels 1 and 2 are deprecated)• “T” – Time:<ul style="list-style-type: none">◦ Echo level: 1◦ Reverb level: 2◦ Flanger level: 3◦ Phase level: 4
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](#).

Note:

This function is available only to a TTS Enabled account and will work only within a licensed domain for the account. Domain specific licensing is necessary to avoid misappropriation of TTS streams. If the account is not 'TTS-Enabled', or the Scene is used within a domain that is not specifically licensed for the account, then this call will have no effect. To enable your account for TTS, please visit the SitePal store. To edit your licensed domains please select Account Info.

Arguments:

txt	Required. String - The text to speak. Most languages are limited to 600 characters for SitePal, 900 characters in Studio. The exceptions are: Chinese & Japanese which are limited to 150 characters in SitePal, 225 characters in Studio. A longer text string 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 listed in Appendix B .
effect	Optional. Character. Audio effect – one of: <ul style="list-style-type: none">• “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:

- o Bullhorn level: 3 (note: levels 1 and 2 are deprecated)
 - “T” – Time:
 - o Echo level: 1
 - o Reverb level: 2
 - o Flanger level: 3
 - o Phase level: 4
- 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

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.

Notes:

- This function is available only to a TTS Enabled account and will work only within a licensed domain for the account. Domain specific licensing is necessary to avoid misappropriation of TTS streams. If the account is not ‘TTS-Enabled’, or if the Scene is used within a domain that is not approved for the account, then this call will have no effect. To enable your account for TTS, please visit the SitePal store. To edit your licensed domains please select Account Info.
- If you need to customize the AI knowledge base to your needs, please send a note to support@oddcast.com.

Arguments:

- txt Required. String - The text to speak. Most languages are limited to 600 characters for SitePal, 900 characters in Studio. The exceptions are: Chinese & Japanese which are limited to 150 characters in SitePal, 225 characters in Studio. A longer text string 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 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:
 - o Bullhorn level: 3 (note: levels 1 and 2 are deprecated)
 - “T” – Time:

- o Echo level: 1
- o Reverb level: 2
- o Flanger level: 3
- o Phase level: 4

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

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 VHost 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)
```

Speech Functions for Exported Scenes

sayAudioExported(name, accountId, [startTime])

Available for: Studio SitePal

Note: This call is only supported for Studio customers whose account has been configured to work with exported Scenes. Contact support@oddcast.com for more information.

Play a specific audio track by name, from an exported Scene.

Arguments:

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

Example:

```
sayAudioExported('audio name', xxxx, 1.9)
```

sayTextExported(txt,voice,lang,engine,accountId,[effect], [effLevel])

Available for: Studio SitePal

Note: This call is only supported for Studio customers whose account has been configured to work with exported Scenes. Contact support@oddcast.com for more information.

Real-time (dynamic) Text-To-Speech (TTS), called from an exported Scene.

For detailed step by step instructions, please review the [Guidelines for Using the TTS API](#).

Note:

This function is available only to a TTS Enabled account and will work only within a licensed domain for the account. Domain specific licensing is necessary to avoid misappropriation of TTS streams. If the account is not 'TTS-Enabled', or the Scene is used within a domain that is not specifically licensed for the account, then this call will have no effect. To enable your account for TTS, please visit the SitePal store. To edit your licensed domains please select Account Info.

Arguments:

txt	Required. String - The text to speak. Most languages are limited to 600 characters for SitePal, 900 characters in Studio. The exceptions are: Chinese & Japanese which are limited to 150 characters in SitePal, 225 characters in Studio. A longer text string 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 listed in Appendix B .

accountId	verified VHSS account id
effect	Optional. Character. Audio effect – one of: <ul style="list-style-type: none"> • “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: <ul style="list-style-type: none"> ○ Bullhorn level: 3 (note: levels 1 and 2 are deprecated) • “T” – Time: <ul style="list-style-type: none"> ○ Echo level: 1 ○ Reverb level: 2 ○ Flanger level: 3 ○ Phase level: 4
effLevel	Optional. Integer. Effect level must be provided if effect is provided.

Examples:

```
sayTextExported('Hello World', 1, 1, 1, xxxx)
sayTextExported('Hello World', 1, 1, 1, xxxx, 'S', -2)
```

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

Available for: Studio SitePal

Note: This call is only supported for Studio customers whose account has been configured to work with exported Scenes. Contact support@oddcast.com for more information.

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.

Notes:

- This function is available only to a TTS Enabled account and will work only within a licensed domain for the account. Domain specific licensing is necessary to avoid misappropriation of TTS streams. If the account is not ‘TTS-Enabled’, or if the Scene is used within a domain that is not approved for the account, then this call will have no effect. To enable your account for TTS, please visit the SitePal store. To edit your licensed domains please select Account Info.
- If you need to customize the AI knowledge base to your needs, please send a note to support@oddcast.com.

Arguments:

txt	Required. String - The text to speak. Most languages are limited to 600 characters for SitePal, 900 characters in Studio. The exceptions are: Chinese & Japanese which are limited to 150 characters in SitePal, 225 characters in Studio. A longer text string 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 listed in Appendix B .
accountId	Verified VHSS account id
botID	Optional. String – The ID of the AI bot to use for generating the response. This parameter is only relevant for VHost Studio AIMC accounts which support multiple bots per account. SitePal users do not need to provide this parameter.
effect	Optional. Character. Audio effect – one of: <ul style="list-style-type: none"> • “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: <ul style="list-style-type: none"> ○ Bullhorn level: 3 (note: levels 1 and 2 are deprecated) • “T” – Time: <ul style="list-style-type: none"> ○ Echo level: 1 ○ Reverb level: 2 ○ Flanger level: 3 ○ Phase level: 4
effLevel	Optional. Integer. Effect level must be provided if effect is provided.

Example:

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

Scene Attributes

setBackground (bgName)

Available for: Studio SitePal

Background is modified for playback session in progress. Change is not persistent.

Arguments:

bgName Required. String. The logical name of the background as specified within the account.

Example:

```
setBackground('background name')
```

setColor (part,color)

Available for: Studio SitePal

Dynamically change the color of the part specified. Colors are based on a gray offset in order to maintain the shading and detail. Therefore, the effect of a color on the specified area may not

exactly match the input hex color value. The results may differ slightly from one character to another.

Arguments:

<code>part</code>	Required. String. The character part to color. One of: 'eyes', 'hair', 'make-up', 'mouth', 'skin'
<code>color</code>	Required. String. Hexadecimal RGB color representation.

Example:

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

setLink (href,target)

Available for: Studio SitePal

Set the URL address for the link of the currently executing scene. For this call to have an effect, the scene should have a specified Trigger to activate the link.

Arguments:

<code>href</code>	Required. String. The URL address for the link.
<code>target</code>	Optional. String . A frame name or a window name An optional parameter specifying the window or HTML frame that the document should load into. For 'Window' you can enter the name of a specific window or choose from the following reserved target names: <code>_self</code> specifies the current frame in the current window <code>_blank</code> specifies a new window <code>_parent</code> specifies the parent of the current frame <code>_top</code> specifies the top-level frame in the current window If no value is specified, default value is " <code>_blank</code> ".

Example:

```
setLink('http://www.yoursite.com', '_blank')
```

setStatus (interruptMode,progressInterval,gazeSpeed)

Available for: Studio SitePal

This function is used to set several status values which govern various aspects of playback.

Arguments:

<code>interruptMode</code>	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.
<code>progressInterval</code>	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

Example:

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

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 the page, the function call will have no effect.

Note: Embed overlay functions are available in JavaScript but not in ActionScript.

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

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

loadScene (sceneIndex)

Available for: Studio SitePal

SitePal ONLY - The current Scene is interrupted, and the specified Scene is loaded instead.

Arguments:

<code>sceneIndex</code>	Required. Integer Indicates the Scene to load.
-------------------------	---

Example

```
loadScene (3)           – load Scene #3
```

loadShow (showIndex)

Available for: Studio SitePal

Studio ONLY - The current Show is interrupted, and the specified Show is loaded instead.

Arguments:

<code>showIndex</code>	Required. Integer Indicates the Show to load.
------------------------	--

Example

```
loadShow (3)           – load Show #3
```

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:

<code>sceneNumber</code>	Required. Integer The index of the scene to preload
--------------------------	--

Example

setNextSceneIndex (sceneRange)

Available for: Studio SitePal

Set the next Scene in the playback order. Actual transition to the specified Scene will not take place until a user clicks on the player's Next button or the function gotoNextScene () is called. Setting the sceneIndex is equivalent of setting the 'Next Scene' attribute in the VHost Show Editor.

Arguments:

SceneRange

Required. String. Indicates the next scene to follow the currently playing scene. Can be either:

- Index of specific scene
- Range of scenes that specify a set of scene indexes. In this case one of the specified set is selected at random. The format is comma delimited, where consecutive ranges can be specified by a hyphen.

Examples:

setNextSceneIndex (3)

– the next scene is a specific scene

setNextSceneIndex (1,4-6,12)

– next scene is a random selection
from the set: 1,4,5,6,12

Status Callback Functions

Status Callback Functions (SCFs) can be useful when embedding a Scene or Show in your web page or Flash application. Using SCFs can help improve coordination between the embedded Scene or Show and your page or application.

SCFs are supported in both Flash movies (ActionScript) and HTML pages (JavaScript). The syntax of the functions is the same in both cases, 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 2.0 (depracated)

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 3.0

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

```
loader:Loader = new Loader();
loader.loaderContentInfo.addEventListener(Event.COMPLETE, setListeners);
loader.load( /* your Scene's flash embed code here */ );
function setListeners():void
{
    MovieClip(loader.content).addEventListener("vh_talkStarted", talkStartedHandler);
    function talkStartedHandler():void{ trace("talk started"); }
}
```

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.

vh_aiResponse ()

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:

response_text Response text

Example - JavaScript & ActionScript2

```
function vh_aiResponse(response_text) {
}
```

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vh_aiResponse", aiResponseHandler);
function aiResponseHandler(event:*):void{
    trace("ai response text: "+ event.data);
}
```

vh_audioLoaded (audioName)

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:

audio_name Loaded audio name

Example - JavaScript & ActionScript2

```
function vh_audioLoaded(audio_name) {
    sayAudio(audio_name);
}
```

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vh_audioLoaded", audioLoadedHandler);
function audioLoadedHandler(event:*):void{
    trace("audio name: "+ event.data);
}
```

vh_ttsLoaded (text)

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:

audio_text Loaded audio text

Example - JavaScript & ActionScript2

```
function vh_ttsLoaded(audio_text) {
    sayText(audio_text, 2, 1, 1);
}
```

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vh_ttsLoaded", ttsLoadedHandler);
function ttsLoadedHandler(event:*):void{
    trace("text to speak: "+ event.data);
}
```

vh_audioProgress (percentPlayed)

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: highlighting 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 vh_audioProgress(percentPlayed) {  
}
```

Example - ActionScript3

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

vh_sceneLoaded (sceneIndex)

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:

`sceneNumber` The number of the loading scene.

Example - JavaScript & ActionScript2

```
function vh_sceneLoaded(sceneIndex) {  
    alert("the scene has started");  
}
```

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vh_sceneLoaded", sceneLoadedHandler);  
function sceneLoadedHandler(event:*):void{  
    trace("scene started. index: "+ event.data);  
}
```

vh_scenePreloaded (sceneIndex)

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:

`none`

Example - JavaScript & ActionScript2

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

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vh_scenePreLoaded", pLoadedHandler);  
function pLoadedHandler(event:*):void{
```

```
        trace("scene preloaded. index: "+ event.data);
    }
```

vh_talkStarted ()

Available for: Studio SitePal

Triggered when the VHost character starts talking

Arguments

caller Path to caller (Flash only). Not in use.

Example - JavaScript & ActionScript2

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

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vh_talkStarted", talkStartedHandler);
function talkStartedHandler():void{
    trace("talk started.");
}
```

vh_talkEnded ()

Available for: Studio SitePal

Triggered when the VHost character is done talking

Arguments

caller Path to caller (Flash only). Not in use.

Example - JavaScript & ActionScript2

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

Example - ActionScript3

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

vh_audioStarted ()

Available for: Studio SitePal

Triggered when audio playback begins. Unlike `vh_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 vh_audioStarted(){
}
```

Example - ActionScript3

```
MovieClip(loader.content).addEventListener("vh_audioStarted",audioStartedHandler);

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

vh_audioEnded ()

Available for: Studio SitePal

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

Example - JavaScript & ActionScript2

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

Example - ActionScript3

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

Appendix A: API Examples

For full source code examples of using the Client API please see:

[Using the Client API in Javascript](#)

[Using the Client API in ActionScript 3 \(Flash 9\)](#)

[Using the Client API in ActionScript 2 \(Flash 8\)](#)

[Advanced Client API examples](#)

Additional reference material can be found in our [support section](#).

Appendix B: Text to Speech Languages and Voices

The following tables list all Voice Family IDs, Language IDs and Voice IDs supported by the sayText & sayAIResponse API calls.

<i>Language</i>	<i>ID</i>
Arabic	27
Basque	22
Catalan	5
Chinese	10
Czech	18
Danish	19
Dutch	11
English	1
Esperanto	31
Finnish	23
French	4
Galician	15
German	3
Greek	8
Hindi	24
Hungarian	29
Icelandic	25
Indonesian	28
Italian	7
Japanese	12
Korean	13
Norwegian	20
Polish	14
Portuguese	6
Romanian	30
Russian	21
Spanish	2
Swedish	9
Thai	26
Turkish	16

TTS Voice Family: AT&T ID = 1

<i>Language</i>	<i>Lang. ID</i>	<i>Voice Name</i>	<i>Voice ID</i>	<i>Gender</i>	<i>Description</i>
English	1	Audrey	5	F	UK
English	1	Charles	6	M	UK
Spanish	2	Rosa	1	F	Latin American
Spanish	2	Alberto	2	M	Latin American
German	3	Klara	1	F	
German	3	Reiner	2	M	
French	4	Juliette	1	F	European
French	4	Alain	2	M	European
French	4	Pierre	3	M	Canadian

TTS Voice Family: Loquendo ID = 2

<i>Language</i>	<i>Lang. ID</i>	<i>Voice Name</i>	<i>Voice ID</i>	<i>Gender</i>	<i>Description</i>	<u><i>Expressive Cues*</i></u>
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	Brazilian	√
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 Loquendo voices, as indicated above. For a complete list of Expressive Cue tags see [Appendix D](#).

TTS Voice Family: NeoSpeech ID = 3

<i>Language</i>	<i>Lang. ID</i>	<i>Voice Name</i>	<i>Voice ID</i>	<i>Gender</i>	<i>Description</i>
English	1	Kate	1	F	US
English	1	Paul	2	M	US
English	1	Julie	3	F	US
English	1	Bridget	4	F	UK
Spanish	2	Violeta	1	F	
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	

TTS Voice Family: Nuance ID = 4

<i>Language</i>	<i>Lang. ID</i>	<i>Voice Name</i>	<i>Voice ID</i>	<i>Gender</i>	<i>Description</i>
English	1	Jennifer	1	F	US
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	Emily	6	F	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	Isabel	2	F	

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	Madalena	1	F	European
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	Ingrid	2	F	
Swedish	9	Oskar	3	M	
Chinese	10	Sin-Ji	1	F	Cantonese
Chinese	10	Ya-Ling	2	F	Taiwanese Mandarin
Chinese	10	Mei-Ling	3	F	Mandarin
Chinese	10	Ting-Ting	4	F	Mandarin
Dutch	11	Ellen	1	F	Belgian
Dutch	11	Clair	2	F	
Dutch	11	Laura	3	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	
Danish	19	Nanna	2	F	
Norwegian	20	Nora	1	F	
Norwegian	20	Stine	2	F	
Russian	21	Katerina	1	F	
Russian	21	Milena	2	F	
Basque	22	Arantxa	1	F	

Finnish	23	Mikko	1	M
Hindi	24	Lekha	1	F
Icelandic	25	Ragga	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

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 syntax 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 TTS Engines available via the VHost API. 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). 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.

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:

```
<Say-as interpret-as="Currency"> $25.32 </say-as> Wrong  
<say-as interpret-as="Currency"> $25.32 </say-as> Correct  
<SAY-AS interpret-as="Currency"> $25.32 </SAY-AS> Correct
```

Say-as Elements

Note – the Neospeech TTS Engine does not support the "interpret-as" attribute, but uses the older (now deprecated) "type" attribute instead. When using a NeoSpeech voice, use "type" instead of "interpret-as".

Example:

AT&T and Loquendo -

```
<say-as interpret-as="Currency"> $25.32 </say-as>
```

NeoSpeech -

```
<say-as type="Currency"> $25.32 </say-as>
```

Currency

The **Currency** context tells the TTS engine to treat the text as currency and expand \$ and decimal numbers appropriately. In the AT&T language family the Currency context works only with US currency and not other currencies. In the Loquendo voice family the "\$" symbol, as well as the following currency indicators are supported: EUR, USD, GBP, JPY.

Syntax: `<say-as interpret-as="Currency"> text </say-as>`

AT&T: US English

Loquendo: English, Spanish, German, French, Italian

Neospeech: Supported (use “type” attribute instead of “interpret-as”)

Example:

```
<say-as interpret-as="Currency"> $25.32 </say-as>  
is pronounced "twenty five dollars and thirty two cents"
```

Example:

```
<say-as interpret-as="Currency">10.32 GBP</say-as>  
is pronounced "10 pounds and thirty two pence"
```

Date:MD

The **Date** context tells the TTS engine to treat the text as date. You may also add qualifiers to provide even more information to the TTS engine but in general the extra qualifiers are not needed.

Syntax: `<say-as interpret-as="Date:MD"> text </say-as>`

AT&T: Supported

Loquendo: Supported

Neospeech: Supported (use “type” attribute instead of “interpret-as”)

Example: `<say-as interpret-as="Date:MD"> Dec 25</say-as>`
is pronounced "December twenty fifth"

Date:MDY

Syntax: `<say-as interpret-as="Date:MDY"> text </say-as>`

AT&T: Supported

Loquendo: Supported

Neospeech: Supported (use “type” attribute instead of “interpret-as”)

Example: `<say-as interpret-as="Date:MDY"> Dec 25, 2001 </say-as>`
is pronounced "December twenty fifth two thousand one"

Date:Y

Syntax: `<say-as interpret-as="Date:Y"> text </say-as>`

AT&T: US English

Loquendo: Supported

Neospeech: Supported (use “type” attribute instead of “interpret-as”)

Example: `<say-as interpret-as="Date:Y"> 2001 </say-as>`
is pronounced "two thousand one"

Number

The **Number** type tells the engine to expect a number.

Syntax: `<say-as interpret-as="Number"> text </say-as>`

AT&T: Supported

Loquendo: Supported

Neospeech: Supported (use “type” attribute instead of “interpret-as”)

Example: `<say-as interpret-as="Number"> 10,243 </say-as>`
is pronounced "ten thousand two hundred forty three"

Number:Decimal

The **Number:Decimal** context tells the TTS engine to treat the text as a decimal number.

Syntax: `<say-as interpret-as="Number:Decimal"> text </say-as>`

AT&T: **Supported**

Loquendo: **Supported**

Neospeech: **Supported (use “type” attribute instead of “interpret-as”)**

Example: `<say-as interpret-as="Number:decimal"> 3.14159 </say-as>`
is pronounced "three point one four one five nine"

Number:Fraction

Syntax: `<say-as interpret-as="Number:Fraction"> text </say-as>`

AT&T: **US English & UK English**

Loquendo: **Supported**

Neospeech: **Supported (use “type” attribute instead of “interpret-as”)**

Example: `<say-as interpret-as="Number:Fraction"> 5 3/4 </say-as>`
is pronounced "five and three fourths"

Telephone

The **Telephone** context tells the TTS engine to treat the text as a telephone number.

Syntax: `<say-as interpret-as="Telephone"> text </say-as>`

AT&T: **US English**

Loquendo: **Supported**

Neospeech: **Supported (use “type” attribute instead of “interpret-as”)**

Example: `<say-as interpret-as="telephone"> (212)555-1212 </say-as>`
is pronounced "two one two five five five one two one two"

Time

The **Time** context tells the TTS engine to treat the text as a time.

Syntax: `<say-as interpret-as="Time"> text </say-as>`

AT&T: **US English**

Loquendo: **Supported**

Neospeech: **Supported (use “type” attribute instead of “interpret-as”)**

Example: `<say-as interpret-as="Time"> 12:34 PM </say-as>`
is pronounced "twelve thirty four P M"

Structure Elements

Break

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

AT&T: **Supported**

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>`

AT&T: **Supported**

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>`

AT&T: **Supported**

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.

AT&T: **Supported**

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..

AT&T: **Supported**

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.

AT&T: Unsupported

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. This feature is currently supported only for the AT&T 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. You need to append "16" to the name of each AT&T voice that you use.

Syntax: `<VOICE`

```
    Gender="male | female | neutral"
    Name="mikel16 | crystal16 | rich16 | rosa16 | reiner16 |
    Klara16 | Audrey16 | charles16 | alain16 | alberto16 |
    Juliette16 | Pierrel16"
</VOICE>
```

AT&T: Supported

Loquendo: Unsupported

Neospeech: Unsupported

Example: `<voice name="mikel16"> This is Mike <Voice Name="rosa16">
 Hola, me llamo Rosa </Voice> This is Mike again. </voice>`

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

Appendix D: Loquendo Expressive Cues

NOTE: This feature is specific to a subset of Loquendo voices.

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 the following subset of Loquendo voices:

- Dutch: Saksia, Willem
- English (UK): Simon, Catherine*
- French: Jolie
- German: Katrin, Stefan
- Italian: Giulia, , Luca, Paola
- Portuguese Eusebio*
- Spanish Carlos*

* These voices have just been added, and although they support Expressive Cues, the detailed list below does not yet include them. This information will be added shortly.

Expressive Cues tags are 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 are the list of Expressive Cue tags supported by each of the voices –

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
_Prff _Prff_01 _Prff_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
_Ttt
_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