Package 'AutoDeskR'

January 20, 2025

Type Package

Title An Interface to the 'AutoDesk' 'API' Platform

Description An interface to the 'AutoDesk' 'API' Platform including the Authentication 'API' for obtaining authentication to the 'AutoDesk' Forge Platform, Data Management 'API' for managing data across the platform's cloud services, Design Automation 'API' for performing automated tasks on design files in the cloud, Model Derivative 'API' for translating design files into different formats, sending them to the viewer app, and extracting design data, and Viewer for rendering 2D and 3D models.

Version 0.1.5

URL https://aps.autodesk.com, https://paulgovan.gitbook.io/autodeskr, http://paulgovan.github.io/AutoDeskR/, https://github.com/paulgovan/AutoDeskR

BugReports https://github.com/paulgovan/AutoDeskR/issues

Depends R (>= 2.10.0)

License Apache License | file LICENSE

Imports httr, jsonlite, shiny

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Paul Govan [aut, cre, cph] (<https://orcid.org/0000-0002-1821-8492>)

Maintainer Paul Govan <paul.govan2@gmail.com>

Repository CRAN

Date/Publication 2024-09-10 23:40:18 UTC

Contents

checkBucket . . . 2
checkFile . . . 3
checkPdf . . . 3
downloadFile . . . 4
getData . . . 5
getMetadata . . . 5
getObjectTree . . . 6
getOutputUrn . . . 7
getToken . . . 8
makeBucket . . . 8
makePdf . . . 9
translateObj . . . 10
translateSvf . . . 10
uploadFile . . . 11
viewer3D . . . 12
viewerUI . . . 13

Index **14**

`checkBucket` *Check the Status of an App-Managed Bucket.*

Description

Check the status of a recently created app-managed bucket using the Data Management API.

Usage

```
checkBucket(token = NULL, bucket = "mybucket")
```

Arguments

`token`	A string. Token generated with `getToken` function with `bucket:create`, `bucket:read`, and `data:write` scopes.
`bucket`	A string. Name of the bucket. Defaults to `mybucket`.

Value

An object containing the `bucketKey`, `bucketOwner`, and `createdDate`.

Examples

```
## Not run:
# Check the status of a bucket with the name "mybucket"
resp <- checkBucket(token = myToken, bucket = "mybucket")
resp

## End(Not run)
```

checkFile *Check the Status of a Translated File.*

Description

Check the status of a recently translated file using the Model Derivative API.

Usage

```
checkFile(urn = NULL, token = NULL)
```

Arguments

urn	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
token	A string. Token generated with `getToken` function with `data:read` and `data:write` scopes.

Examples

```
## Not run:
# Check the status of the translated "aerial.dwg" svf file
resp <- checkFile(urn = myEncodedUrn, token = myToken)
resp

## End(Not run)
```

checkPdf *Check the status of a PDF.*

Description

Check the status of a recently created PDF file using the Design Automation API.

Usage

```
checkPdf(source = NULL, destination = NULL, token = NULL)
```

Arguments

source	A string. Publicly accessible web address of the input dwg file.
destination	A string. Publicly accessible web address for the output pdf file.
token	A string. Token generated with `getToken` function with `code:all` scope.

Examples

```
## Not run:
mySource <- "http://download.autodesk.com/us/samplefiles/acad/visualization_-_aerial.dwg"
myDestination <- "https://drive.google.com/folderview?id=0BygncDVHf60mTDZVNDltLThLNmM&usp=sharing"
resp <- checkPdf(mySource, myDestination, token = myToken)
resp

## End(Not run)
```

`downloadFile` *Download a file locally.*

Description

Download a file from the Forge Platform using the Model Derivative API.

Usage

```
downloadFile(urn = NULL, output_urn = NULL, token = NULL)
```

Arguments

`urn`	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
`output_urn`	A string. Output_urn retrieved via `getOutputUrn`
`token`	A string. Token generated with `getToken` function with `data:read` and `data:write` scopes.

Value

An object containing the `result`, `urn`, and additional activity information.

Examples

```
## Not run:
# Download the "aerial.dwg" png file
myEncodedOutputUrn <- jsonlite::base64_enc(myOutputUrn)
resp <- downloadFile(urn <- myEncodedUrn, output_urn <- myEncodedOutputUrn, token = myToken)

## End(Not run)
```

getData *Get the Geometry Data for a File.*

Description

Get the geometry of an uploaded file using the Model Derivative API.

Usage

```
getData(guid = NULL, urn = NULL, token = NULL)
```

Arguments

guid	A string. GUID retrieved via the getMetadata function.
urn	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the jsonlite::base64_enc function.
token	A string. Token generated with getToken function with data:read and data:write scopes.

Value

An object containing the geometry data for the selected file.

Examples

```
## Not run:
# Get the geometry data for the "aerial.dwg" svf file
resp <- getData(guid <- myGuid, urn <- myEncodedUrn, token = myToken)

## End(Not run)
```

getMetadata *Get the Metadata for a File.*

Description

Get the metadata of an uploaded file using the Model Derivative API.

Usage

```
getMetadata(urn = NULL, token = NULL)
```

Arguments

urn	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
token	A string. Token generated with `getToken` function with `data:read` and `data:write` scopes.

Value

An object containing the `type`, `name`, and `guid` of the file.

Examples

```
## Not run:
# Get the metadata for the "aerial.dwg" svf file
resp <- getMetadata(urn <- myEncodedUrn, token = myToken)
myGuid <- resp$content$data$metadata[[1]]$guid

## End(Not run)
```

`getObjectTree` *Get the Object Tree of a File.*

Description

Get the object tree of an uploaded file using the Model Derivative API.

Usage

```
getObjectTree(guid = NULL, urn = NULL, token = NULL)
```

Arguments

guid	A string. GUID retrieved via the `getMetadata` function.
urn	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
token	A string. Token generated with `getToken` function with `data:read` and `data:write` scopes.

Value

An object containing the object tree for the selected file. the file.

Examples

```
## Not run:
# Get the object tree for the "aerial.dwg" svf file
resp <- getObjectTree(guid <- myGuid, urn <- myEncodedUrn, token = myToken)
resp

## End(Not run)
```

`getOutputUrn` *Get the Output URN for a File.*

Description

Get the output urn of a translated file using the Model Derivative API.

Usage

```
getOutputUrn(urn, token)
```

Arguments

`urn`	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
`token`	A string. Token generated with `getToken` function with `data:read` and `data:write` scopes.

Value

An object containing the `result`, `urn`, and additional activity information.

Examples

```
## Not run:
# Get the output urn for the "aerial.dwg" obj file
resp <- getOutputUrn(urn <- myUrn, token = Sys.getenv("token"))
resp

## End(Not run)
```

getToken *Get a 2-Legged Token for Authentication.*

Description

Get a 2-legged token for OAuth-based authentication to the AutoDesk Forge Platform.

Usage

```
getToken(id = NULL, secret = NULL, scope = "data:write data:read")
```

Arguments

id	A string. Client ID for the app generated from the AutoDesk Dev Portal.
secret	A string. Client Secret for the app generated from the AutoDesk Dev Portal.
scope	A string. Space-separated list of required scopes. May be `user-profile:read`, `data:read`, `data:write`, `data:create`, `data:search`, `bucket:create`, `bucket:read`, `bucket:update`, `bucket:delete`, `code:all`, `account:read`, `account:write`, or a combination of these.

Value

An object containing the `access_token`, `code_type`, and `expires_in` milliseconds.

Examples

```
## Not run:
# Get a 2-legged token with the "data:read" and "data:write" scopes
resp <- getToken(id = Sys.getenv("client_id"), secret = Sys.getenv("client_secret"),
           scope = "data:write data:read")
myToken <- resp$content$access_token

## End(Not run)
```

makeBucket *Make a Bucket for an App.*

Description

Make an app-based bucket for storage of design files using the Data Management API.

Usage

```
makeBucket(token = NULL, bucket = "mybucket", policy = "transient")
```

Arguments

token	A string. Token generated with getToken function with bucket:create, bucket:read, and data:write scopes.
bucket	A string. Unique bucket name. Defaults to mybucket.
policy	A string. May be transient, temporary, or persistent.

Value

An object containing the bucketKey, bucketOwner, and createdDate.

Examples

```
## Not run:
# Make a transient bucket with the name "mybucket"
resp <- makeBucket(token = myToken, bucket = "mybucket", policy = "transient")

## End(Not run)
```

---

makePdf *Convert a DWG to a PDF.*

Description

Convert a publicly accessible DWG file to a publicly accessible PDF using the Design Automation API.

Usage

```
makePdf(source = NULL, destination = NULL, token = NULL)
```

Arguments

source	A string. Publicly accessible web address of the input dwg file.
destination	A string. Publicly accessible web address for the output pdf file.
token	A string. Token generated with getToken function with code:all scope.

Examples

```
## Not run:
mySource <- "http://download.autodesk.com/us/samplefiles/acad/visualization_-_aerial.dwg"
myDestination <- "https://drive.google.com/folderview?id=0BygncDVHf60mTDZVNDltLThLNmM&usp=sharing"
resp <- makePdf(mySource, myDestination, token = myToken)

## End(Not run)
```

translateObj *Translate a File into OBJ Format.*

Description

Translate an uploaded file into OBJ format using the Model Derivative API.

Usage

```
translateObj(urn = NULL, token = NULL)
```

Arguments

urn	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
token	A string. Token generated with `getToken` function with `data:read` and `data:write` scopes.

Value

An object containing the `result`, `urn`, and additional activity information.

Examples

```
## Not run:
# Translate the "aerial.dwg" file into a obj file
resp <- translateObj(urn <- myEncodedUrn, token = myToken)

## End(Not run)
```

translateSvf *Translate a File into SVF Format.*

Description

Translate an uploaded file into SVF format using the Model Derivative API.

Usage

```
translateSvf(urn = NULL, token = NULL)
```

Arguments

urn	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the jsonlite::base64_enc function.
token	A string. Token generated with getToken function with data:read and data:write scopes.

Value

An object containing the result, urn, and additional activity information.

Examples

```
## Not run:
# Translate the "aerial.dwg" file into a svf file
myEncodedUrn <- jsonlite::base64_enc(myUrn)
resp <- translateSvf(urn = myEncodedUrn, token = myToken)

## End(Not run)
```

---

uploadFile *Upload a File to an App-Managed Bucket.*

Description

Upload a design file to an app-managed bucket using the Data Management API.

Usage

```
uploadFile(file = NULL, token = NULL, bucket = "mybucket")
```

Arguments

file	A string. File path.
token	A string. Token generated with getToken function with bucket:create, bucket:read, and data:write scopes.
bucket	A string. Unique bucket name. Defaults to mybucket.

Value

An object containing the bucketKey, objectId (i.e. urn), objectKey (i.e. file name), size, contentType (i.e. "application/octet-stream"), location. and other content information.

Examples

```
## Not run:
# Upload the "aerial.dwg" file to "mybucket"
resp <- uploadFile(file = system.file("inst/samples/aerial.dwg", package = "AutoDeskR"),
           token = myToken, bucket = "mybucket")
myUrn <- resp$content$objectId

## End(Not run)
```

`viewer3D` *Launch the Viewer.*

Description

Launch the Viewer.

Usage

```
viewer3D(urn = NULL, token = NULL, viewerType = "header")
```

Arguments

`urn`	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
`token`	A string. Token generated with `getToken` function with `data:read` scope.
`viewerType`	A string. The type of viewer to instantiate. Either "header" for the default viewer, "headless" for a viewer without toolbar or panels, or "vr" to enter WebVR mode on a mobile device.

Examples

```
## Not run:
# View the "aerial.dwg" file in the AutoDesk viewer
myEncodedUrn <- jsonlite::base64_enc(myUrn)
viewer3D(urn <- myEncodedUrn, token = myToken)

## End(Not run)
```

viewerUI *UI Module Function.*

Description

UI Module Function.

Usage

```
viewerUI(id, urn = NULL, token = NULL, viewerType = "header")
```

Arguments

id	A string. A namespace for the module.
urn	A string. Source URN (objectId) for the file. Note the URN must be Base64 encoded. To encode the URN, see, for example, the `jsonlite::base64_enc` function.
token	A string. Token generated with `getToken` function with `data:read` scope.
viewerType	A string. The type of viewer to instantiate. Either "header" for the default viewer or "headless" for a viewer without toolbar or panels.

Examples

```
## Not run:
ui <- function(request) {
 shiny::fluidPage(
   viewerUI("pg", myEncodedUrn, myToken)
 )
}
server <- function(input, output, session) {
}
shiny::shinyApp(ui, server)

## End(Not run)
```

Index

checkBucket, 2
checkFile, 3
checkPdf, 3

downloadFile, 4

getData, 5
getMetadata, *5*, 5, *6*
getObjectTree, 6
getOutputUrn, *4*, 7
getToken, *2–7*, 8, *9–13*

makeBucket, 8
makePdf, 9

translateObj, 10
translateSvf, 10

uploadFile, 11

viewer3D, 12
viewerUI, 13