

# Package ‘nzilbb.labbcats’

February 2, 2023

**Version** 1.2-0

**Date** 2023-01-19

**Title** Accessing Data Stored in 'LaBB-CAT' Instances

**Imports** jsonlite, httr, stringr, utils, rstudioapi

**Description** 'LaBB-CAT' is a web-based language corpus management system developed by the New Zealand Institute of Language, Brain and Behaviour (NZILBB) - see <<https://labbcats.canterbury.ac.nz>>. This package defines functions for accessing corpus data in a 'LaBB-CAT' instance. You must have at least version 20230202.1600 of 'LaBB-CAT' to use this package.

For more information about 'LaBB-CAT', see Robert Fromont and Jennifer Hay (2008) <[doi:10.3366/E1749503208000142](https://doi.org/10.3366/E1749503208000142)> or Robert Fromont (2017) <[doi:10.1016/j.csl.2017.01.004](https://doi.org/10.1016/j.csl.2017.01.004)>.

**License** GPL (>= 3)

**Copyright** New Zealand Institute of Language, Brain and Behaviour, University of Canterbury

**URL** <https://nzilbb.github.io/labbcats-R/>,  
<https://labbcats.canterbury.ac.nz>

**RoxygenNote** 7.2.3

**Suggests** testthat (>= 2.1.0)

**NeedsCompilation** no

**Author** Robert Fromont [aut, cre] (<<https://orcid.org/0000-0001-5271-5487>>)

**Maintainer** Robert Fromont <[robert.fromont@canterbury.ac.nz](mailto:robert.fromont@canterbury.ac.nz)>

**Repository** CRAN

**Date/Publication** 2023-02-02 22:00:02 UTC

## R topics documented:

addDictionaryEntry . . . . . 3

addLayerDictionaryEntry . . . . .	4
annotatorExt . . . . .	5
countAnnotations . . . . .	6
countMatchingAnnotations . . . . .	7
deleteLayer . . . . .	8
deleteLexicon . . . . .	9
deleteTranscript . . . . .	10
formatTranscript . . . . .	10
generateLayer . . . . .	12
generateLayerUtterances . . . . .	13
getAllUtterances . . . . .	14
getAnchors . . . . .	15
getAnnotations . . . . .	16
getAnnotatorDescriptor . . . . .	18
getAvailableMedia . . . . .	19
getCorpusIds . . . . .	20
getDeserializerDescriptors . . . . .	21
getDictionaries . . . . .	22
getDictionaryEntries . . . . .	22
getFragmentAnnotations . . . . .	23
getFragments . . . . .	25
getGraphIds . . . . .	26
getGraphIdsInCorpus . . . . .	27
getGraphIdsWithParticipant . . . . .	28
getId . . . . .	28
getLayer . . . . .	29
getLayerIds . . . . .	30
getLayers . . . . .	31
getMatchAlignments . . . . .	32
getMatches . . . . .	33
getMatchingAnnotations . . . . .	37
getMatchingGraphIds . . . . .	39
getMatchingParticipantIds . . . . .	41
getMatchingTranscriptIds . . . . .	42
getMatchLabels . . . . .	44
getMedia . . . . .	45
getMediaTracks . . . . .	46
getMediaUrl . . . . .	47
getParticipantAttributes . . . . .	48
getParticipantIds . . . . .	49
getSerializerDescriptors . . . . .	49
getSoundFragments . . . . .	50
getSystemAttribute . . . . .	52
getTranscriptAttributes . . . . .	52
getTranscriptIds . . . . .	53
getTranscriptIdsInCorpus . . . . .	54
getTranscriptIdsWithParticipant . . . . .	55
getUserInfo . . . . .	55

labbbcatCredentials . . . . .	56
labbbcatTimeout . . . . .	57
loadLexicon . . . . .	58
newLayer . . . . .	59
newTranscript . . . . .	61
nzilbb.labbbcat . . . . .	62
praatScriptCentreOfGravity . . . . .	66
praatScriptFastTrack . . . . .	67
praatScriptFormants . . . . .	70
praatScriptIntensity . . . . .	72
praatScriptPitch . . . . .	73
processWithPraat . . . . .	76
removeDictionaryEntry . . . . .	78
removeLayerDictionaryEntry . . . . .	79
saveLayer . . . . .	80
updateFragment . . . . .	82
updateTranscript . . . . .	83

<b>Index</b>	<b>84</b>
--------------	-----------

---

addDictionaryEntry	<i>Adds an entry to a dictionary.</i>
--------------------	---------------------------------------

---

## Description

This function creates adds a new entry to the given dictionary.

## Usage

```
addDictionaryEntry(labbbcat.url, manager.id, dictionary.id, key, entry)
```

## Arguments

labbbcat.url	URL to the LaBB-CAT instance
manager.id	The layer manager ID of the dictionary, as returned by getDictionaries
dictionary.id	The ID of the dictionary, as returned by getDictionaries
key	The key (word) in the dictionary to add an entry for.
entry	The value (definition) for the given key.

## Details

You must have edit privileges in LaBB-CAT in order to be able to use this function.

## Value

NULL if the entry was added, or a list of error messages if not.

**See Also**[getDictionaries](#)[getDictionaryEntries](#)**Examples**

```
## Not run:  
## Add the word "robert" to the CELEX wordform pronunciation dictionary  
addDictionaryEntry(labbcat.url, "CELEX-EN", "Phonology (wordform)", "robert", "'rQ-b@t")  
  
## End(Not run)
```

---

`addLayerDictionaryEntry`*Adds an entry to a layer dictionary.*

---

**Description**

This function adds a new entry to the dictionary that manages a given layer, and updates all affected tokens in the corpus. Words can have multiple entries.

**Usage**

```
addLayerDictionaryEntry(labbcat.url, layer.id, key, entry)
```

**Arguments**

<code>labbcat.url</code>	URL to the LaBB-CAT instance
<code>layer.id</code>	The ID of the layer with a dictionary configured to manage it.
<code>key</code>	The key (word) in the dictionary to add an entry for.
<code>entry</code>	The value (definition) for the given key.

**Details**

You must have edit privileges in LaBB-CAT in order to be able to use this function.

**Value**

NULL if the entry was added, or a list of error messages if not.

**See Also**[generateLayer](#)

**Examples**

```
## Not run:
## Add a pronunciation for the word "robert" to the phonemes layer dictionary
addLayerDictionaryEntry(labbcats.url, "phonemes", "robert", "'rQ-b@t")

## End(Not run)
```

---

annotatorExt

*Retrieve annotator's "ext" resource.*


---

**Description**

Retrieve a given resource from an annotator's "ext" web app. Annotators are modules that perform different annotation tasks, and can optionally implement functionality for providing extra data or extending functionality in an annotator-specific way. If the annotator implements an "ext" web app, it can provide resources and implement a mechanism for interrogating the annotator. This function provides a mechanism for accessing these resources via R.

**Usage**

```
annotatorExt(labbcats.url, annotator.id, resource, parameters = NULL)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance.
annotator.id	ID of the annotator to interrogate.
resource	The name of the file to retrieve or instance method (function) to invoke. Possible values for this depend on the specific annotator being interrogated.
parameters	Optional list of ordered parameters for the instance method (function).

**Value**

The resource requested.

**Examples**

```
## Not run:
## Get the version of the currently installed LabelMapper annotator:
annotatorExt(labbcats.url, "LabelMapper", "getVersion")

## Get the summary of the segment to speakerDependentPhone mapping
## implemented by the LabelMapper annotator:
summaryJson <- annotatorExt(labbcats.url,
  "LabelMapper", "summarizeMapping", list("segment", "speakerDependentPhone"))
summary <- jsonlite::fromJSON(summaryJson)

## End(Not run)
```

---

countAnnotations	<i>Gets the number of annotations on the given layer of the given transcript.</i>
------------------	---

---

### Description

Returns the number of annotations on the given layer of the given transcript.

### Usage

```
countAnnotations(labbcats.url, id, layer.id, max.ordinal = NULL)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
id	A transcript ID (i.e. transcript name)
layer.id	A layer ID
max.ordinal	The maximum ordinal for the counted annotations. e.g. a max.ordinal of 1 will ensure that only the first annotation for each parent is returned. If max.ordinal is null, then all annotations are counted, regardless of their ordinal.

### Value

The number of annotations on that layer

### See Also

[getTranscriptIds](#) [getTranscriptIdsInCorpus](#) [getTranscriptIdsWithParticipant](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Count the number of words in UC427_ViktoriaPapp_A_ENG.eaf
token.count <- countAnnotations(labbcats.url, "UC427_ViktoriaPapp_A_ENG.eaf", "orthography")

## End(Not run)
```

---

`countMatchingAnnotations`*Gets the number of annotations matching a particular pattern.*

---

## Description

Returns the number of annotations in the corpus that match the given expression.

## Usage

```
countMatchingAnnotations(labbcat.url, expression)
```

## Arguments

<code>labbcat.url</code>	URL to the LaBB-CAT instance
<code>expression</code>	An expression that determines which annotations match. This must match by either <code>id</code> or <code>layer.id</code> . The expression language is currently not well defined, but is based on JavaScript syntax. e.g. <ul style="list-style-type: none"><li><code>id == 'ew_0_456'</code></li><li><code>['ew_2_456', 'ew_2_789', 'ew_2_101112'].includes(id)</code></li><li><code>layerId == 'orthography' &amp;&amp; !/th[aeiou].+/.test(label)</code></li><li><code>graph.id == 'AdaAicheson-01.trs' &amp;&amp; layer.id == 'orthography' &amp;&amp; start.offset &gt; 10.5</code></li><li><code>layer.id == 'utterance' &amp;&amp; all('word').includes('ew_0_456')</code></li><li><code>layerId = 'utterance' &amp;&amp; labels('orthography').includes('foo')</code></li><li><code>layerId = 'utterance' &amp;&amp; labels('participant').includes('Ada')</code></li></ul>

## Value

The number of annotations that match the expression.

## See Also

[getMatchingAnnotations](#)

## Examples

```
## Not run:
## define the LaBB-CAT URL
labbcat.url <- "https://labbcat.canterbury.ac.nz/demo/"

## count the number of topic tags that include the word 'quake'
countMatchingAnnotations(labbcat.url, "layer.id == 'topic' && /. *quake.* /.test(label)")

## End(Not run)
```

---

deleteLayer	<i>Deletes an existing layer.</i>
-------------	-----------------------------------

---

### Description

This function deletes an existing annotation layer, including all annotation data associated with it.

### Usage

```
deleteLayer(labbcat.url, layer.id)
```

### Arguments

labbcat.url	URL to the LaBB-CAT instance
layer.id	The ID of the layer to delete.

### Details

You must have administration privileges in LaBB-CAT in order to be able to use this function.

### Value

NULL, or an error message if deletion failed.

### See Also

[newLayer](#) [saveLayer](#)

### Examples

```
## Not run:  
## Delete the phonemes layer  
deleteLayer(labbcat.url, "phonemes")  
  
## End(Not run)
```

---

deleteLexicon	<i>Delete a previously loaded lexicon.</i>
---------------	--

---

### Description

By default LaBB-CAT includes a layer manager called the Flat Lexicon Tagger, which can be configured to annotate words with data from a dictionary loaded from a plain text file (e.g. a CSV file).

### Usage

```
deleteLexicon(labbcat.url, lexicon)
```

### Arguments

labbcat.url	URL to the LaBB-CAT instance.
lexicon	The name of the lexicon to delete, e.g. 'cmudict'

### Details

This function deletes such a lexicon, which was previously added using loadLexicon.

You must have editing privileges in LaBB-CAT in order to be able to use this function.

### Value

NULL if the deletion was successful, or an error message if not.

### See Also

[loadLexicon](#)

### Examples

```
## Not run:  
## Delete the previously loaded CMU Pronouncing Dictionary lexicon  
deleteLexicon(labbcat.url, "cmudict")  
  
## End(Not run)
```

---

deleteTranscript      *Delete a transcript from the corpus.*

---

**Description**

This function deletes the given transcript, and all associated files.

**Usage**

```
deleteTranscript(labbcats.url, id)
```

**Arguments**

labbcats.url      URL to the LaBB-CAT instance  
id                The ID transcript to delete.

**Details**

For this function to work, the credentials used to connect to the server must have at least 'edit' access.

**Value**

The ID of the deleted transcript

**Examples**

```
## Not run:  
## define the LaBB-CAT URL  
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"  
  
## delete a transcript from the server  
deleteTranscript(labbcats.url, "my-transcript.eaf")  
  
## End(Not run)
```

---

formatTranscript      *Gets transcript(s) in a given format.*

---

**Description**

This function gets whole transcripts from 'LaBB-CAT', converted to a given format (by default, Praat TextGrid).

**Usage**

```
formatTranscript(
  labbcat.url,
  id,
  layer.ids,
  mime.type = "text/praat-textgrid",
  path = ""
)
```

**Arguments**

labbcat.url	URL to the LaBB-CAT instance
id	The transcript ID (transcript name) of the sound recording, or a vector of transcript IDs. If the same ID appears more than one, the formatted file is downloaded only once.
layer.ids	A vector of layer IDs.
mime.type	Optional content-type - "text/praat-textgrid" is the default, but your LaBB-CAT installation may support other formats, which can be discovered using <a href="#">getSerializerDescriptors</a> .
path	Optional path to directory where the files should be saved.

**Details**

*NB* Although many formats will generate exactly one file for each interval (e.g. mime.type=text/praat-textgrid), this is not guaranteed; some formats generate a single file or a fixed collection of files regardless of how many IDs there are.

**Value**

The name of the file, which is saved in the current directory, or the given path, or a list of names of files, if multiple id's were specified.

If a list of files is returned, they are in the order that they were returned by the server, which *should* be the order that they were specified in the id list.

**See Also**

[getSerializerDescriptors](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcat.url <- "https://labbcat.canterbury.ac.nz/demo/"

## Get the TextGrid of a recording
textgrid.file <- formatTranscript(labbcat.url, "AP2505_Nelson.eaf",
  c("transcript", "phonemes"), path="textgrids")
```

```
## Get all the transcripts of a given participant
transcript.ids <- getTranscriptIdsWithParticipant(labbcats.url, "AP2505_Nelson")

## Download all the TextGrids, including the utterances, transcript, and segment layers
textgrid.files <- formatTranscript(
  labbcats.url, transcript.ids, c("utterance", "word", "segment"))

## End(Not run)
```

---

generateLayer	<i>Generates a layer.</i>
---------------	---------------------------

---

### Description

Generates annotations on a given layer for all transcripts in the corpus.

### Usage

```
generateLayer(labbcats.url, layer.id, no.progress = FALSE)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
layer.id	The ID of the layer to generate.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

### Value

The final status of the layer generation task.

### See Also

[getAllUtterances](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Generate all phonemic transcription annotations
generateLayer(labbcats.url, "phonemes")

## End(Not run)
```

---

`generateLayerUtterances`*Generates a layer for a given set of utterances.*

---

### Description

Generates annotations on a given layer for a given set of utterances, e.g. force-align selected utterances of a participant.

### Usage

```
generateLayerUtterances(  
  labbcats.url,  
  match.ids,  
  layer.id,  
  collection.name = NULL,  
  no.progress = FALSE  
)
```

### Arguments

<code>labbcats.url</code>	URL to the LaBB-CAT instance
<code>match.ids</code>	A vector of annotation IDs, e.g. the MatchId column, or the URL column, of a results set.
<code>layer.id</code>	The ID of the layer to generate.
<code>collection.name</code>	An optional name for the collection, e.g. the participant ID.
<code>no.progress</code>	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when <code>interactive()</code> .

### Value

The final status of the layer generation task.

### See Also

[getAllUtterances](#)

### Examples

```
## Not run:  
## define the LaBB-CAT URL  
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"  
  
## Get all utterances of a participant  
allUtterances <- getAllUtterances(labbcats.url, "AP2505_Nelson")
```

```
## Force-align the participant's utterances
generateLayerUtterances(labbcats.url, allUtterances$MatchId, "htk", "AP2505_Nelson")

## End(Not run)
```

---

getAllUtterances	<i>Get all utterances of participants.</i>
------------------	--

---

### Description

Identifies all utterances of a given set of participants.

### Usage

```
getAllUtterances(
  labbcats.url,
  participant.ids,
  transcript.types = NULL,
  main.participant = TRUE,
  max.matches = NULL,
  no.progress = FALSE
)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
participant.ids	A list of participant IDs to identify the utterances of.
transcript.types	An optional list of transcript types to limit the results to. If null, all transcript types will be searched.
main.participant	TRUE to search only main-participant utterances, FALSE to search all utterances.
max.matches	The maximum number of matches to return, or null to return all.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

### Value

A data frame identifying matches, containing the following columns:

- *SearchName* A name based on the pattern – the same for all rows
- *Number* Row number
- *Transcript* Name of the transcript in which the match was found

- *Line* The start offset of the utterance/line
- *LineEnd* The end offset of the utterance/line
- *MatchId* A unique ID for the matching target token
- *Before.Match* Transcript text immediately before the match
- *Text* Transcript text of the match
- *After.Match* Transcript text immediately after the match
- *Target.word* Text of the target word token
- *Target.word.start* Start offset of the target word token
- *Target.word.end* End offset of the target word token
- *Target.segment* Label of the target segment (only present if the segment layer is included in the pattern)
- *Target.segment.start* Start offset of the target segment (only present if the segment layer is included in the pattern)
- *Target.segment.end* End offset of the target segment (only present if the segment layer is included in the pattern)

### See Also

[getParticipantIds](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## get all utterances of the given participants
participant.ids <- getParticipantIds(labbcats.url)[1:3]
results <- getAllUtterances(labbcats.url, participant.ids)

## results$MatchId can be used to access results

## End(Not run)
```

---

getAnchors

*Gets the given anchors in the given transcript.*

---

### Description

Lists the given anchors in the given transcript.

### Usage

```
getAnchors(labbcats.url, id, anchor.id, page.length = 1000)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
id	A transcript ID (i.e. transcript name)
anchor.id	A vector of anchor IDs (or a string representing one anchor ID)
page.length	In order to prevent timeouts when there are a large number of matches or the network connection is slow, rather than retrieving anchors in one big request, they are retrieved using many smaller requests. This parameter controls the number of anchors retrieved per request.

**Value**

A named list of anchors, with members:

- *id* The annotation's unique ID,
- *offset* The offset from the beginning (in seconds if it's a transcript of a recording, or in characters if it's a text document)
- *confidence* A rating from 0-100 of the confidence of the offset, e.g. 10: default value, 50: force-aligned, 100: manually aligned

**See Also**

[getAnnotations](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get the first 20 orthography tokens in UC427_ViktoriaPapp_A_ENG.eaf
orthography <- getAnnotations(labbcats.url, "UC427_ViktoriaPapp_A_ENG.eaf", "orthography", 20, 0)

## Get the start anchors for the above tokens
word.starts <- getAnchors(labbcats.url, "UC427_ViktoriaPapp_A_ENG.eaf", orthography$startId)

## End(Not run)
```

---

getAnnotations

*Gets the annotations on the given layer of the given transcript.*

---

**Description**

Returns the annotations on the given layer of the given transcript.

**Usage**

```
getAnnotations(
  labbcats.url,
  id,
  layer.id,
  max.ordinal = NULL,
  page.length = NULL,
  page.number = NULL
)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
id	A transcript ID (i.e. transcript name)
layer.id	A layer ID
max.ordinal	The maximum ordinal for the returned annotations. e.g. a max.ordinal of 1 will ensure that only the first annotation for each parent is returned. If max.ordinal is null, then all annotations are returned, regardless of their ordinal.
page.length	The maximum number of annotations to return, or null to return all
page.number	The zero-based page number to return, or null to return the first page

**Value**

A named list of annotations, with members:

- *id* The annotation's unique ID
- *layerId* The name of the layer it comes from
- *label* The value of the annotation
- *startId* The ID of the start anchor,
- *endId* The ID of the end anchor,
- *parentId* The ID of the parent annotation,
- *ordinal* The ordinal of the annotation among its peers,
- *confidence* A rating from 0-100 of the confidence of the label e.g. 10: default value, 50: automatically generated, 100: manually annotated

**See Also**

[getTranscriptIds](#) [getTranscriptIdsInCorpus](#) [getTranscriptIdsWithParticipant](#) [countAnnotations](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get all the orthography tokens in UC427_ViktoriaPapp_A_ENG.eaf
```

```

orthography <- getAnnotations(labbcats.url, "UC427_ViktoriaPapp_A_ENG.eaf", "orthography")

## Get the first 20 orthography tokens in UC427_ViktoriaPapp_A_ENG.eaf
orthography <- getAnnotations(labbcats.url, "UC427_ViktoriaPapp_A_ENG.eaf", "orthography", 20, 0)

## End(Not run)

```

---

`getAnnotatorDescriptor`

*Gets annotator information.*

---

### Description

Retrieve information about an annotator. Annotators are modules that perform different annotation tasks. This function provides information about a given annotator, for example the currently installed version of the module, what configuration parameters it requires, etc.

### Usage

```
getAnnotatorDescriptor(labbcats.url, annotator.id)
```

### Arguments

`labbcats.url` URL to the LaBB-CAT instance.  
`annotator.id` ID of the annotator module.

### Value

The annotator info:

- *annotatorId* The annotators's unique ID
- *version* The currently install version of the annotator.
- *info* HTML-encoded description of the function of the annotator.
- *infoText* A plain text version of `$info` (converted automatically).
- *hasConfigWebapp* Determines whether the annotator includes a web-app for installation or general configuration.
- *configParameterInfo* An HTML-encoded definition of the installation config parameters, including a list of all parameters, and the encoding of the parameter string.
- *configParameterInfoText* A plain text version of `$configParameterInfo` (converted automatically).
- *hasTaskWebapp* Determines whether the annotator includes a web-app for task parameter configuration.
- *taskParameterInfo* An HTML-encoded definition of the task parameters, including a list of all parameters, and the encoding of the parameter string.

- *taskParameterInfoText* A plain text version of `$taskParameterInfo` (converted automatically).
- *hasExtWebapp* Determines whether the annotator includes an extras web-app which implements functionality for providing extra data or extending functionality in an annotator-specific way.
- *extApiInfo* An HTML-encoded document containing information about what endpoints are published by the ext web-app.
- *extApiInfoText* A plain text version of `$extApiInfo` (converted automatically).

### See Also

[annotatorExt newLayer](#)

### Examples

```
## Not run:
## Get information about the BAS Annotator
basAnnotator <- getAnnotatorDescriptor("https://labbcat.canterbury.ac.nz/demo/", "BASAnnotator")
cat(basAnnotator$infoText)

## End(Not run)
```

---

getAvailableMedia      *List the media available for the given transcript.*

---

### Description

List the media available for the given transcript.

### Usage

```
getAvailableMedia(labbcat.url, id)
```

### Arguments

labbcat.url	URL to the LaBB-CAT instance
id	A transcript ID (i.e. transcript name)

### Value

A named list of media files available for the given transcript, with members:

- *trackSuffix* The track suffix of the media
- *contentType* The MIME type of the file
- *url* URL to the content of the file
- *name* Name of the file

**See Also**[getTranscriptIds](#)**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## List the media files available for BR2044_01ly0h1son.eaf
media <- getAvailableMedia(labbcats.url, "BR2044_01ly0h1son.eaf")

## End(Not run)
```

---

getCorpusIds	<i>Gets a list of corpus IDs.</i>
--------------	-----------------------------------

---

**Description**

Returns a list of corpora in the given 'LaBB-CAT' instance.

**Usage**

```
getCorpusIds(labbcats.url)
```

**Arguments**

labbcats.url      URL to the LaBB-CAT instance

**Value**

A list of corpus IDs

**Examples**

```
## Not run:
## List corpora
corpora <- getCorpusIds("https://labbcats.canterbury.ac.nz/demo/")

## End(Not run)
```

---

`getDeserializerDescriptors`

*Lists the descriptors of all registered deserializers.*

---

### Description

Returns a list of deserializers, which are modules that import transcriptions and annotation structures from a specific file format, e.g. Praat TextGrid, plain text, etc.

### Usage

```
getDeserializerDescriptors(labbcat.url)
```

### Arguments

`labbcat.url`      URL to the LaBB-CAT instance

### Value

A list of serializers, each including the following information:

- *name* The name of the format.
- *version* The installed version of the serializer module.
- *fileSuffixes* The normal file name suffixes (extensions) of the files.,
- *mimeType* The MIME type of the format, i.e. the value to use as the *mimeType* parameter of [getFragments](#),

### Examples

```
## Not run:
## List file upload formats supported
formats <- getDeserializerDescriptors("https://labbcat.canterbury.ac.nz/demo/")

## can we upload as plain text?
plainTextSupported <- "text/plain" %in% formats$mimeType

## End(Not run)
```

---

getDictionaries      *List the dictionaries available.*

---

**Description**

List the dictionaries available.

**Usage**

```
getDictionaries(labbcat.url)
```

**Arguments**

labbcat.url      URL to the LaBB-CAT instance

**Value**

A named list of layer manager IDs, each of which containing a list of dictionaries that the layer manager makes available.

**See Also**

[getDictionaryEntries](#)

**Examples**

```
## Not run:  
## List the dictionaries available  
dictionaries <- getDictionaries("https://labbcat.canterbury.ac.nz/demo/")  
  
## End(Not run)
```

---

getDictionaryEntries      *Lookup entries in a dictionary.*

---

**Description**

Lookup entries in a dictionary.

**Usage**

```
getDictionaryEntries(labbcat.url, manager.id, dictionary.id, keys)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
manager.id	The layer manager ID of the dictionary, as returned by getDictionaries
dictionary.id	The ID of the dictionary, as returned by getDictionaries
keys	A list of keys (words) identifying entries to look up

**Value**

A data frame with the keys and their dictionary entries, if any.

**See Also**

[getDictionaries](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

keys <- c("the", "quick", "brown", "fox")

## get the pronunciations according to CELEX
entries <- getDictionaryEntries(labbcats.url, "CELEX-EN", "Phonology (wordform)", keys)

## End(Not run)
```

---

getFragmentAnnotations

*Gets annotations in fragments.*

---

**Description**

This function gets annotations between given start/end times on given layers. If more than one annotation matches, labels are concatenated together.

**Usage**

```
getFragmentAnnotations(
  labbcats.url,
  transcript.id,
  participant.id,
  start,
  end,
  layer.ids,
  sep = " ",
```

```

    partial.containment = FALSE,
    no.progress = FALSE
  )

```

### Arguments

<code>labbcats.url</code>	URL to the LaBB-CAT instance
<code>transcript.id</code>	The transcript ID (transcript name) of the sound recording, or a vector of transcript IDs.
<code>participant.id</code>	The participant ID of the annotations, or a vector of participant IDs.
<code>start</code>	The start time in seconds, or a vector of start times.
<code>end</code>	The end time in seconds, or a vector of end times.
<code>layer.ids</code>	A vector of layer IDs.
<code>sep</code>	The separator to use when concatenating labels when multiple annotations are in the given interval.
<code>partial.containment</code>	Whether to include annotations that are only partially contained in the given interval.
<code>no.progress</code>	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when <code>interactive()</code> .

### Value

A data frame with three columns for each layer in `layer.ids`:

- The annotation labels concatenated together
- The start time of the first annotation
- The end time of the last annotation

### See Also

[getFragments](#)

[getSoundFragments](#)

### Examples

```

## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get some span-layer intervals
topics <- getMatches(labbcats.url, list(topic = ".*quake.*"))

## Get concatenated word tokens for each topic annotation
topic.tokens <- getFragmentAnnotations(
  labbcats.url, topics$Transcript, topics$Participant, topics$topic.start, topics$topic.end,
  c("word"))

## End(Not run)

```

---

getFragments	<i>Gets transcript fragments in a given format.</i>
--------------	---

---

### Description

This function gets fragments of transcripts from 'LaBB-CAT', converted to a given format (by default, Praat TextGrid).

### Usage

```
getFragments(  
  labbcats.url,  
  id,  
  start,  
  end,  
  layer.ids,  
  mime.type = "text/praat-textgrid",  
  path = ""  
)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
id	The transcript ID (transcript name) of the sound recording, or a vector of transcript IDs.
start	The start time in seconds, or a vector of start times.
end	The end time in seconds, or a vector of end times.
layer.ids	A vector of layer IDs.
mime.type	Optional content-type - "text/praat-textgrid" is the default, but your LaBB-CAT installation may support other formats, which can be discovered using <a href="#">getSerializerDescriptors</a> .
path	Optional path to directory where the files should be saved.

### Details

*NB* Although many formats will generate exactly one file for each interval (e.g. mime.type=text/praat-textgrid), this is not guaranteed; some formats generate a single file or a fixed collection of files regardless of how many fragments there are.

### Value

The name of the file, which is saved in the current directory, or a list of names of files, if multiple id's/start's/end's were specified

If a list of files is returned, they are in the order that they were returned by the server, which *should* be the order that they were specified in the id/start/end lists.

**See Also**[getSerializerDescriptors](#)**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get the 5 seconds starting from 10s after the beginning of a recording
textgrid.files <- getFragments(labbcats.url, "AP2505_Nelson.eaf", 10.0, 15.0,
  c("transcript", "phonemes"), path="samples")

## Load some search results previously exported from LaBB-CAT
results <- read.csv("results.csv", header=T)

## Get a list of fragment TextGrids, including the utterances, transcript, and phonemes layers
textgrid.files <- getFragments(
  labbcats.url, results$Transcript, results$Line, results$LineEnd,
  c("utterance", "word", "phonemes"))

## Get a list of fragment TextGrids
textgrid.files <- getFragments(
  labbcats.url, results$Transcript, results$Line, results$LineEnd)

## End(Not run)
```

---

`getGraphIds`*Deprecated synonym for `getTranscriptIds`.*

---

**Description**

Returns a list of graph IDs (i.e. transcript names).

**Usage**

```
getGraphIds(labbcats.url)
```

**Arguments**

`labbcats.url` URL to the LaBB-CAT instance

**Value**

A list of graph IDs

**See Also**[getTranscriptIds](#)

### Examples

```
## Not run:  
## List all transcripts  
transcripts <- getGraphIds("https://labbcats.canterbury.ac.nz/demo/")  
  
## End(Not run)
```

---

`getGraphIdsInCorpus`    *Deprecated synonym for `getTranscriptIdsInCorpus`.*

---

### Description

Returns a list of corpora in the given 'LaBB-CAT' instance.

### Usage

```
getGraphIdsInCorpus(labbcats.url, id)
```

### Arguments

<code>labbcats.url</code>	URL to the LaBB-CAT instance
<code>id</code>	The ID (name) of the corpus

### Value

A list of corpus IDs

### See Also

[getGraphIdsInCorpus](#)

### Examples

```
## Not run:  
## List transcripts in the QB corpus  
transcripts <- getGraphIdsInCorpus("https://labbcats.canterbury.ac.nz/demo/", "QB")  
  
## End(Not run)
```

getGraphIdsWithParticipant

*Deprecated synonym for getTranscriptIdsWithParticipant.*

---

### Description

Returns a list of IDs of graphs (i.e. transcript names) that include the given participant.

### Usage

```
getGraphIdsWithParticipant(labbcats.url, id)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
id	A participant ID

### Value

A list of graph IDs

### See Also

[getTranscriptIdsWithParticipant](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## List transcripts in which UC427_ViktoriaPapp_A_ENG speaks
transcripts <- getGraphIdsWithParticipant(labbcats.url, "UC427_ViktoriaPapp_A_ENG")

## End(Not run)
```

---

getId

*Gets the store's ID.*

---

### Description

The store's ID - i.e. the ID of the 'LaBB-CAT' instance.

### Usage

```
getId(labbcats.url)
```

**Arguments**

labbcat.url      URL to the LaBB-CAT instance

**Value**

The annotation store's ID

**Examples**

```
## Not run:
## Get ID of LaBB-CAT instance
instance.id <- getId("https://labbcat.canterbury.ac.nz/demo/")

## End(Not run)
```

---

getLayer	<i>Gets a layer definition.</i>
----------	---------------------------------

---

**Description**

Gets a layer definition.

**Usage**

```
getLayer(labbcat.url, id)
```

**Arguments**

labbcat.url      URL to the LaBB-CAT instance  
id                ID of the layer to get the definition for

**Value**

The definition of the given layer, with members:

- *id* The layer's unique ID
- *parentId* The layer's parent layer ID
- *description* The description of the layer
- *alignment* The layer's alignment - 0 for none, 1 for point alignment, 2 for interval alignment
- *peers* Whether children have peers or not
- *peersOverlap* Whether child peers can overlap or not
- *parentIncludes* Whether the parent t-includes the child
- *saturated* Whether children must temporally fill the entire parent duration (true) or not (false)
- *parentIncludes* Whether the parent t-includes the child
- *type* The type for labels on this layer
- *validLabels* List of valid label values for this layer

**See Also**

[getLayerIds](#) [getLayers](#)

**Examples**

```
## Not run:  
## Get the definition of the orthography layer  
orthography.layer <- getLayer("https://labbcat.canterbury.ac.nz/demo/", "orthography")  
  
## End(Not run)
```

---

getLayerIds

*Gets a list of layer IDs.*

---

**Description**

Layer IDs are annotation 'types'.

**Usage**

```
getLayerIds(labbcat.url)
```

**Arguments**

labbcat.url      URL to the LaBB-CAT instance

**Value**

A list of layer IDs

**Examples**

```
## Not run:  
## Get names of all layers  
layer.ids <- getLayerIds("https://labbcat.canterbury.ac.nz/demo/")  
  
## End(Not run)
```

---

getLayers	<i>Gets a list of layer definitions.</i>
-----------	--

---

**Description**

Gets a list of layer definitions.

**Usage**

```
getLayers(labbcat.url)
```

**Arguments**

labbcat.url      URL to the LaBB-CAT instance

**Value**

A list of layer definitions, with members:

- *id* The layer's unique ID
- *parentId* The layer's parent layer ID
- *description* The description of the layer
- *alignment* The layer's alignment - 0 for none, 1 for point alignment, 2 for interval alignment
- *peers* Whether children have peers or not
- *peersOverlap* Whether child peers can overlap or not
- *parentIncludes* Whether the parent t-includes the child
- *saturated* Whether children must temporally fill the entire parent duration (true) or not (false)
- *parentIncludes* Whether the parent t-includes the child
- *type* The type for labels on this layer
- *validLabels* List of valid label values for this layer

**See Also**

[getLayerIds](#)

**Examples**

```
## Not run:  
## Get definitions of all layers  
layers <- getLayers("https://labbcat.canterbury.ac.nz/demo/")  
  
## End(Not run)
```

---

getMatchAlignments      *Gets temporal alignments of matches on a given layer.*

---

### Description

Gets labels and start/end offsets of annotations on a given layer, identified by given match IDs.

### Usage

```
getMatchAlignments(
  labbcats.url,
  match.ids,
  layer.ids,
  target.offset = 0,
  annotations.per.layer = 1,
  anchor.confidence.min = 50,
  include.match.ids = FALSE,
  page.length = 1000,
  no.progress = FALSE
)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
match.ids	A vector of annotation IDs, e.g. the MatchId column, or the URL column, of a results set.
layer.ids	A vector of layer IDs.
target.offset	The distance from the original target of the match, e.g. <ul style="list-style-type: none"> <li>• 0 – find annotations of the match target itself,</li> <li>• 1 – find annotations of the token immediately <i>after</i> match target</li> <li>• -1 – find annotations of the token immediately <i>before</i> match target</li> </ul>
annotations.per.layer	The number of annotations on the given layer to retrieve. In most cases, there's only one annotation available. However, tokens may, for example, be annotated with 'all possible phonemic transcriptions', in which case using a value of greater than 1 for this parameter provides other phonemic transcriptions, for tokens that have more than one.
anchor.confidence.min	The minimum confidence for alignments, e.g. <ul style="list-style-type: none"> <li>• 0 – return all alignments, regardless of confidence;</li> <li>• 50 – return only alignments that have been at least automatically aligned;</li> <li>• 100 – return only manually-set alignments.</li> </ul>
include.match.ids	Whether or not the data frame returned includes the original MatchId column or not.

page.length	In order to prevent timeouts when there are a large number of matches or the network connection is slow, rather than retrieving matches in one big request, they are retrieved using many smaller requests. This parameter controls the number of results retrieved per request.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

### Details

You can specify a threshold for confidence in the alignment, which is a value from 0 (not aligned) to 100 (manually aligned). The default is 50 (automatically aligned), so only alignments that have been at least automatically aligned are specified. For cases where there's a token but its alignment confidence falls below the threshold, a label is returned, but the start/end times are NA.

### Value

A data frame with label, start time, and end time, for each layer.

### See Also

[getMatches](#) [getMatchLabels](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Perform a search
results <- getMatches(labbcats.url, list(segment="I"))

## Get the segment following the token, with alignment if it's been manually aligned
following.segment <- getMatchAlignments(labbcats.url, results$MatchId, "segment",
  target.offset=1, anchor.confidence.min=100)

## End(Not run)
```

---

getMatches

*Search for tokens.*

---

### Description

Searches through transcripts for tokens matching the given pattern.

**Usage**

```
getMatches(
  labbcat.url,
  pattern,
  participant.ids = NULL,
  transcript.types = NULL,
  main.participant = TRUE,
  aligned = FALSE,
  matches.per.transcript = NULL,
  words.context = 0,
  max.matches = NULL,
  overlap.threshold = NULL,
  anchor.confidence.min = 50,
  page.length = 1000,
  no.progress = FALSE
)
```

**Arguments**

`labbcat.url` URL to the LaBB-CAT instance  
`pattern` An object representing the pattern to search for.

This can be:

- A string, representing a search of the orthography layer - spaces are taken to be word boundaries
- A single named list, representing a one-column search - names are taken to be layer IDs
- A list of named lists, representing a multi-column search - the outer list represents the columns of the search matrix where each column 'immediately follows' the previous, and the names of the inner lists are taken to be layer IDs
- A named list fully replicating the structure of the search matrix in the LaBB-CAT browser interface, with one element called "columns", containing a named list for each column.

Each element in the "columns" named list contains an element named "layers", whose value is a named list for patterns to match on each layer, and optionally an element named "adj", whose value is a number representing the maximum distance, in tokens, between this column and the next column - if "adj" is not specified, the value defaults to 1, so tokens are contiguous. Each element in the "layers" named list is named after the layer it matches, and the value is a named list with the following possible elements:

- *pattern* A regular expression to match against the label
- *min* An inclusive minimum numeric value for the label
- *max* An exclusive maximum numeric value for the label
- *not* TRUE to negate the match
- *anchorStart* TRUE to anchor to the start of the annotation on this layer (i.e. the matching word token will be the first at/after the start of the matching annotation on this layer)

- *anchorEnd* TRUE to anchor to the end of the annotation on this layer (i.e. the matching word token will be the last before/at the end of the matching annotation on this layer)
- *target* TRUE to make this layer the target of the search; the results will contain one row for each match on the target layer

Examples of valid pattern objects include:

```
## the word 'the' followed immediately by a word starting with an orthographic vowel
pattern <- "the [aeiou]"
```

```
## a word spelt with "k" but pronounced "n" word initially
pattern <- list(orthography = "k.*", phonemes = "n.*")
```

```
## the word 'the' followed immediately by a word starting with an phonemic vowel
pattern <- list(
  list(orthography = "the"),
  list(phonemes = "[cCEFHIPqQuUV0123456789~#\${@}.*")
```

```
## the word 'the' followed immediately or with one intervening word by
## a hapax legomenon (word with a frequency of 1) that doesn't start with a vowel
pattern <- list(columns = list(
  list(layers = list(
    orthography = list(pattern = "the")),
    adj = 2),
  list(layers = list(
    phonemes = list(not = TRUE, pattern = "[cCEFHIPqQuUV0123456789~#\${@}.*"),
    frequency = list(max = "2")))))
```

*participant.ids*

An optional list of participant IDs to search the utterances of. If not supplied, all utterances in the corpus will be searched.

*transcript.types*

An optional list of transcript types to limit the results to. If null, all transcript types will be searched.

*main.participant*

TRUE to search only main-participant utterances, FALSE to search all utterances.

*aligned*

true to include only words that are aligned (i.e. have anchor confidence  $\geq$  50, false to search include un-aligned words as well.

*matches.per.transcript*

Optional maximum number of matches per transcript to return. NULL means all matches.

*words.context*

Number of words context to include in the 'Before.Match' and 'After.Match' columns in the results.

*max.matches*

The maximum number of matches to return, or null to return all.

*overlap.threshold*

The percentage overlap with other utterances before simultaneous speech is excluded, or null to include overlapping speech.

anchor.confidence.min	The minimum confidence for alignments, e.g. <ul style="list-style-type: none"> <li>• 0 – return all alignments, regardless of confidence;</li> <li>• 50 – return only alignments that have been at least automatically aligned;</li> <li>• 100 – return only manually-set alignments.</li> </ul>
page.length	In order to prevent timeouts when there are a large number of matches or the network connection is slow, rather than retrieving matches in one big request, they are retrieved using many smaller requests. This parameter controls the number of results retrieved per request.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

### Value

A data frame identifying matches, containing the following columns:

- *SearchName* A name based on the pattern – the same for all rows
- *MatchId* A unique ID for the matching target token
- *Transcript* Name of the transcript in which the match was found
- *Participant* Name of the speaker
- *Corpus* The corpus of the transcript
- *Line* The start offset of the utterance/line
- *LineEnd* The end offset of the utterance/line
- *Before.Match* Transcript text immediately before the match
- *Text* Transcript text of the match
- *After.Match* Transcript text immediately after the match
- *Number* Row number
- *URL* URL of the first matching word token
- *Target.word* Text of the target word token
- *Target.word.start* Start offset of the target word token
- *Target.word.end* End offset of the target word token
- *Target.segment* Label of the target segment (only present if the segment layer is included in the pattern)
- *Target.segment.start* Start offset of the target segment (only present if the segment layer is included in the pattern)
- *Target.segment.end* End offset of the target segment (only present if the segment layer is included in the pattern)

**See Also**

[getFragments](#)  
[getSoundFragments](#)  
[getMatchLabels](#)  
[getMatchAlignments](#)  
[processWithPraat](#)  
[getParticipantIds](#)

**Examples**

```

## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## the word 'the' followed immediately by a word starting with an orthographic vowel
theThenOrthVowel <- getMatches(labbcats.url, "the [aeiou]")

## a word spelt with "k" but pronounced "n" word initially
knWords <- getMatches(labbcats.url, list(orthography = "k.*", phonemes = "n.*"))

## the word 'the' followed immediately by a word starting with a phonemic vowel
theThenPhonVowel <- getMatches(
  labbcats.url, list(
    list(orthography = "the"),
    list(phonemes = "[cCEFHIPqQuUV0123456789~#\$\@].*")))

## the word 'the' followed immediately or with one intervening word by
## a hapax legomenon (word with a frequency of 1) that doesn't start with a vowel
results <- getMatches(
  labbcats.url, list(columns = list(
    list(layers = list(
      orthography = list(pattern = "the"),
      adj = 2),
    list(layers = list(
      phonemes = list(not=TRUE, pattern = "[cCEFHIPqQuUV0123456789~#\$\@].*"),
      frequency = list(max = "2"))))),
  overlap.threshold = 5)

## results$Text is the text that matched
## results$MatchId can be used to access results using other functions

## End(Not run)

```

---

getMatchingAnnotations

*Gets a list of annotations that match a particular pattern.*

---

**Description**

Returns the annotations in the corpus that match the given expression.

**Usage**

```
getMatchingAnnotations(
  labbcats.url,
  expression,
  page.length = NULL,
  page.number = NULL
)
```

**Arguments**

<code>labbcats.url</code>	URL to the LaBB-CAT instance
<code>expression</code>	An expression that determines which annotations match. This must match by either <code>id</code> or <code>layer.id</code> . The expression language is currently not well defined, but is based on JavaScript syntax. e.g. <ul style="list-style-type: none"> <li><code>id == 'ew_0_456'</code></li> <li><code>['ew_2_456', 'ew_2_789', 'ew_2_101112'].includes(id)</code></li> <li><code>layerId == 'orthography' &amp;&amp; !/th[aeiou].+/.test(label)</code></li> <li><code>graph.id == 'AdaAicheson-01.trs' &amp;&amp; layer.id == 'orthography' &amp;&amp; start.offset &gt; 10.5</code></li> <li><code>layer.id == 'utterance' &amp;&amp; all('word').includes('ew_0_456')</code></li> <li><code>layerId = 'utterance' &amp;&amp; labels('orthography').includes('foo')</code></li> <li><code>layerId = 'utterance' &amp;&amp; labels('participant').includes('Ada')</code></li> </ul>
<code>page.length</code>	The maximum number of IDs to return, or null to return all
<code>page.number</code>	The zero-based page number to return, or null to return the first page

**Details**

The results can be exhaustive, by omitting `page.length` and `page.number`, or they can be a subset (a 'page') of results, by given `page.length` and `page.number` values.

**Value**

A list of annotations.

**See Also**

[countMatchingAnnotations](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcat.url <- "https://labbcat.canterbury.ac.nz/demo/"

## get all topic annotations whose label includes the word 'quake'
quake.topics <- getMatchingAnnotations(
  labbcat.url, "layer.id == 'topic' && /.quake.*/.test(label)")

## End(Not run)
```

---

getMatchingGraphIds     *Deprecated synonym for getMatchingTranscriptIds.*

---

**Description**

Gets a list of IDs of graphs (i.e. transcript names) that match a particular pattern.

**Usage**

```
getMatchingGraphIds(
  labbcat.url,
  expression,
  page.length = NULL,
  page.number = NULL,
  order = NULL
)
```

**Arguments**

labbcat.url	URL to the LaBB-CAT instance
expression	An expression that determines which graphs match
page.length	The maximum number of IDs to return, or null to return all
page.number	The zero-based page number to return, or null to return the first page
order	An expression that determines the order the graphs are listed in - if specified, this must include the keyword 'ASC' for ascending or 'DESC' for descending order.

**Details**

The results can be exhaustive, by omitting pageLength and page.number, or they can be a subset (a 'page') of results, by given pageLength and page.number values.

The order of the list can be specified. If omitted, the graphs are listed in ID order.

The expression language is currently not well defined, but is based on JavaScript syntax.

- The *labels* function can be used to represent a list of all the annotation labels on a given layer. For example, each transcript can have multiple participants, so the participant labels (names) are represented by: *labels('participant')*
- Use the *includes* function on a list to test whether the list contains a given element. e.g. to match transcripts that include the participant 'Joe' use: *labels('participant').includes('Joe')*
- Use the *first* function to identify the first (or the only) annotation on a given layer. e.g. the annotation representing the transcript's corpus is: *first('corpus')*
- Single annotations have various attributes, including 'id', 'label', 'ordinal', etc. e.g. the name of the transcript's corpus is: *first('corpus').label*
- Regular expressions can be matched by using expressions like *'/regex/.test(str)'*, e.g. to test if the ID starts with 'BR' use: */^BR.+/.test(id)* or to test if the transcript's corpus includes a B use: */.\*B.\*/.test(first('corpus').label)*

Expressions such as those in the examples can be used.

## Value

A list of graph IDs (i.e. transcript names)

## Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get all transcripts whose names start with "BR"
transcripts <- getMatchingGraphIds(labbcats.url, "^BR.+/.test(id)")

## Get the first twenty transcripts in the "QB" corpus
transcripts <- getMatchingGraphIds(
  labbcats.url, "first('corpus').label = 'QB'", 20, 0)

## Get the second transcript that has "QB247_Jacqui" as a speaker
transcripts <- getMatchingGraphIds(
  labbcats.url, "labels('participant').includes('QB247_Jacqui'", 1, 1)

## Get all transcripts in the QB corpus whose names start with "BR"
## in word-count order
transcripts <- getMatchingGraphIds(
  labbcats.url, "first('corpus').label = 'QB' && /^BR.+/.test(id)",
  order="first('transcript_word_count').label ASC")

## End(Not run)
```

---

```
getMatchingParticipantIds
```

*Gets a list of IDs of participants that match a particular pattern.*

---

### Description

Gets a list of IDs of participants that match a particular pattern.

### Usage

```
getMatchingParticipantIds(
  labbcat.url,
  expression,
  page.length = NULL,
  page.number = NULL
)
```

### Arguments

<code>labbcat.url</code>	URL to the LaBB-CAT instance
<code>expression</code>	An expression that determines which participants match
<code>page.length</code>	The maximum number of IDs to return, or null to return all
<code>page.number</code>	The zero-based page number to return, or null to return the first page

### Details

The results can be exhaustive, by omitting `page.length` and `page.number`, or they can be a subset (a 'page') of results, by given `page.length` and `page.number` values.

The expression language is currently not well defined, but is based on JavaScript syntax.

- The *labels* function can be used to represent a list of all the annotation labels on a given layer. For example, each participant can have multiple corpora, so the corpus labels (names) are represented by: *labels('corpus')*
- Use the *includes* function on a list to test whether the list contains a given element. e.g. to match participants that include the corpus 'QB' use: *labels('corpus').includes('QB')*
- Use the *first* function to identify the first (or the only) annotation on a given layer. e.g. the annotation representing the participant's gender is: *first('participant\_gender')*
- Single annotations have various attributes, including 'id', 'label', 'ordinal', etc. e.g. the label of the participant's gender is: *first('participant\_gender').label*
- Regular expressions can be matched by using expressions like */regex/.test(str)*, e.g. to test if the ID starts with 'BR' use: *^BR.+/.test(id)* or to test if the participant's gender includes 'binary' use: */.\*binary.\*/.test(first('participant\_gender').label)*

Expressions such as those in the examples can be used.

**Value**

A list of participant IDs

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get all participants whose IDs start with "BR"
participants <- getMatchingParticipantIds(labbcats.url, "^BR.+/.test(id)")

## Get the first twenty transcripts in the "QB" corpus
participants <- getMatchingParticipantIds(
  labbcats.url, "labels('corpus').includes('QB')", 20, 0)

## Get all participants in the "QB" corpus that have "Jacqui" as part of the ID
participants <- getMatchingTranscriptParticipantIds(
  labbcats.url, "labels('corpus').includes('QB') && /^BR.+/.test(id)")

## End(Not run)
```

---

getMatchingTranscriptIds

*Gets a list of IDs of transcripts that match a particular pattern.*

---

**Description**

Gets a list of IDs of transcripts (i.e. transcript names) that match a particular pattern.

**Usage**

```
getMatchingTranscriptIds(
  labbcats.url,
  expression,
  page.length = NULL,
  page.number = NULL,
  order = NULL
)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
expression	An expression that determines which transcripts match
page.length	The maximum number of IDs to return, or null to return all
page.number	The zero-based page number to return, or null to return the first page

order            An expression that determines the order the transcripts are listed in - if specified, this must include the keyword 'ASC' for ascending or 'DESC' for descending order.

## Details

The results can be exhaustive, by omitting `page.length` and `page.number`, or they can be a subset (a 'page') of results, by given `page.length` and `page.number` values.

The order of the list can be specified. If omitted, the transcripts are listed in ID order.

The expression language is currently not well defined, but is based on JavaScript syntax.

- The *labels* function can be used to represent a list of all the annotation labels on a given layer. For example, each transcript can have multiple participants, so the participant labels (names) are represented by: `labels('participant')`
- Use the *includes* function on a list to test whether the list contains a given element. e.g. to match transcripts that include the participant 'Joe' use: `labels('participant').includes('Joe')`
- Use the *first* function to identify the first (or the only) annotation on a given layer. e.g. the annotation representing the transcript's corpus is: `first('corpus')`
- Single annotations have various attributes, including 'id', 'label', 'ordinal', etc. e.g. the name of the transcript's corpus is: `first('corpus').label`
- Regular expressions can be matched by using expressions like `/regex/.test(str)`, e.g. to test if the ID starts with 'BR' use: `/^BR.+/.test(id)` or to test if the transcript's corpus includes a B use: `/.*B.*/.test(first('corpus').label)`

Expressions such as those in the examples can be used.

## Value

A list of transcript IDs (i.e. transcript names)

## Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get all transcripts whose names start with "BR"
transcripts <- getMatchingTranscriptIds(labbcats.url, "/^BR.+/.test(id)")

## Get the first twenty transcripts in the "QB" corpus
transcripts <- getMatchingTranscriptIds(
  labbcats.url, "first('corpus').label = 'QB'", 20, 0)

## Get the second transcript that has "QB247_Jacqui" as a speaker
transcripts <- getMatchingTranscriptIds(
  labbcats.url, "labels('participant').includes('QB247_Jacqui')", 1, 1)

## Get all transcripts in the QB corpus whose names start with "BR"
## in word-count order
```

```

transcripts <- getMatchingTranscriptIds(
  labbcats.url, "first('corpus').label = 'QB' && /^BR.+/.test(id)",
  order="first('transcript_word_count').label ASC")

## End(Not run)

```

---

getMatchLabels	<i>Gets labels of annotations on a given layer, identified by given match IDs.</i>
----------------	--

---

### Description

Gets labels of annotations on a given layer, identified by given match IDs.

### Usage

```

getMatchLabels(
  labbcats.url,
  match.ids,
  layer.ids,
  target.offset = 0,
  annotations.per.layer = 1,
  include.match.ids = FALSE,
  page.length = 1000,
  no.progress = FALSE
)

```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
match.ids	A vector of annotation IDs, e.g. the MatchId column, or the URL column, of a results set.
layer.ids	A vector of layer IDs.
target.offset	The distance from the original target of the match, e.g. <ul style="list-style-type: none"> <li>• 0 – find annotations of the match target itself,</li> <li>• 1 – find annotations of the token immediately <i>after</i> match target</li> <li>• -1 – find annotations of the token immediately <i>before</i> match target</li> </ul>
annotations.per.layer	The number of annotations on the given layer to retrieve. In most cases, there's only one annotation available. However, tokens may, for example, be annotated with 'all possible phonemic transcriptions', in which case using a value of greater than 1 for this parameter provides other phonemic transcriptions, for tokens that have more than one.
include.match.ids	Whether or not the data frame returned includes the original MatchId column or not.

page.length	In order to prevent timeouts when there are a large number of matches or the network connection is slow, rather than retrieving matches in one big request, they are retrieved using many smaller requests. This parameter controls the number of results retrieved per request.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

**Value**

A data frame of labels.

**See Also**

[getMatches](#) [getMatchAlignments](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Perform a search
results <- getMatches(labbcats.url, list(orthography="quake"))

## Get the topic annotations for the matches
topics <- getMatchLabels(labbcats.url, results$MatchId, "topic")

## End(Not run)
```

---

getMedia

*Downloads a given media track for a given transcript.*

---

**Description**

Downloads a given media track for a given transcript.

**Usage**

```
getMedia(
  labbcats.url,
  id,
  track.suffix = "",
  mime.type = "audio/wav",
  path = ""
)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance.
id	A transcript ID (i.e. transcript name).
track.suffix	The track suffix of the media.
mime.type	The MIME type of the media, e.g. "audio/wav" or "application/f0".
path	Optional path to directory where the file should be saved.

**Value**

The name of the file, which is saved in the current directory, or the given path if specified

**See Also**

[getTranscriptIds](#)

[getMediaUrl](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Download the WAV file for BR2044_OllyOhlson.eaf
wav <- getMedia(labbcats.url, "BR2044_OllyOhlson.eaf")

## Download the 'QuakeFace' video file for BR2044_OllyOhlson.eaf
quakeFaceMp4 <- getMedia(labbcats.url, "BR2044_OllyOhlson.eaf", "_face", "video/mp4")

## End(Not run)
```

---

getMediaTracks

*List the predefined media tracks available for transcripts.*

---

**Description**

List the predefined media tracks available for transcripts.

**Usage**

```
getMediaTracks(labbcats.url)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
--------------	------------------------------

**Value**

A list of media track definitions.

**Examples**

```
## Not run:  
## Get the media tracks configured in LaBB-CAT  
tracks <- getMediaTracks("https://labbcat.canterbury.ac.nz/demo/")  
  
## End(Not run)
```

---

getMediaUrl	<i>Gets the URL of the given media track for a given transcript.</i>
-------------	--

---

**Description**

Gets the URL of the given media track for a given transcript.

**Usage**

```
getMediaUrl(labbcat.url, id, track.suffix = "", mime.type = "audio/wav")
```

**Arguments**

- labbcat.url      URL to the LaBB-CAT instance.
- id                A transcript ID (i.e. transcript name).
- track.suffix    The track suffix of the media.
- mime.type        The MIME type of the media, e.g. "audio/wav" or "application/f0".

**Value**

A URL to the given media for the given transcript.

**See Also**

- [getTranscriptIds](#)
- [getMedia](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get URL for the WAV file for BR2044_01ly0h1son.eaf
wavUrl <- getMediaUrl(labbcats.url, "BR2044_01ly0h1son.eaf")

## Get URL for the 'QuakeFace' video file for BR2044_01ly0h1son.eaf
quakeFaceMp4Url <- getMediaUrl(labbcats.url, "BR2044_01ly0h1son.eaf", "_face", "video/mp4")

## End(Not run)
```

---

```
getParticipantAttributes
```

*Gets participant attribute values for given participant IDs.*

---

**Description**

Gets participant attribute values for given participant IDs.

**Usage**

```
getParticipantAttributes(labbcats.url, participant.ids, layer.ids)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
participant.ids	A vector of participant IDs
layer.ids	A vector of layer IDs corresponding to participant attributes. In general, these are layers whose ID is prefixed 'participant_', however formally it's any layer where layer\$parentId == 'participant' && layer\$alignment == 0.

**Value**

A data frame of attribute value labels.

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get gender and age for all participants
attributes <- getParticipantAttributes(labbcats.url,
  getParticipantIds(labbcats.url),
```

```
c('participant_gender', 'participant_age'))
```

```
## End(Not run)
```

---

getParticipantIds      *Gets a list of participant IDs.*

---

### Description

Returns a list of participant IDs.

### Usage

```
getParticipantIds(labbcat.url)
```

### Arguments

labbcat.url      URL to the LaBB-CAT instance

### Value

A list of participant IDs

### Examples

```
## Not run:  
## List all speakers  
speakers <- getParticipantIds("https://labbcat.canterbury.ac.nz/demo/")  
  
## End(Not run)
```

---

getSerializerDescriptors

*Lists the descriptors of all registered serializers.*

---

### Description

Returns a list of serializers, which are modules that export annotation structures as a specific file format, e.g. Praat TextGrid, plain text, etc., so the *contentType* of descriptors reflects what *mimeTypes* can be specified for [getFragments](#).

### Usage

```
getSerializerDescriptors(labbcat.url)
```

### Arguments

labbcats.url      URL to the LaBB-CAT instance

### Value

A list of serializers, each including the following information:

- *name* The name of the format.
- *version* The installed version of the serializer module.
- *fileSuffixes* The normal file name suffixes (extensions) of the files.,
- *mimeType* The MIME type of the format, i.e. the value to use as the *mimeType* parameter of [getFragments](#),

### See Also

[getFragments](#)

### Examples

```
## Not run:
## List file export formats supported
formats <- getSerializerDescriptors("https://labbcats.canterbury.ac.nz/demo/")

## can we export as plain text?
plainTextSupported <- "text/plain" %in% formats$mimeType

## End(Not run)
```

---

getSoundFragments      *Gets sound fragments from 'LaBB-CAT'.*

---

### Description

Gets sound fragments from 'LaBB-CAT'.

### Usage

```
getSoundFragments(
  labbcats.url,
  ids,
  start.offsets,
  end.offsets,
  sample.rate = NULL,
  path = "",
  no.progress = FALSE
)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
ids	The transcript ID (transcript name) of the sound recording, or a vector of transcript IDs.
start.offsets	The start time in seconds, or a vector of start times.
end.offsets	The end time in seconds, or a vector of end times.
sample.rate	Optional sample rate in Hz - if a positive integer, then the result is a mono file with the given sample rate.
path	Optional path to directory where the files should be saved.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

**Value**

The name of the file, which is saved in the current directory, or a list of names of files, if multiple id's/start's/end's were specified

If a list of files is returned, they are in the order that they were returned by the server, which *should* be the order that they were specified in the id/start/end lists.

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get the 5 seconds starting from 10s after the beginning of a recording
wav.file <- getSoundFragments(labbcats.url, "AP2505_Nelson.eaf", 10.0, 15.0, path="samples")

## Get the 5 seconds starting from 10s as a mono 22kHz file
wav.file <- getSoundFragments(labbcats.url, "AP2505_Nelson.eaf", 10.0, 15.0, 22050)

## Load some search results previously exported from LaBB-CAT
results <- read.csv("results.csv", header=T)

## Get a list of fragments
wav.files <- getSoundFragments(labbcats.url, results$Transcript, results$Line, results$LineEnd)

## Get a list of fragments
wav.file <- getSoundFragments(
  labbcats.url, results$Transcript, results$Line, results$LineEnd)

## End(Not run)
```

getSystemAttribute      *Gets the value of the given system attribute.*

---

**Description**

Gets the value of the given system attribute.

**Usage**

```
getSystemAttribute(labbcat.url, attribute)
```

**Arguments**

labbcat.url      URL to the LaBB-CAT instance  
attribute        Name of the attribute.

**Value**

The value of the given attribute.

[getLayers](#)

**Examples**

```
## Not run:  
## Get the name of the LaBB-CAT instance  
title <- getSystemAttribute("https://labbcat.canterbury.ac.nz/demo/", "title")  
  
## End(Not run)
```

---

getTranscriptAttributes  
*Gets transcript attribute values for given transcript IDs.*

---

**Description**

Gets transcript attribute values for given transcript IDs.

**Usage**

```
getTranscriptAttributes(labbcat.url, transcript.ids, layer.ids)
```

**Arguments**

labbcats.url      URL to the LaBB-CAT instance

transcript.ids    A vector of transcript IDs

layer.ids         A vector of layer IDs corresponding to transcript attributes. In general, these are layers whose ID is prefixed 'transcript\_', however formally it's any layer where layer\$parentId == 'transcript' && layer\$alignment == 0, which includes 'corpus' as well as transcript attribute layers.

**Value**

A data frame of attribute value labels.

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get language, duration, and corpus for transcripts starting with 'BR'
attributes <- getTranscriptAttributes(labbcats.url,
  getMatchingTranscriptIds(labbcats.url, "'BR.+/.test(id)"),
  c('transcript_language', 'transcript_duration', 'corpus'))

## End(Not run)
```

---

getTranscriptIds      *Gets a list of transcript IDs.*

---

**Description**

Returns a list of transcript IDs (i.e. transcript names).

**Usage**

```
getTranscriptIds(labbcats.url)
```

**Arguments**

labbcats.url      URL to the LaBB-CAT instance

**Value**

A list of transcript IDs

**Examples**

```
## Not run:  
## List all transcripts  
transcripts <- getTranscriptIds("https://labbcat.canterbury.ac.nz/demo/")  
  
## End(Not run)
```

---

getTranscriptIdsInCorpus

*Gets a list of transcript in a corpus.*

---

**Description**

Returns a list of transcript IDs in the given corpus.

**Usage**

```
getTranscriptIdsInCorpus(labbcat.url, id)
```

**Arguments**

labbcat.url	URL to the LaBB-CAT instance
id	The ID (name) of the corpus

**Value**

A list of transcript IDs

**Examples**

```
## Not run:  
## List transcripts in the QB corpus  
transcripts <- getTranscriptIdsInCorpus("https://labbcat.canterbury.ac.nz/demo/", "QB")  
  
## End(Not run)
```

---

`getTranscriptIdsWithParticipant`*Gets a list of IDs of transcripts that include the given participant.*

---

**Description**

Returns a list of IDs of transcripts (i.e. transcript names) that include the given participant.

**Usage**

```
getTranscriptIdsWithParticipant(labbcat.url, id)
```

**Arguments**

<code>labbcat.url</code>	URL to the LaBB-CAT instance
<code>id</code>	A participant ID

**Value**

A list of transcript IDs

**See Also**

[getParticipantIds](#)

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcat.url <- "https://labbcat.canterbury.ac.nz/demo/"

## List transcripts in which UC427_ViktoriaPapp_A_ENG speaks
transcripts <- getTranscriptIdsWithParticipant(labbcat.url, "UC427_ViktoriaPapp_A_ENG")

## End(Not run)
```

---

`getUserInfo`*Gets information about the current user.*

---

**Description**

Returns information about the current user, including the roles or groups they are in.

**Usage**

```
getUserInfo(labbcat.url)
```

**Arguments**

labbcats.url      URL to the LaBB-CAT instance

**Value**

A named list containing information about current the LaBB-CAT user.

**See Also**

[labbcatsCredentials](#)

**Examples**

```
## Not run:
## List file export formats supported
me <- getUserInfo("https://labbcats.canterbury.ac.nz/demo/")

## am I an administrator?
admin <- "admin" %in% me$roles

## End(Not run)
```

---

labbcatsCredentials	<i>Sets the username and password that the package should use for connecting to a given LaBB-CAT server in future function calls.</i>
---------------------	---

---

**Description**

This step is optional, as all functions will prompt the user for the username and password if required. If the script is running in RStudio, then the RStudio password input dialog is used, hiding the credentials from view. Otherwise, the console is used, and credentials are visible.

**Usage**

```
labbcatsCredentials(labbcats.url, username, password)
```

**Arguments**

labbcats.url      URL to the LaBB-CAT instance  
 username          The LaBB-CAT username, if it is password-protected  
 password          The LaBB-CAT password, if it is password-protected

**Details**

The recommended approach is to *not* use labbcatsCredentials, to avoid saving user credentials in script files that may eventually become visible to other. Use labbcatsCredentials *only* in cases where the script execution is unsupervised.

**Value**

NULL if the username/password are correct, and a string describing the problem if a problem occurred, e.g. "Credentials rejected" if the username/password are incorrect, or a string starting "Version mismatch" if the server's version of LaBB-CAT is lower than the minimum required.

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcat.url <- "https://labbcat.canterbury.ac.nz/demo/"

## specify the username/password in the script
## (only use labbcatCredentials for scripts that must execute unsupervised!)
labbcatCredentials(labbcat.url, "demo", "demo")

## End(Not run)
```

---

labbcatTimeout	<i>Sets the timeout for request to the LaBB-CAT server in future function calls. The default timeout is 10 seconds.</i>
----------------	---

---

**Description**

Sets the timeout for request to the LaBB-CAT server in future function calls. The default timeout is 10 seconds.

**Usage**

```
labbcatTimeout(seconds = NULL)
```

**Arguments**

seconds            The number of seconds before requests return with a timeout error.

**Value**

The request timeout in seconds

**Examples**

```
## Not run:
## the request timeout
labbcatTimeout(30)

## End(Not run)
```

---

loadLexicon

*Upload a flat lexicon file for lexical tagging.*


---

### Description

By default LaBB-CAT includes a layer manager called the Flat Lexicon Tagger, which can be configured to annotate words with data from a dictionary loaded from a plain text file (e.g. a CSV file). The file must have a 'flat' structure in the sense that it's a simple list of dictionary entries with a fixed number of columns/fields, rather than having a complex structure.

### Usage

```
loadLexicon(
  labbcats.url,
  file,
  lexicon,
  field.delimiter,
  field.names,
  quote = "",
  comment = "",
  skip.first.line = FALSE,
  no.progress = FALSE
)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance.
file	The full path name of the lexicon file.
lexicon	The name for the resulting lexicon. If the named lexicon already exists, it will be completely replaced with the contents of the file (i.e. all existing entries will be deleted before adding new entries from the file). e.g. 'cmudict'
field.delimiter	The character used to delimit fields in the file. If this is " - ", rows are split on only the <i>first</i> space, in line with common dictionary formats. e.g. ',' for Comma Separated Values (CSV) files.
field.names	A list of field names, delimited by field.delimiter, e.g. 'Word,Pronunciation'.
quote	The character used to quote field values (if any), e.g. ''.
comment	The character used to indicate a line is a comment (not an entry) (if any) e.g. '#'.
skip.first.line	Whether to ignore the first line of the file (because it contains field names).
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

## Details

This function uploads such a lexicon file, for use in tagging tokens.

You must have editing privileges in LaBB-CAT in order to be able to use this function.

## Value

An error message, or NULL if the upload was successful.

## See Also

[getDictionaries](#) [deleteLexicon](#)

## Examples

```
## Not run:  
## Upload the CMU Pronouncing Dictionary  
loadLexicon(labbcats.url, "cmudict", " - ", "", ";", "Word - Pron", FALSE, "cmudict.txt")  
  
## End(Not run)
```

---

newLayer

*Creates a new layer.*

---

## Description

This function creates a new annotation layer. The layer may be configured with a layer manager ID and task parameters, for automatic annotation. If so, this function will create the layer and the automation task, but automatic annotation will not be run by this function. To generate the automatic annotations, use [generateLayer](#).

## Usage

```
newLayer(  
  labbcats.url,  
  layer.id,  
  description,  
  type = "string",  
  alignment = 0,  
  category = "General",  
  parent.id = "word",  
  annotator.id = NULL,  
  annotator.task.parameters = NULL  
)
```

**Arguments**

<code>labbcats.url</code>	URL to the LaBB-CAT instance
<code>layer.id</code>	The ID of the layer to create, which must be unique to the LaBB-CAT instance.
<code>description</code>	A description of the annotations the layer will contain.
<code>type</code>	The type of data the labels will represent. Valid values are "string", "number", "ipa" (for phoneme representations), or "boolean" (labels "0" or "1").
<code>alignment</code>	How annotations on the layer will relate to time alignment; valid values are 0 (no alignment; annotations are just tags on the parent annotation), 1 (instants; annotations mark a single point in time), or 2 (intervals; annotations have a start and end time).
<code>category</code>	The project/category the layer belongs to.
<code>parent.id</code>	The parent layer; valid values are "word" (for word layers), "segment" (for segment layers) "turn" (for phrase layers), or "transcript" (for span layers).
<code>annotator.id</code>	The ID of the layer manager that automatically fills in annotations on the layer, if any
<code>annotator.task.parameters</code>	The configuration the layer manager should use when filling the layer with annotations. This is a string whose format is specific to each layer manager.

**Details**

You must have administration privileges in LaBB-CAT in order to be able to use this function.

**Value**

The resulting layer definition, with members:

- *id* The layer's unique ID
- *parentId* The layer's parent layer ID
- *description* The description of the layer
- *alignment* The layer's alignment - 0 for none, 1 for point alignment, 2 for interval alignment
- *peers* Whether children have peers or not
- *peersOverlap* Whether child peers can overlap or not
- *parentIncludes* Whether the parent t-includes the child
- *saturated* Whether children must temporally fill the entire parent duration (true) or not (false)
- *parentIncludes* Whether the parent t-includes the child
- *type* The type for labels on this layer
- *validLabels* List of valid label values for this layer

**See Also**

[generateLayer](#) [saveLayer](#) [deleteLayer](#)

**Examples**

```

## Not run:
## Upload the CMU Pronouncing Dictionary
loadLexicon(labbcats.url, "cmudict", " - ", "", ";", "Word - Pron", FALSE, "cmudict.txt")

## Create a layer that tags each token with its CMU Pronouncing Dictionary pronunciation
newLayer(labbcats.url, "pronunciation", "CMU Dict pronunciations encoded in ARPAbet",
         annotator.id="FlatFileDictionary",
         annotator.task.parameters=
           "tokenLayerId=orthography&tagLayerId=phonemes&dictionary=cmudict:Word->Pron")

## Generate the pronunciation tags
generateLayer(labbcats.url, "pronunciation")

## End(Not run)

```

---

newTranscript	<i>Upload a new transcript.</i>
---------------	---------------------------------

---

**Description**

This function adds a transcript and optionally a media file to the corpus.

**Usage**

```

newTranscript(
  labbcats.url,
  transcript,
  media = NULL,
  transcript.type = NULL,
  corpus = NULL,
  episode = NULL,
  no.progress = FALSE
)

```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
transcript	The path to the transcript to upload.
media	The path to the media to upload, if any.
transcript.type	The transcript type.
corpus	The corpus to add the transcript to.
episode	The transcript's episode.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

**Details**

For this function to work, the credentials used to connect to the server must have at least 'edit' access.

**Value**

The ID of the new transcript in the corpus

**Examples**

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get attributes for new transcript
corpus <- getCorpusIds(labbcats.url)[1]
transcript.type.layer <- getLayer(labbcats.url, "transcript_type")
transcript.type <- transcript.type.layer$validLabels[[1]]

## upload transcript
newTranscript(
  labbcats.url, "my-transcript.eaf", "my-transcript.wav",
  "", transcript.type, corpus, "episode-1")

## End(Not run)
```

---

 nzilbb.labbcats

*Accessing Data Stored in 'LaBB-CAT' Instances*


---

**Description**

'LaBB-CAT' is a web-based language corpus management system developed by the New Zealand Institute of Language, Brain and Behaviour (NZILBB) - see <<https://labbcats.canterbury.ac.nz>>. This package defines functions for accessing corpus data in a 'LaBB-CAT' instance. You must have at least version 20230202.1600 of 'LaBB-CAT' to use this package. For more information about 'LaBB-CAT', see Robert Fromont and Jennifer Hay (2008) <[doi:10.3366/E1749503208000142](https://doi.org/10.3366/E1749503208000142)> or Robert Fromont (2017) <[doi:10.1016/j.csl.2017.01.004](https://doi.org/10.1016/j.csl.2017.01.004)>.

**Details**

Package:	nzilbb.labbcats
Version:	1.2-0
Date:	2023-01-19
Title:	Accessing Data Stored in 'LaBB-CAT' Instances
Authors@R:	c(person("Robert", "Fromont", role = c("aut", "cre"), email = "robert.fromont@canterbury.ac.nz", comment =
Imports:	jsonlite, httr, stringr, utils, rstudioapi
Description:	'LaBB-CAT' is a web-based language corpus management system developed by the New Zealand Institute of
License:	GPL (>= 3)

Copyright: New Zealand Institute of Language, Brain and Behaviour, University of Canterbury  
 URL: <https://nzilbb.github.io/labbcats-R/>, <https://labbcats.canterbury.ac.nz>  
 RoxygenNote: 7.2.3  
 Suggests: testthat (>= 2.1.0)  
 Author: Robert Fromont [aut, cre] (<<https://orcid.org/0000-0001-5271-5487>>)  
 Maintainer: Robert Fromont <[robert.fromont@canterbury.ac.nz](mailto:robert.fromont@canterbury.ac.nz)>

#### Index of help topics:

addDictionaryEntry Adds an entry to a dictionary.  
 addLayerDictionaryEntry Adds an entry to a layer dictionary.  
 annotatorExt Retrieve annotator's "ext" resource.  
 countAnnotations Gets the number of annotations on the given layer of the given transcript.  
 countMatchingAnnotations Gets the number of annotations matching a particular pattern.  
 deleteLayer Deletes an existing layer.  
 deleteLexicon Delete a previously loaded lexicon.  
 deleteTranscript Delete a transcript from the corpus.  
 formatTranscript Gets transcript(s) in a given format.  
 generateLayer Generates a layer.  
 generateLayerUtterances Generates a layer for a given set of utterances.  
 getAllUtterances Get all utterances of participants.  
 getAnchors Gets the given anchors in the given transcript.  
 getAnnotations Gets the annotations on the given layer of the given transcript.  
 getAnnotatorDescriptor Gets annotator information.  
 getAvailableMedia List the media available for the given transcript.  
 getCorpusIds Gets a list of corpus IDs.  
 getDeserializerDescriptors Lists the descriptors of all registered deserializers.  
 getDictionaries List the dictionaries available.  
 getDictionaryEntries Lookup entries in a dictionary.  
 getFragmentAnnotations Gets annotations in fragments.  
 getFragments Gets transcript fragments in a given format.  
 getGraphIds Deprecated synonym for getTranscriptIds.  
 getGraphIdsInCorpus Deprecated synonym for getTranscriptIdsInCorpus.  
 getGraphIdsWithParticipant

	Deprecated synonym for getTranscriptIdsWithParticipant.
getId	Gets the store's ID.
getLayer	Gets a layer definition.
getLayerIds	Gets a list of layer IDs.
getLayers	Gets a list of layer definitions.
getMatchAlignments	Gets temporal alignments of matches on a given layer.
getMatchLabels	Gets labels of annotations on a given layer, identified by given match IDs.
getMatches	Search for tokens.
getMatchingAnnotations	Gets a list of annotations that match a particular pattern.
getMatchingGraphIds	Deprecated synonym for getMatchingTranscriptIds.
getMatchingParticipantIds	Gets a list of IDs of participants that match a particular pattern.
getMatchingTranscriptIds	Gets a list of IDs of transcripts that match a particular pattern.
getMedia	Downloads a given media track for a given transcript.
getMediaTracks	List the predefined media tracks available for transcripts.
getMediaUrl	Gets the URL of the given media track for a given transcript.
getParticipantAttributes	Gets participant attribute values for given participant IDs.
getParticipantIds	Gets a list of participant IDs.
getSerializerDescriptors	Lists the descriptors of all registered serializers.
getSoundFragments	Gets sound fragments from 'LaBB-CAT'.
getSystemAttribute	Gets the value of the given system attribute.
getTranscriptAttributes	Gets transcript attribute values for given transcript IDs.
getTranscriptIds	Gets a list of transcript IDs.
getTranscriptIdsInCorpus	Gets a list of transcript in a corpus.
getTranscriptIdsWithParticipant	Gets a list of IDs of transcripts that include the given participant.
getUserInfo	Gets information about the current user.
labbcatsCredentials	Sets the username and password that the package

	should use for connecting to a given LaBB-CAT server in future function calls.
labbcatsTimeout	Sets the timeout for request to the LaBB-CAT server in future function calls. The default timeout is 10 seconds.
loadLexicon	Upload a flat lexicon file for lexical tagging.
newLayer	Creates a new layer.
newTranscript	Upload a new transcript.
nzilbb.labbcats	Accessing Data Stored in 'LaBB-CAT' Instances
praatscriptCentreOfGravity	Generates a script for extracting the CoG, for use with processWithPraad.
praatscriptFastTrack	Generates a script for extracting formants using FastTrack, for use with processWithPraad.
praatscriptFormants	Generates a script for extracting formants, for use with processWithPraad.
praatscriptIntensity	Generates a script for extracting maximum intensity, for use with processWithPraad.
praatscriptPitch	Generates a script for extracting pitch, for use with processWithPraad.
processWithPraad	Process a set of intervals with Praat.
removeDictionaryEntry	Removes an entry from a dictionary.
removeLayerDictionaryEntry	Removes an entry from a layer dictionary.
saveLayer	Saves the details of an existing layer.
updateFragment	Update a transcript fragment.
updateTranscript	Update an existing transcript.

'LaBB-CAT' is a web-based language corpus management system and this package provides access to data stored in a 'LaBB-CAT' instance. You must have at least version 20230202.1600 'LaBB-CAT' to use this package.

### Author(s)

NA

### References

*Robert Fromont and Jennifer Hay, "ONZE Miner: the development of a browser-based research tool", 2008 Robert Fromont, "Toward a format-neutral annotation store", 2017*

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Perform a search
results <- getMatches(labbcats.url, list(segment="I"))
```

```
## Get the phonemic transcriptions for the matches
phonemes <- getMatchLabels(labbcat.url, results$MatchId, "phonemes")

## Get sound fragments for the matches
wav.files <- getSoundFragments(labbcat.url, results$Transcript, results$Line, results$LineEnd)

## End(Not run)
```

---

praatScriptCentreOfGravity

*Generates a script for extracting the CoG, for use with [processWithPraat](#).*

---

## Description

This function generates a Praat script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#), in order to extract one or more spectral centre of gravity (CoG) measurements.

## Usage

```
praatScriptCentreOfGravity(powers = c(2), spectrum.fast = TRUE)
```

## Arguments

`powers`            A vector of numbers specifying which powers to query for to extract, e.g. `c(1.0,2.0)`.  
`spectrum.fast`    Whether to use the 'fast' option when creating the spectrum object to query .

## Value

A script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#)

## See Also

[processWithPraat](#)  
[praatScriptFormants](#)  
[praatScriptIntensity](#)  
[praatScriptPitch](#)  
[praatScriptFastTrack](#)

## Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Perform a search
results <- getMatches(labbcats.url, list(segment="I"))

## Get centres of gravity for all matches
cog <- processWithPraat(
  labbcats.url,
  results$MatchId, results$Target.segment.start, results$Target.segment.end,
  praatScriptCentreOfGravity(powers=c(1.0,2.0)))

## End(Not run)
```

---

praatScriptFastTrack *Generates a script for extracting formants using FastTrack, for use with [processWithPraat](#).*

---

## Description

This function generates a Praat script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#), in order to extract selected formants using the FastTrack Praat plugin.

## Usage

```
praatScriptFastTrack(
  formants = c(1, 2),
  sample.points = c(0.5),
  lowest.analysis.frequency = 5000,
  lowest.analysis.frequency.male = 4500,
  highest.analysis.frequency = 7000,
  highest.analysis.frequency.male = 6500,
  gender.attribute = "participant_gender",
  value.for.male = "M",
  time.step = 0.002,
  tracking.method = "burg",
  number.of.formants = 3,
  maximum.f1.frequency = 1200,
  maximum.f1.bandwidth = NULL,
  maximum.f2.bandwidth = NULL,
  maximum.f3.bandwidth = NULL,
  minimum.f4.frequency = 2900,
  enable.rhotic.heuristic = TRUE,
  enable.f3.f4.proximity.heuristic = TRUE,
  number.of.steps = 20,
  number.of.coefficients = 5
)
```

**Arguments**

<code>formants</code>	A vector of integers specifying which formants to extract, e.g c(1,2) for the first and second formant.
<code>sample.points</code>	A vector of numbers ( $0 \leq \text{sample.points} \leq 1$ ) specifying multiple points at which to take the measurement. The default is a single point at 0.5 - this means one measurement will be taken halfway through the target interval. If, for example, you wanted eleven measurements evenly spaced throughout the interval, you would specify <code>sample.points</code> as being c(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0).
<code>lowest.analysis.frequency</code>	Lowest analysis frequency (Hz) by default.
<code>lowest.analysis.frequency.male</code>	Lowest analysis frequency (Hz) for male speakers, or NULL to use the same value as <code>lowest.analysis.frequency</code> .
<code>highest.analysis.frequency</code>	Highest analysis frequency (Hz) by default.
<code>highest.analysis.frequency.male</code>	Highest analysis frequency (Hz) for male speakers, or NULL to use the same value as <code>highest.analysis.frequency</code> .
<code>gender.attribute</code>	Name of the LaBB-CAT participant attribute that contains the participant's gender - normally this is "participant_gender".
<code>value.for.male</code>	The value that the <code>gender.attribute</code> has when the participant is male.
<code>time.step</code>	Time step in seconds.
<code>tracking.method</code>	<code>tracking_method</code> parameter for <code>trackAutoselectProcedure</code> ; "burg" or "robust".
<code>number.of.formants</code>	Number of formants to track - 3 or 4.
<code>maximum.f1.frequency</code>	Specifying a non-NULL value enables the F1 frequency heuristic: Median F1 frequency should not be higher than this value.
<code>maximum.f1.bandwidth</code>	Specifying a non-NULL value (e.g. 500) enables the F1 bandwidth heuristic: Median F1 bandwidth should not be higher than this value.
<code>maximum.f2.bandwidth</code>	Specifying a non-NULL value (e.g. 600) enables the F2 bandwidth heuristic: Median F2 bandwidth should not be higher than this value.
<code>maximum.f3.bandwidth</code>	Specifying a non-NULL value (e.g. 900) enables the F3 bandwidth heuristic: Median F3 bandwidth should not be higher than this value.
<code>minimum.f4.frequency</code>	Specifying a non-NULL value enables the F4 frequency heuristic: Median F4 frequency should not be lower than this value.

<code>enable.rhotic.heuristic</code>	Whether to enable the rhotic heuristic: If $F3 < 2000$ Hz, F1 and F2 should be at least 500 Hz apart.
<code>enable.f3.f4.proximity.heuristic</code>	Whether to enable the F3/F4 proximity heuristic: If $(F4 - F3) < 500$ Hz, F1 and F2 should be at least 1500 Hz apart.
<code>number.of.steps</code>	Number of analyses between low and high analysis limits. More analysis steps may improve results, but will increase analysis time (50 percent more steps = around 50 percent longer to analyze).
<code>number.of.coefficients</code>	Number of coefficients for formant prediction. More coefficients allow for more sudden, and 'wiggly' formant motion.

### Details

The FastTrack Praat plugin, developed by Santiago Barreda, automatically runs multiple formant analyses on each segment, selects the best (the smoothest, with optional heuristics), and makes the winning formant object available for measurement. For more information, see <https://github.com/santiagobarreda/FastTrack>

### Value

A script fragment which can be passed as the `praat.script` parameter of `processWithPraat`

### See Also

[processWithPraat](#)  
[praatScriptCentreOfGravity](#)  
[praatScriptIntensity](#)  
[praatScriptPitch](#)  
[praatScriptFormants](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get all tokens of the KIT vowel
results <- getMatches(labbcats.url, list(segment="I"))

## Get the first 3 formants at three points during the vowel
formants <- processWithPraat(
  labbcats.url,
  results$MatchId, results$Target.segment.start, results$Target.segment.end,
  window.offset=0.025,
  praatScriptFastTrack(formants=c(1,2,3),
  sample.points=c(0.25,0.5,0.75)))
```

```
## End(Not run)
```

---

praatScriptFormants     *Generates a script for extracting formants, for use with [processWithPraat](#).*

---

### Description

This function generates a Praat script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#), in order to extract selected formants.

### Usage

```
praatScriptFormants(
  formants = c(1, 2),
  sample.points = c(0.5),
  time.step = 0,
  max.number.formants = 5,
  max.formant = 5500,
  max.formant.male = 5000,
  gender.attribute = "participant_gender",
  value.for.male = "M",
  window.length = 0.025,
  preemphasis.from = 50
)
```

### Arguments

<code>formants</code>	A vector of integers specifying which formants to extract, e.g <code>c(1,2)</code> for the first and second formant.
<code>sample.points</code>	A vector of numbers ( $0 \leq \text{sample.points} \leq 1$ ) specifying multiple points at which to take the measurement. The default is a single point at 0.5 - this means one measurement will be taken halfway through the target interval. If, for example, you wanted eleven measurements evenly spaced throughout the interval, you would specify <code>sample.points</code> as being <code>c(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)</code> .
<code>time.step</code>	Time step in seconds, or 0.0 for 'auto'.
<code>max.number.formants</code>	Maximum number of formants.
<code>max.formant</code>	Maximum formant value (Hz) for all speakers, or for female speakers, if <code>max.formant.male</code> is also specified.
<code>max.formant.male</code>	Maximum formant value (Hz) for male speakers, or NULL to use the same value as <code>max.formant</code> .

<code>gender.attribute</code>	Name of the LaBB-CAT participant attribute that contains the participant's gender - normally this is "participant_gender".
<code>value.for.male</code>	The value that the <code>gender.attribute</code> has when the participant is male.
<code>window.length</code>	Window length in seconds.
<code>preemphasis.from</code>	Pre-emphasis from (Hz)

### Details

The [praatScriptFastTrack](#) function provides an alternative to this function which uses the FastTrack Praat plugin for formant analysis.

### Value

A script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#)

### See Also

[processWithPraat](#)  
[praatScriptCentreOfGravity](#)  
[praatScriptIntensity](#)  
[praatScriptPitch](#)  
[praatScriptFastTrack](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Get all tokens of the KIT vowel
results <- getMatches(labbcats.url, list(segment="I"))

## Get the first 3 formants at three points during the vowel
formants <- processWithPraat(
  labbcats.url,
  results$MatchId, results$Target.segment.start, results$Target.segment.end,
  window.offset=0.025,
  praatScriptFormants(formants=c(1,2,3),
  sample.points=c(0.25,0.5,0.75)))

## End(Not run)
```

---

praatScriptIntensity *Generates a script for extracting maximum intensity, for use with [processWithPraat](#).*

---

### Description

This function generates a Praat script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#), in order to extract maximum intensity value.

### Usage

```
praatScriptIntensity(
  minimum.pitch = 100,
  time.step = 0,
  subtract.mean = TRUE,
  get.maximum = TRUE,
  sample.points = NULL,
  interpolation = "cubic",
  skip.errors = TRUE
)
```

### Arguments

<code>minimum.pitch</code>	Minimum pitch (Hz).
<code>time.step</code>	Time step in seconds, or 0.0 for 'auto'.
<code>subtract.mean</code>	Whether to subtract the mean or not.
<code>get.maximum</code>	Extract the maximum intensity for the sample.
<code>sample.points</code>	A vector of numbers ( $0 \leq \text{sample.points} \leq 1$ ) specifying multiple points at which to take the measurement. The default is NULL, meaning no individual measurements will be taken (only the aggregate values identified by <code>get.mean</code> , <code>get.minimum</code> , and <code>get.maximum</code> ). A single point at 0.5 means one measurement will be taken halfway through the target interval. If, for example, you wanted eleven measurements evenly spaced throughout the interval, you would specify <code>sample.points</code> as being <code>c(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)</code> .
<code>interpolation</code>	If <code>sample.points</code> are specified, this is the interpolation to use when getting individual values. Possible values are 'nearest', 'linear', 'cubic', 'sinc70', or 'sinc700'.
<code>skip.errors</code>	Sometimes, for some segments, Praat fails to create an Intensity object. If <code>skip.errors = TRUE</code> , analysis those segments will be skipped, and corresponding pitch values will be returned as "-undefined-". If <code>skip.errors = FALSE</code> , the error message from Praat will be returned in the Error field, but no pitch measures will be returned for any segments in the same recording.

### Value

A script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#)

**See Also**

[processWithPraat](#)  
[praatScriptFormants](#)  
[praatScriptCentreOfGravity](#)  
[praatScriptPitch](#)  
[praatScriptFastTrack](#)

**Examples**

```
## Not run:  
## define the LaBB-CAT URL  
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"  
  
## Perform a search  
results <- getMatches(labbcats.url, list(segment="s"))  
  
## Get max intensity, and intensity at three points during the segment, for all matches  
intensity <- processWithPraat(  
  labbcats.url,  
  results$MatchId, results$Target.segment.start, results$Target.segment.end,  
  praatScriptIntensity(sample.points = c(.25, .5, .75)))  
  
## End(Not run)
```

---

praatScriptPitch	<i>Generates a script for extracting pitch, for use with <a href="#">processWithPraat</a>.</i>
------------------	--

---

**Description**

This function generates a Praat script fragment which can be passed as the praat.script parameter of [processWithPraat](#), in order to extract pitch information.

**Usage**

```
praatScriptPitch(  
  get.mean = TRUE,  
  get.minimum = FALSE,  
  get.maximum = FALSE,  
  time.step = 0,  
  pitch.floor = 60,  
  max.number.of.candidates = 15,  
  very.accurate = FALSE,  
  silence.threshold = 0.03,  
  voicing.threshold = 0.5,  
  octave.cost = 0.01,  
  octave.jump.cost = 0.35,
```

```

voiced.unvoiced.cost = 0.35,
pitch.ceiling = 500,
pitch.floor.male = 30,
voicing.threshold.male = 0.4,
pitch.ceiling.male = 250,
gender.attribute = "participant_gender",
value.for.male = "M",
sample.points = NULL,
interpolation = "linear",
skip.errors = TRUE
)

```

### Arguments

<code>get.mean</code>	Extract the mean pitch for the sample.
<code>get.minimum</code>	Extract the minimum pitch for the sample.
<code>get.maximum</code>	Extract the maximum pitch for the sample.
<code>time.step</code>	Step setting for praat command
<code>pitch.floor</code>	Minimum pitch (Hz) for all speakers, or for female speakers, if <code>pitch.floor.male</code> is also specified.
<code>max.number.of.candidates</code>	Maximum number of candidates setting for praat command
<code>very.accurate</code>	Accuracy setting for praat command
<code>silence.threshold</code>	Silence threshold setting for praat command
<code>voicing.threshold</code>	Voicing threshold (Hz) for all speakers, or for female speakers, if <code>voicing.threshold.male</code> is also specified.
<code>octave.cost</code>	Octave cost setting for praat command
<code>octave.jump.cost</code>	Octave jump cost setting for praat command
<code>voiced.unvoiced.cost</code>	Voiced/unvoiced cost setting for praat command
<code>pitch.ceiling</code>	Maximum pitch (Hz) for all speakers, or for female speakers, if <code>pitch.floor.male</code> is also specified.
<code>pitch.floor.male</code>	Minimum pitch (Hz) for male speakers.
<code>voicing.threshold.male</code>	Voicing threshold (Hz) for male speakers.
<code>pitch.ceiling.male</code>	Maximum pitch (Hz) for male speakers.
<code>gender.attribute</code>	Name of the LaBB-CAT participant attribute that contains the participant's gender - normally this is "participant_gender".
<code>value.for.male</code>	The value that the <code>gender.attribute</code> has when the participant is male.

- `sample.points` A vector of numbers ( $0 \leq \text{sample.points} \leq 1$ ) specifying multiple points at which to take the measurement. The default is NULL, meaning no individual measurements will be taken (only the aggregate values identified by `get.mean`, `get.minimum`, and `get.maximum`). A single point at 0.5 means one measurement will be taken halfway through the target interval. If, for example, you wanted eleven measurements evenly spaced throughout the interval, you would specify `sample.points` as being `c(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)`.
- `interpolation` If `sample.points` are specified, this is the interpolation to use when getting individual values. Possible values are 'nearest' or 'linear'.
- `skip.errors` Sometimes, for some segments, Praat fails to create a Pitch object. If `skip.errors = TRUE`, analysis those segments will be skipped, and corresponding pitch values will be returned as "-undefined-". If `skip.errors = FALSE`, the error message from Praat will be returned in the Error field, but no pitch measures will be returned for any segments in the same recording.

### Value

A script fragment which can be passed as the `praat.script` parameter of [processWithPraat](#)

### See Also

[processWithPraat](#)  
[praatScriptFormants](#)  
[praatScriptCentreOfGravity](#)  
[praatScriptIntensity](#)  
[praatScriptFastTrack](#)

### Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Perform a search
results <- getMatches(labbcats.url, list(segment="I"))

## Get pitch mean, max, and min, and the midpoint of the segment, for each match
pitch <- processWithPraat(
  labbcats.url,
  results$MatchId, results$Target.segment.start, results$Target.segment.end,
  praatScriptPitch(get.mean=TRUE, get.minimum=TRUE, get.maximum=TRUE,
    sample.points = c(.5))

## End(Not run)
```

---

processWithPraat      *Process a set of intervals with Praat.*

---

### Description

This function instructs the LaBB-CAT server to invoke Praat for a set of sound intervals, in order to extract acoustic measures.

### Usage

```
processWithPraat(
  labbcats.url,
  match.ids,
  start.offsets,
  end.offsets,
  praat.script,
  window.offset = 0,
  gender.attribute = "participant_gender",
  attributes = NULL,
  no.progress = FALSE
)
```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
match.ids	A vector of annotation IDs, e.g. the MatchId column, or the URL column, of a results set.
start.offsets	The start time in seconds, or a vector of start times.
end.offsets	The end time in seconds, or a vector of end times.
praat.script	Script to run on each match. This may be a single string or a character vector.
window.offset	In many circumstances, you will want some context before and after the sample start/end time. For this reason, you can specify a "window offset" - this is a number of seconds to subtract from the sample start and add to the sample end time, before extracting that part of the audio for processing. For example, if the sample starts at 2.0s and ends at 3.0s, and you set the window offset to 0.5s, then Praat will extract a sample of audio from 1.5s to 3.5s, and do the selected processing on that sample.
gender.attribute	Which participant attribute represents the participant's gender.
attributes	Vector of participant attributes to make available to the script. For example, if you want to use different acoustic parameters depending on what the gender of the speaker is, including the "participant_gender" attribute will make a variable called participant_gender\$ available to the praat script, whose value will be the gender of the speaker for that segment.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

## Details

The exact measurements to return depend on the praat.script that is invoked. This is a Praat script fragment that will run once for each sound interval specified.

There are functions to allow the generation of a number of pre-defined praat scripts for common tasks such as formant, pitch, intensity, and centre of gravity – see [praatScriptFormants](#), [praatScriptCentreOfGravity](#), [praatScriptIntensity](#) and [praatScriptPitch](#).

You can provide your own script, either by building a string with your code, or loading one from a file.

LaBB-CAT prefixes praat.script with code to open a sound file and extract a defined part of it into a Sound object which is then selected.

LaBB-CAT 'Remove's this Sound object after the script finishes executing. Any other objects created by the script must be 'Remove'd before the end of the script (otherwise Praat runs out of memory during very large batches)

LaBB-CAT assumes that all calls to the function 'print' correspond to fields for export and each field must be printed on its own line. Specifically it scans for lines of the form:

```
print 'myOutputVariable' 'newline$'
```

Variables that can be assumed to be already set in the context of the script are:

- *windowOffset* – the value used for the Window Offset; how much context to include.
- *windowAbsoluteStart* – the start time of the window extracted relative to the start of the original audio file.
- *windowAbsoluteEnd* – the end time of the window extracted relative to the start of the original audio file.
- *windowDuration* – the duration of the window extracted (including window offset).
- *targetAbsoluteStart* – the start time of the target interval relative to the start of the original audio file.
- *targetAbsoluteEnd* – the end time of the target interval relative to the start of the original audio file.
- *targetStart* – the start time of the target interval relative to the start of the window extracted.
- *targetEnd* – the end time of the target interval relative to the start of the window extracted.
- *targetDuration* – the duration of the target interval.
- *sampleNumber* – the number of the sample within the set of samples being processed.
- *sampleName\$* – the name of the extracted/selected Sound object.

## Value

A data frame of acoustic measures, one row for each matchId.

## See Also

[praatScriptFormants](#)  
[praatScriptCentreOfGravity](#)  
[praatScriptIntensity](#)  
[praatScriptPitch](#)

**Examples**

```

## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## Perform a search
results <- getMatches(labbcats.url, list(segment="I"))

## get F1 and F2 for the mid point of the vowel
formants <- processWithPraat(
  labbcats.url,
  results$MatchId, results$Target.segment.start, results$Target.segment.end,
  praatScriptFormants())

## get first 3 formants at three points during the sample, the mean, min, and max
## pitch, the max intensity, and the CoG using powers 1 and 2
acoustic.measurements <- processWithPraat(
  labbcats.url,
  results$MatchId, results$Target.segment.start, results$Target.segment.end,
  paste(
    praatScriptFormants(c(1,2,3), c(0.25,0.5,0.75)),
    praatScriptPitch(get.mean=TRUE, get.minimum=TRUE, get.maximum=TRUE),
    praatScriptIntensity(),
    praatScriptCentreOfGravity(powers=c(1.0,2.0))),
  window.offset=0.5)

## execute a custom script loaded from a file
acoustic.measurements <- processWithPraat(
  labbcats.url,
  results$MatchId, results$Target.segment.start, results$Target.segment.end,
  readLines("acousticMeasurements.praat"))

## End(Not run)

```

---

removeDictionaryEntry *Removes an entry from a dictionary.*

---

**Description**

This function removes an existing entry from the given dictionary.

**Usage**

```

removeDictionaryEntry(
  labbcats.url,
  manager.id,
  dictionary.id,
  key,
  entry = NULL
)

```

### Arguments

labbcats.url	URL to the LaBB-CAT instance
manager.id	The layer manager ID of the dictionary, as returned by getDictionaries
dictionary.id	The ID of the dictionary, as returned by getDictionaries
key	The key (word) in the dictionary to remove an entry for.
entry	The value (definition) for the given key, or NULL to remove all entries for the key.

### Details

You must have edit privileges in LaBB-CAT in order to be able to use this function.

### Value

NULL if the entry was removed, or a list of error messages if not.

### See Also

[getDictionaries](#)

[getDictionaryEntries](#)

### Examples

```
## Not run:
## Remove a pronunciation of the word "robert" from the CELEX wordform pronunciation dictionary
removeDictionaryEntry(labbcats.url, "CELEX-EN", "Phonology (wordform)", "robert", "'rQ-bErt")

## End(Not run)
```

---

removeLayerDictionaryEntry

*Removes an entry from a layer dictionary.*

---

### Description

This function removes an existing entry from the dictionary that manages a given layer, and updates all affected tokens in the corpus. Words can have multiple entries.

### Usage

```
removeLayerDictionaryEntry(labbcats.url, layer.id, key, entry = NULL)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
layer.id	The ID of the layer with a dictionary configured to manage it.
key	The key (word) in the dictionary to remove an entry from.
entry	The value (definition) for the given key, or NULL to remove all entries for the given key.

**Details**

You must have edit privileges in LaBB-CAT in order to be able to use this function.

**Value**

NULL if the entry was added, or a list of error messages if not.

**See Also**

[generateLayer](#)

**Examples**

```
## Not run:
## Remove a pronunciation for "robert" from the phonemes layer dictionary
removeLayerDictionaryEntry(labbcats.url, "phonemes", "robert", "'rQ-bErt")

## End(Not run)
```

---

saveLayer

*Saves the details of an existing layer.*

---

**Description**

This function saves the definition of an existing annotation layer.

**Usage**

```
saveLayer(labbcats.url, layer)
```

**Arguments**

labbcats.url	URL to the LaBB-CAT instance
layer	A named list object representing the layer attributes, as would be returned by <a href="#">getLayer</a> or <a href="#">newLayer</a> , with members: <ul style="list-style-type: none"> <li>• <i>id</i> The layer's unique ID</li> <li>• <i>parentId</i> The layer's parent layer ID</li> </ul>

- *description* The description of the layer
- *alignment* The layer's alignment - 0 for none, 1 for point alignment, 2 for interval alignment
- *peers* Whether children have peers or not
- *peersOverlap* Whether child peers can overlap or not
- *parentIncludes* Whether the parent t-includes the child
- *saturated* Whether children must temporally fill the entire parent duration (true) or not (false)
- *parentIncludes* Whether the parent t-includes the child
- *type* The type for labels on this layer
- *validLabels* List of valid label values for this layer

### Details

You must have administration privileges in LaBB-CAT in order to be able to use this function.

### Value

The resulting layer definition, with members:

- *id* The layer's unique ID
- *parentId* The layer's parent layer ID
- *description* The description of the layer
- *alignment* The layer's alignment - 0 for none, 1 for point alignment, 2 for interval alignment
- *peers* Whether children have peers or not
- *peersOverlap* Whether child peers can overlap or not
- *parentIncludes* Whether the parent t-includes the child
- *saturated* Whether children must temporally fill the entire parent duration (true) or not (false)
- *parentIncludes* Whether the parent t-includes the child
- *type* The type for labels on this layer
- *validLabels* List of valid label values for this layer

### See Also

[newLayer](#) [getLayer](#) [deleteLayer](#)

### Examples

```
## Not run:
## Get the pronunciation layer definition
pronunciation <- getLayer(labbcats.url, "pronunciation")

## Change some details of the definition
pronunciation$description <- "CMU Dict pronunciations encoded in DISC"
pronunciation$type <- "ipa"
```

```
## Save the changes to the layer definition
saveLayer(labbcats.url, pronunciation)

## End(Not run)
```

---

updateFragment	<i>Update a transcript fragment.</i>
----------------	--------------------------------------

---

## Description

This function uploads a file (e.g. Praat TextGrid) representing a fragment of a transcript, with annotations or alignments to update in LaBB-CAT's version of the transcript.

## Usage

```
updateFragment(labbcats.url, fragment.path)
```

## Arguments

labbcats.url	URL to the LaBB-CAT instance
fragment.path	The path to the fragment to upload.

## Details

For this function to work, the credentials used to connect to the server must have at least 'edit' access.

## Value

A named list with information about the fragment that was updated.

## Examples

```
## Not run:
## define the LaBB-CAT URL
labbcats.url <- "https://labbcats.canterbury.ac.nz/demo/"

## upload new version of transcript transcript
updateFragment(labbcats.url, "my-transcript__1.234-5.678.TextGrid")

## End(Not run)
```

---

updateTranscript	<i>Update an existing transcript.</i>
------------------	---------------------------------------

---

**Description**

This function uploads a new version of an existing transcript.

**Usage**

```
updateTranscript(labbcat.url, transcript.path, no.progress = FALSE)
```

**Arguments**

labbcat.url	URL to the LaBB-CAT instance
transcript.path	The path to the transcript to upload.
no.progress	TRUE to suppress visual progress bar. Otherwise, progress bar will be shown when interactive().

**Details**

For this function to work, the credentials used to connect to the server must have at least 'edit' access.

**Value**

The ID of the updated transcript in the corpus

**Examples**

```
## Not run:  
## define the LaBB-CAT URL  
labbcat.url <- "https://labbcat.canterbury.ac.nz/demo/"  
  
## upload new version of transcript transcript  
updateTranscript(labbcat.url, "my-transcript.eaf")  
  
## End(Not run)
```

# Index

- \* **TextGrid**
  - formatTranscript, 10
  - getFragmentAnnotations, 23
  - getFragments, 25
- \* **anchor**
  - getAnchors, 15
- \* **annotation**
  - addDictionaryEntry, 3
  - addLayerDictionaryEntry, 4
  - countMatchingAnnotations, 7
  - deleteLayer, 8
  - generateLayer, 12
  - generateLayerUtterances, 13
  - getMatchAlignments, 32
  - getMatchingAnnotations, 37
  - getMatchLabels, 44
  - getParticipantAttributes, 48
  - getTranscriptAttributes, 52
  - newLayer, 59
  - removeDictionaryEntry, 78
  - removeLayerDictionaryEntry, 79
  - saveLayer, 80
- \* **annotator**
  - getAnnotatorDescriptor, 18
- \* **audio**
  - getAvailableMedia, 19
  - getMedia, 45
  - getMediaUrl, 47
- \* **connect**
  - getUserInfo, 55
  - labcatCredentials, 56
  - labcatTimeout, 57
- \* **corpora**
  - getCorpusIds, 20
  - getGraphIdsInCorpus, 27
  - getTranscriptIdsInCorpus, 54
- \* **corpus**
  - getGraphIdsInCorpus, 27
  - getTranscriptIdsInCorpus, 54
- \* **dictionary**
  - getDictionaries, 22
  - getDictionaryEntries, 22
- \* **expression**
  - countMatchingAnnotations, 7
  - getMatchingAnnotations, 37
  - getMatchingGraphIds, 39
  - getMatchingParticipantIds, 41
  - getMatchingTranscriptIds, 42
- \* **format**
  - getDeserializerDescriptors, 21
  - getSerializerDescriptors, 49
- \* **fragment**
  - getFragmentAnnotations, 23
  - getFragments, 25
  - getSoundFragments, 50
- \* **graph**
  - getGraphIdsWithParticipant, 28
  - getMatchingGraphIds, 39
  - getTranscriptIdsWithParticipant, 55
- \* **label**
  - generateLayer, 12
  - generateLayerUtterances, 13
  - getMatchAlignments, 32
  - getMatchLabels, 44
  - getParticipantAttributes, 48
  - getTranscriptAttributes, 52
- \* **layer**
  - addDictionaryEntry, 3
  - addLayerDictionaryEntry, 4
  - deleteLayer, 8
  - generateLayer, 12
  - generateLayerUtterances, 13
  - getAnnotatorDescriptor, 18
  - getLayer, 29
  - getLayerIds, 30
  - getLayers, 31
  - getMatchAlignments, 32

- getMatchLabels, 44
- getParticipantAttributes, 48
- getTranscriptAttributes, 52
- newLayer, 59
- removeDictionaryEntry, 78
- removeLayerDictionaryEntry, 79
- saveLayer, 80
- \* **lexicon**
  - deleteLexicon, 9
  - loadLexicon, 58
- \* **management**
  - deleteTranscript, 10
  - newTranscript, 61
  - updateFragment, 82
  - updateTranscript, 83
- \* **media**
  - getAvailableMedia, 19
  - getMedia, 45
  - getMediaTracks, 46
  - getMediaUrl, 47
- \* **package**
  - nzilbb.labcat, 62
- \* **participant**
  - getParticipantIds, 49
- \* **password**
  - labcatCredentials, 56
  - labcatTimeout, 57
- \* **praat**
  - praatScriptCentreOfGravity, 66
  - praatScriptFastTrack, 67
  - praatScriptFormants, 70
  - praatScriptIntensity, 72
  - praatScriptPitch, 73
  - processWithPraat, 76
- \* **sample**
  - getFragmentAnnotations, 23
  - getFragments, 25
  - getSoundFragments, 50
- \* **search**
  - getAllUtterances, 14
  - getMatches, 33
- \* **sound**
  - getMediaTracks, 46
  - getSoundFragments, 50
- \* **speaker**
  - getParticipantIds, 49
- \* **timeout**
  - labcatCredentials, 56
  - labcatTimeout, 57
- \* **transcript**
  - countAnnotations, 6
  - deleteTranscript, 10
  - formatTranscript, 10
  - getAnnotations, 16
  - getGraphIds, 26
  - getGraphIdsWithParticipant, 28
  - getMatchingGraphIds, 39
  - getMatchingParticipantIds, 41
  - getMatchingTranscriptIds, 42
  - getTranscriptIds, 53
  - getTranscriptIdsWithParticipant, 55
  - newTranscript, 61
  - updateFragment, 82
  - updateTranscript, 83
- \* **username**
  - getUserInfo, 55
  - labcatCredentials, 56
  - labcatTimeout, 57
- \* **wav**
  - getSoundFragments, 50
- addDictionaryEntry, 3
- addLayerDictionaryEntry, 4
- annotatorExt, 5, 19
- countAnnotations, 6, 17
- countMatchingAnnotations, 7, 38
- deleteLayer, 8, 60, 81
- deleteLexicon, 9, 59
- deleteTranscript, 10
- formatTranscript, 10
- generateLayer, 4, 12, 59, 60, 80
- generateLayerUtterances, 13
- getAllUtterances, 12, 13, 14
- getAnchors, 15
- getAnnotations, 16, 16
- getAnnotatorDescriptor, 18
- getAvailableMedia, 19
- getCorpusIds, 20
- getDeserializerDescriptors, 21
- getDictionaries, 4, 22, 23, 59, 79
- getDictionaryEntries, 4, 22, 22, 79
- getFragmentAnnotations, 23

getFragments, [21](#), [24](#), [25](#), [37](#), [49](#), [50](#)  
getGraphIds, [26](#)  
getGraphIdsInCorpus, [27](#), [27](#)  
getGraphIdsWithParticipant, [28](#)  
getId, [28](#)  
getLayer, [29](#), [80](#), [81](#)  
getLayerIds, [30](#), [30](#), [31](#)  
getLayers, [30](#), [31](#), [52](#)  
getMatchAlignments, [32](#), [37](#), [45](#)  
getMatches, [33](#), [33](#), [45](#)  
getMatchingAnnotations, [7](#), [37](#)  
getMatchingGraphIds, [39](#)  
getMatchingParticipantIds, [41](#)  
getMatchingTranscriptIds, [42](#)  
getMatchLabels, [33](#), [37](#), [44](#)  
getMedia, [45](#), [47](#)  
getMediaTracks, [46](#)  
getMediaUrl, [46](#), [47](#)  
getParticipantAttributes, [48](#)  
getParticipantIds, [15](#), [37](#), [49](#), [55](#)  
getSerializerDescriptors, [11](#), [25](#), [26](#), [49](#)  
getSoundFragments, [24](#), [37](#), [50](#)  
getSystemAttribute, [52](#)  
getTranscriptAttributes, [52](#)  
getTranscriptIds, [6](#), [17](#), [20](#), [26](#), [46](#), [47](#), [53](#)  
getTranscriptIdsInCorpus, [6](#), [17](#), [54](#)  
getTranscriptIdsWithParticipant, [6](#), [17](#),  
[28](#), [55](#)  
getUserInfo, [55](#)  
  
labbcacredentials, [56](#), [56](#)  
labbcacTimeout, [57](#)  
loadLexicon, [9](#), [58](#)  
  
newLayer, [8](#), [19](#), [59](#), [80](#), [81](#)  
newTranscript, [61](#)  
nzilbb.labbcac, [62](#)  
  
praascriptCentreOfGravity, [66](#), [69](#), [71](#),  
[73](#), [75](#), [77](#)  
praascriptFastTrack, [66](#), [67](#), [71](#), [73](#), [75](#)  
praascriptFormants, [66](#), [69](#), [70](#), [73](#), [75](#), [77](#)  
praascriptIntensity, [66](#), [69](#), [71](#), [72](#), [75](#), [77](#)  
praascriptPitch, [66](#), [69](#), [71](#), [73](#), [73](#), [77](#)  
processWithPraat, [37](#), [66](#), [67](#), [69–73](#), [75](#), [76](#)  
  
removeDictionaryEntry, [78](#)  
removeLayerDictionaryEntry, [79](#)  
  
saveLayer, [8](#), [60](#), [80](#)  
  
updateFragment, [82](#)  
updateTranscript, [83](#)