

# Package ‘aws.polly’

March 11, 2020

**Type** Package

**Title** Client for AWS Polly

**Version** 0.1.5

**Date** 2020-03-10

**Description** A client for AWS Polly <<http://aws.amazon.com/documentation/polly>>, a speech synthesis service.

**License** GPL (>= 2)

**URL** <https://github.com/cloudyr/aws.polly>

**BugReports** <https://github.com/cloudyr/aws.polly/issues>

**Imports** httr, jsonlite, aws.signature (>= 0.3.4), tuneR

**Encoding** UTF-8

**RoxygenNote** 7.0.2

**NeedsCompilation** no

**Author** Thomas J. Leeper [aut] (<<https://orcid.org/0000-0003-4097-6326>>),  
Antoine Sachet [cre]

**Maintainer** Antoine Sachet <[antoine.sac@gmail.com](mailto:antoine.sac@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-03-11 12:40:05 UTC

## R topics documented:

aws.polly-package . . . . .	2
get_lexicon . . . . .	2
get_synthesis . . . . .	3
list_voices . . . . .	4
pollyHTTP . . . . .	5

<b>Index</b>	<b>7</b>
--------------	----------

---

aws.polly-package	<i>aws.polly</i>
-------------------	------------------

---

**Description**

Client for AWS Polly

**Details**

This is a client for AWS Polly, a speech synthesis service.

**Author(s)**

Thomas J. Leeper <thosjleeper@gmail.com>

**See Also**

[AWS Polly Documentation](#)

---

get_lexicon	<i>Lexicons</i>
-------------	-----------------

---

**Description**

List, put, and delete lexicons

**Usage**

```
get_lexicon(lexicon, token, ...)
```

```
put_lexicon(lexicon, content, ...)
```

```
delete_lexicon(lexicon, ...)
```

**Arguments**

lexicon	A character string specifying the name of a lexicon. If missing, a list of available lexicons is returned.
token	Optionally, a pagination token.
...	Additional arguments passed to <a href="#">pollyHTTP</a> .
content	A character string containing the content of the PLS lexicon.

**Details**

Note: put\_lexicon will overwrite an existing lexicon with the same name.

**Value**

A list.

**Examples**

```
## Not run:  
list_lexicons()  
  
## End(Not run)
```

---

get\_synthesis

*Synthesize Speech*

---

**Description**

Pass text to the synthesis API and return an audio file

**Usage**

```
get_synthesis(  
  text,  
  voice,  
  format = c("mp3", "ogg_vorbis", "pcm"),  
  rate = c(22050, 16000, 8000),  
  lexicon = NULL,  
  ssml = FALSE,  
  ...  
)  
  
synthesize(text, voice, ...)
```

**Arguments**

text	Either a plain text character string (maximum 1500 characters) or a character string containing SSML (ssml should be set to TRUE).
voice	A character string specifying the name of an AWS Polly voice. See <a href="#">list_voices</a> .
format	A character string specifying an output file format.
rate	An integer value specifying the audio frequency in Hertz.
lexicon	Optionally, a character vector (max length 5) specifying the names of lexicons to apply during synthesis. See <a href="#">get_lexicon</a> .
ssml	A logical indicating whether text contains SSML markup.
...	Additional arguments passed to <a href="#">pollyHTTP</a> .

**Value**

get\_synthesis returns a raw vector (i.e., the bytes representing the audio as the requested file format). synthesize is a convenience wrapper around that, which returns an object of class “Wave” (see [Wave](#)).

**Examples**

```
## Not run:
hello <- synthesize("hello world!", voice = "Geraint")
if (interactive() & require("tuneR")) {
  try(play(hello))
}

## End(Not run)
```

---

list_voices	<i>List available voices</i>
-------------	------------------------------

---

**Description**

Retrieve a list of available voices

**Usage**

```
list_voices(language = "en-US", token, ...)
```

**Arguments**

language	An ISO 3166 country identification tag.
token	Optionally, a pagination token.
...	Additional arguments passed to <a href="#">pollyHTTP</a> .

**Value**

A data frame of available names.

**Examples**

```
## Not run:
list_voices(language = "cy-GB")

## End(Not run)
```

**Description**

This is the workhorse function to execute calls to the Polly API.

**Usage**

```
pollyHTTP(  
  action,  
  query = list(),  
  headers = list(),  
  body = NULL,  
  verb = c("GET", "POST", "PUT", "DELETE"),  
  version = "v1",  
  raw_response = if (verb == "POST") TRUE else FALSE,  
  verbose = getOption("verbose", FALSE),  
  region = Sys.getenv("AWS_DEFAULT_REGION", "us-east-1"),  
  key = NULL,  
  secret = NULL,  
  session_token = NULL,  
  ...  
)
```

**Arguments**

<code>action</code>	A character string specifying the API action to take
<code>query</code>	An optional named list containing query string parameters and their character values.
<code>headers</code>	A list of headers to pass to the HTTP request.
<code>body</code>	A request body
<code>verb</code>	A character string specifying the HTTP verb to implement.
<code>version</code>	A character string specifying the API version.
<code>raw_response</code>	A logical indicating whether to return the raw response body.
<code>verbose</code>	A logical indicating whether to be verbose. Default is given by <code>options("verbose")</code> .
<code>region</code>	A character string specifying an AWS region. See <a href="#">locate_credentials</a> .
<code>key</code>	A character string specifying an AWS Access Key. See <a href="#">locate_credentials</a> .
<code>secret</code>	A character string specifying an AWS Secret Key. See <a href="#">locate_credentials</a> .
<code>session_token</code>	Optionally, a character string specifying an AWS temporary Session Token to use in signing a request. See <a href="#">locate_credentials</a> .
<code>...</code>	Additional arguments passed to <a href="#">GET</a> .

**Details**

This function constructs and signs an Polly API request and returns the results thereof, or relevant debugging information in the case of error.

**Value**

If successful, a named list. Otherwise, a data structure of class “aws-error” containing any error message(s) from AWS and information about the request attempt.

**Author(s)**

Thomas J. Leeper

# Index

## \*Topic **package**

aws.polly-package, 2

aws.polly (aws.polly-package), 2

aws.polly-package, 2

delete\_lexicon (get\_lexicon), 2

GET, 5

get\_lexicon, 2, 3

get\_synthesis, 3

list\_voices, 3, 4

locate\_credentials, 5

pollyHTTP, 2-4, 5

put\_lexicon (get\_lexicon), 2

synthesize (get\_synthesis), 3

Wave, 4