Package ‘AzureCognitive’

November 12, 2019

Title Interface to Azure Cognitive Services
Version 1.0.0
Description An interface to Azure Cognitive Services <https://docs.microsoft.com/azure/cognitive- services/welcome>. Both an ‘Azure Resource Manager’ interface, for deploying Cognitive Services resources, and a client framework are supplied. While ‘AzureCognitive’ can be called by the end-user, it is meant to provide a foundation for other packages that will support specific services, like Computer Vision, Custom Vision, language translation, and so on. Part of the 'AzureR' family of packages.

URL https://github.com/Azure/AzureCognitive
https://github.com/Azure/AzureR
BugReports https://github.com/Azure/AzureCognitive/issues
License MIT + file LICENSE
Depends R (>= 3.3)
Imports AzureAuth (>= 1.2.0), AzureRMR, jsonlite, httr (>= 1.3),
Suggests knitr, testthat
VignetteBuilder knitr
RoxygenNote 6.1.1
NeedsCompilation no
Author Hong Ooi [aut, cre],
Microsoft [cph]
Maintainer Hong Ooi <hongooi@microsoft.com>
Repository CRAN
Date/Publication 2019-11-12 12:50:02 UTC

R topics documented:

az_cognitive_service .................................................. 2
call_cognitive_endpoint ............................................. 3
cognitive_endpoint ................................................... 5
create_cognitive_service .......................................... 6
get_cognitive_token ................................................ 8
Description

Class representing a cognitive service resource, exposing methods for working with it.

Usage

az_cognitive_service

Format

An object of class R6ClassGenerator of length 24.

Methods

The following methods are available, in addition to those provided by the AzureRMR::az_resource class:

- list_keys(): Return the access keys for this service.
- get_endpoint(): Return the service endpoint, along with an access key. See ‘Endpoints’ below.
- regen_key(key): Regenerates (creates a new value for) an access key. The argument key can be 1 or 2.
- list_service_tiers(): List the service pricing tiers (SKUs) available for this service.

Initialization

Initializing a new object of this class can either retrieve an existing service, or create a new service on the host. Generally, the best way to initialize an object is via the get_cognitive_service and create_cognitive_service methods of the az_resource_group class, which handle the details automatically.

Endpoints

The client-side interaction with a cognitive service is via an endpoint. Endpoint interaction in AzureCognitive is implemented using S3 classes. You can create a new endpoint object via the get_endpoint() method, or with the standalone cognitive_endpoint() function. If you use the latter, you will also have to supply any necessary authentication credentials, eg a subscription key or token.

See Also

cognitive_endpoint, create_cognitive_service, get_cognitive_service
Examples

```r
## Not run:

# recommended way of creating a new resource: via resource group method
rg <- AzureRMR::get_azure_login()
  get_subscription("sub_id")
  get_resource_group("rgname")
cogsvc <- rg$create_cognitive_service("myvisionservice",
          service_type="ComputerVision", service_tier="F0")
cogsvc$list_service_tiers()
cogsvc$list_keys()
cogsvc$regen_key()
cogsvc$get_endpoint()

## End(Not run)
```

call_cognitive_endpoint

Call a Cognitive Service REST endpoint

Description

Call a Cognitive Service REST endpoint

Usage

```r
call_cognitive_endpoint(endpoint, ...)
```

## S3 method for class 'cognitive_endpoint'
call_cognitive_endpoint(endpoint, operation, options = list(), headers = list(), body = NULL, encode = NULL, ..., http_verb = c("GET", "POST", "PUT", "PATCH", "DELETE", "HEAD"), http_status_handler = c("stop", "warn", "message", "pass"))

Arguments

- `endpoint` An object of class `cognitive_endpoint`.
- `...` Further arguments passed to lower-level functions. For the default method, these are passed to `httr::content`; in particular, you can convert a structured JSON response into a data frame by specifying `simplifyDataFrame=TRUE`.
- `operation` The operation to perform.
- `options` Any query parameters that the operation takes.
- `headers` Any optional HTTP headers to include in the REST call. Note that `call_cognitive_endpoint` will handle authentication details automatically, so don’t include them here.
body

The body of the HTTP request for the REST call.

encode

The encoding (really content-type) for the body. See the encode argument for http::POST. The default value of NULL will use raw encoding if the body is a raw vector, and json encoding for anything else.

http_verb

The HTTP verb for the REST call.

http_status_handler

How to handle a failed REST call. stop, warn and message will call the corresponding *_for_status handler in the httr package; pass will return the raw response object unchanged. The last one is mostly intended for debugging purposes.

Details

This function does the low-level work of constructing a HTTP request and then calling the REST endpoint. It is meant to be used by other packages that provide higher-level views of the service functionality.

Value

For a successful REST call, the contents of the response. This will usually be a list, obtained by translating the raw JSON body into R. If the call returns a non-success HTTP status code, based on the http_status_handler argument.

See Also

cognitive_endpoint, create_cognitive_service, get_cognitive_service

Examples

## Not run:

```r
endp <- cognitive_service("https://myvisionservice.api.cognitive.azure.com",
  service_type="Computervision", key="key")

# analyze an online image
call_cognitive_endpoint(endp,
  operation="analyze",
  body=list(url=img_link),
  options=list(details="celebrities"),
  http_verb="POST")

# analyze an image on the local machine
img_raw <- readBin("image.jpg", "raw", file.info("image.jpg")$size)
call_cognitive_endpoint(endp,
  operation="analyze",
  body=img_raw,
  encode="raw",
  http_verb="POST")
```
cognitive_endpoint

## End(Not run)

<table>
<thead>
<tr>
<th>cognitive_endpoint</th>
<th>Object representing an Azure Cognitive Service endpoint</th>
</tr>
</thead>
</table>

**Description**

Object representing an Azure Cognitive Service endpoint

**Usage**

```r
cognitive_endpoint(url, service_type, key = NULL, aad_token = NULL, cognitive_token = NULL, auth_header = "ocp-apim-subscription-key")
```

**Arguments**

- `url`: The URL of the endpoint.
- `service_type`: What type (or kind) of service the endpoint provides. See below for the services that AzureCognitive currently recognises.
- `key`: The subscription key (single- or multi-service) to use to authenticate with the endpoint.
- `aad_token`: An Azure Active Directory (AAD) OAuth token, as an alternative to a key for the services that allow AAD authentication.
- `cognitive_token`: A Cognitive Service token, as another alternative to a key for the services that accept it.
- `auth_header`: The name of the HTTP request header for authentication. Only used if a subscription key is supplied.

**Details**

Currently, `cognitive_endpoint` recognises the following service types:

- `CognitiveServices`: multiple service types
- `ComputerVision`: generic computer vision service
- `Face`: face recognition
- `LUIS`: language understanding
- `CustomVision.Training`: Training endpoint for a custom vision service
- `CustomVision.Prediction`: Prediction endpoint for a custom vision service
- `ContentModerator`: Content moderation (text and images)
- `Text`: text analytics
- `TextTranslate`: text translation
create_cognitive_service

Value
An object inheriting from class cognitive_endpoint, that can be used to communicate with the
REST endpoint. The subclass of the object indicates the type of service provided.

See Also
call_cognitive_endpoint, create_cognitive_service, get_cognitive_service

Examples
## Not run:
cognitive_service("https://myvisionservice.api.cognitive.azure.com",
  service_type="Computervision", key="key")
cognitive_service("https://mylangservice.api.cognitive.azure.com",
  service_type="LUIS", key="key")
# authenticating with AAD
token <- AzureAuth::get_azure_token("https://cognitiveservices.azure.com",
  tenant="mytenant", app="app_id", password="password")
cognitive_service("https://myvisionservice.api.cognitive.azure.com",
  service_type="Computervision", aad_token=token))
## End(Not run)

create_cognitive_service
Create, retrieve or delete an Azure Cognitive Service

Description
Methods for the AzureRMR::az_resource_group class.

Usage
create_cognitive_service(name, service_type, service_tier, location = self$location,
  subdomain = name, properties = list(), ...)
get_cognitive_service(name)
delete_cognitive_service(name, confirm = TRUE, wait = FALSE)

Arguments
• name: The name for the cognitive service resource.
• service_type: The type of service (or "kind") to create. See 'Details' below.
• service_tier: The pricing tier (SKU) for the service.
create_cognitive_service

- **location**: The Azure location in which to create the service. Defaults to the resource group’s location.
- **subdomain**: The subdomain name to assign to the service; defaults to the resource name. Set this to NULL if you don’t want to assign the service a subdomain of its own.
- **properties**: For `create_cognitive_service`, an optional named list of other properties for the service.
- **confirm**: For `delete_cognitive_service`, whether to prompt for confirmation before deleting the resource.
- **wait**: For `delete_cognitive_service`, whether to wait until the deletion is complete before returning.

**Details**

These are methods to create, get or delete a cognitive service resource within a resource group. For `create_cognitive_service`, the type of service created can be one of the following:

- **CognitiveServices**: multiple service types
- **ComputerVision**: generic computer vision service
- **Face**: face recognition
- **LUIS**: language understanding
- **CustomVision.Training**: Training endpoint for a custom vision service
- **CustomVision.Prediction**: Prediction endpoint for a custom vision service
- **ContentModerator**: Content moderation (text and images)
- **Text**: text analytics
- **TextTranslate**: text translation

The possible tiers depend on the type of service created. Consult the Azure Cognitive Service documentation for more information. Usually there will be at least one free tier available.

**Value**

For `create_cognitive_service` and `get_cognitive_service`, an object of class `az_cognitive_service`.

**See Also**

cognitive_endpoint, call_cognitive_endpoint

Azure Cognitive Services documentation, REST API reference

**Examples**

```r
## Not run:
rg <- AzureRMR::get_azure_login()$get_subscription("sub_id")$get_resource_group("rgname")
```
rg$create_cognitive_service("myvisionservice",
    service_type="ComputerVision", service_tier="F0")

rg$create_cognitive_service("mylangservice",
    service_type="LUIS", service_tier="F0")

rg$get_cognitive_service("myvisionservice")

rg$delete_cognitive_service("myvisionservice")

## End(Not run)

get_cognitive_token  

Obtain authentication token for a cognitive service

Description

Obtain authentication token for a cognitive service

Usage

get_cognitive_token(key, region, token_host = NULL)

Arguments

key  The subscription key for the service.
region  The Azure region where the service is located.
token_host  Optionally, the URL for the token endpoint.
Index

*Topic datasets
  az_cognitive_service, 2
  az_resource_group, 2
  AzureRMR::az_resource, 2
  AzureRMR::az_resource_group, 6

  call_cognitive_endpoint, 3, 6, 7
  cognitive_endpoint, 2, 4, 5, 7
  create_cognitive_service, 2, 4, 6, 6
  delete_cognitive_service
    (create_cognitive_service), 6

  get_cognitive_service, 2, 4, 6
  get_cognitive_service
    (create_cognitive_service), 6
  get_cognitive_token, 8

httr::content, 3
httr::POST, 4