Package: grabr (via r-universe)

October 30, 2024

Type Package

Title OHA/SI APIs Package

Version 2.1.2

Description Provides a series of base functions useful to the GH OHA SI team. These function extend the utility functions in glamr, focusing primarily on API utility functions.

License MIT + file LICENSE

URL https://github.com/USAID-OHA-SI/grabr, https://usaid-oha-si.github.io/grabr/

BugReports https://github.com/USAID-OHA-SI/grabr/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports aws.s3, crayon, curl, dplyr, getPass, glue, httr, jsonlite, lubridate, magrittr, purrr, readxl, rvest, stringr, tools, tidyr, tidyselect, usethis, urltools, stats, cli

Suggests fs, zip, googlesheets4, janitor, keyring, knitr, plyr, rmarkdown, testthat (>= 3.0.0), tibble, utils, vroom, countrycode

Remotes USAID-OHA-SI/glamr, USAID-OHA-SI/gagglr

VignetteBuilder knitr

Depends R (>= 4.0)

Config/testthat/edition 3

Repository https://usaid-oha-si.r-universe.dev

RemoteUrl https://github.com/USAID-OHA-SI/grabr

RemoteRef HEAD

RemoteSha 483438770cc8e88ecd2e23fe16e13516b54ddb48

Contents

`datim_dimension` *Get PEPFAR/DATIM Dimension ID*

Description

Get PEPFAR/DATIM Dimension ID

Usage

```
datim_dimension(name, username, password, baseurl = "https://final.datim.org/")
```

Arguments

`name`	Dimension name
`username`	DATIM Account Username, recommended using glamr::datim_user()‘
`password`	DATIM Account passward, recommended using glamr::datim_pwd()‘
`baseurl`	DATIM API End point, default value is ‘https://final.datim.org/‘

Value

dimension uid

Examples

```
## Not run:
  library(grabr)
  datim_dimension("OU Level")

## End(Not run)
```

`datim_dimensions` *Get PEPFAR/DATIM dimensions*

Description

Get PEPFAR/DATIM dimensions

Usage

```
datim_dimensions(
  username,
  password,
  var = NULL,
  baseurl = "https://final.datim.org"
)
```

Arguments

`username`	DATIM Account Username
`password`	DATIM Account passward
`var`	Column name to pull all values from, default is NULL, options are: id, dimension
`baseurl`	DATIM API End point, , default value is 'https://final.datim.org'

Value

Dimensions as tibble or list of ids / dimension names

Examples

```
## Not run:
  library(grabr)

  datim_dimensions()

## End(Not run)
```

datim_dim_item *Get dimension / item id*

Description

Get dimension / item id

Usage

```
datim_dim_item(
  dimension,
  name,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

dimension	Dimension name
name	Item name
username	DATIM Account Username
password	DATIM Account password
baseurl	DATIM API end point

Value

UID of item

Examples

```
## Not run:
  library(grabr)

  datim_dim_item(dimension = "Funding Agency", name = "USAID")

  datim_dim_item(dimension = "Targets / Results", name = "MER Results")
  datim_dim_item(dimension = "Targets / Results", name = "MER Targets")

## End(Not run)
```

`datim_dim_items` *Get PEPFAR/DATIM Dimension Items*

Description

Get PEPFAR/DATIM Dimension Items

Usage

```
datim_dim_items(
  dimension,
  username,
  password,
  var = NULL,
  fields = NULL,
  baseurl = "https://final.datim.org/"
)
```

Arguments

`dimension`	Dimension name
`username`	DATIM Account Username
`password`	DATIM Account passward
`var`	column name to pull values from, id or item
`fields`	list of column names to return, this will overwrite ‘var‘
`baseurl`	DATIM API end point

Value

Dimension’s items as tibble or vector

Examples

```
## Not run:
  library(grabr)

  datim_dim_items(dimension = "Funding Agency")
  datim_dim_items(dimension = "Funding Agency", var = "item")

## End(Not run)
```

`datim_dim_url` *Build PEPFAR/DATIM Query dimension*

Description

Build PEPFAR/DATIM Query dimension

Usage

```
datim_dim_url(
  dimension,
  items = NULL,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

`dimension`	Dimension name
`items`	Item name
`username`	DATIM Account Username
`password`	DATIM Account passward
`baseurl`	DATIM API End point

Value

Valid DATIM Query Params url

Examples

```
## Not run:
  library(grabr)

  datim_dim_url(dimension = "Sex")

  datim_dim_url(
    dimension = "Disaggregation Type",
    items = "Age/Sex/HIVStatus"
  )

  datim_dim_url(
    dimension = "Disaggregation Type",
    items = c("Age/Sex", "Age/Sex/HIVStatus")
  )
```

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

`datim_execute_query` *Execute Datim Query*

Description

Execute Datim Query

Usage

```
datim_execute_query(url, username, password, flatten = FALSE)
```

Arguments

`url`	API Base url & all query parameters
`username`	Datim username, recommend using ‘glamr::datim_user()‘
`password`	Datim password, recommend using ‘glamr::datim_pwd()‘
`flatten`	Should query json result be flatten? Default is false

Value

returns query results as json object, or NULL when error occurs.

Examples

```
## Not run:
  library(grabr)

  datim_execute_query(
    url = 'https://www.datim.org/api/sqlViews/<uid>?format=json',
    username =glamr::datim_user(),
    password =glamr::datim_pwd(),
    flatten = TRUE
  )

## End(Not run)
```

datim_mechs *Extract Mechanisms infos from Datim*

Description

Extract Mechanisms infos from Datim

Usage

```
datim_mechs(cntry, username, password, agency = "USAID", baseurl = NULL)
```

Arguments

cntry	Country name
username	Datim username
password	Datim password
agency	Agency name
baseurl	Datim API Base URL

Value

OU Mechanisms as data frame

Examples

```
## Not run:
  library(grabr)

  datim_mechs(
   cntry = "Mozambique",
   username = glamr::datim_user(),
   password = glamr::datim_pwd()
  )

## End(Not run)
```

datim_orgunits *Pull Orgunits SQLView*

Description

Pull Orgunits SQLView

Usage

```
datim_orgunits(
  cntry,
  username,
  password,
  reshape = FALSE,
  baseurl = "https://final.datim.org/"
)
```

Arguments

cntry	Country name
username	Datim username
password	Datim password
reshape	Unpack parent org units
baseurl	Datim API Base URL, default to https://final.datim.org/

Value

OU/Country Orgunits as a data frame

Examples

```
## Not run:
  library(grabr)

  datim_orgunits(
   cntry = "Mozambique",
   username = glamr::datim_user(),
   password = glamr::datim_pwd()
  )

## End(Not run)
```

---

datim_pops *Extract PLHIV and General POP Estimates from datim*

Description

Extract PLHIV and General POP Estimates from datim

Usage

```
datim_pops(
  ou,
  username,
  password,
  level = "country",
  fy = NULL,
  hierarchy = FALSE,
  baseurl = "https://final.datim.org/"
)
```

Arguments

ou	Operatingunit
username	Datim account username
password	Datim account password
level	Organization level
fy	Fiscal Year
hierarchy	Should additional organizational hierarchy be added?, default is FALSE
baseurl	DATIM API base url

Value

PLHIV and POP_EST Data

Examples

```
## Not run:
  library(grabr)

  datim_pops(ou = "Nigeria")

  datim_pops(ou = "Nigeria", fy = 2021)

  datim_pops(ou = "Nigeria", level = "psnu", fy = 2021, hierarchy = TRUE)

## End(Not run)
```

---

datim_process_query *Process Datim Query results*

Description

Process Datim Query results

Usage

```
datim_process_query(url, username, password)
```

Arguments

url	Datim API Call url
username	Datim username, recommend using ‘glamr::datim_user()‘
password	Datim password, recommend using ‘glamr::datim_pwd()‘

Value

Data as tibble or NULL when error occurs.

Examples

```
## Not run:
 library(grabr)
 datim_process_query("<full-api-call-url>")

## End(Not run)
```

---

datim_pull_hierarchy *Extract PEPFAR Org Hierarchy*

Description

Extract PEPFAR Org Hierarchy

Usage

```
datim_pull_hierarchy(
  ou_uid,
  username,
  password,
  add_geom = FALSE,
  baseurl = "https://final.datim.org/",
  folderpath_output = NULL
)
```

Arguments

ou_uid	UID for the country, recommend using identify_ouuids
username	DATIM username, recommend using datim_user
password	DATIM password, recommend using datim_pwd
add_geom	Add geometry column to the output, default is false

baseurl API base url, default = https://final.datim.org/

folderpath_output
provide the full path to the folder for saving

Note

This function is migrated from Wavelength and is similar to datim_orgunits

Examples

```
## Not run:
#get OU UID
  ouuid <- identify_ouuids() %>% dplyr::filter(country == "Kenya")
#pull hierarchy (paths are all UIDs)
  df <- datim_pull_hierarchy(ouuid, username = myuser, password = mypwd(myuser))
## End(Not run)
```

---

datim_query *Query PEPFAR/DATIM targets/results data*

Description

Query PEPFAR/DATIM targets/results data

Usage

```
datim_query(
  ou,
  username,
  password,
  level = "prioritization",
  pe = "THIS_FINANCIAL_YEAR",
  ta = "PLHIV",
  value = NULL,
  disaggs = NULL,
  dimensions = NULL,
  property = "SHORTNAME",
  metadata = TRUE,
  hierarchy = TRUE,
  baseurl = "https://final.datim.org/",
  verbose = FALSE
)
```

Arguments

ou	Operatingunit
username	Datim username
password	Datim username
level	Organization hierarchy level
pe	Reporting period. This can be expressed as relative or fixed periods Eg.: "THIS_FINANCIAL_YEAR", "2020Oct", "QUARTERS_THIS_YEAR", "2021Q2" default is "THIS_FINANCIAL_YEAR"
ta	Technical Area, valid option can be obtain from glamr::datim_dim_items("Technical Area")‘
value	Type of value to return, MER Targets or Results or both
disaggs	Disaggregation Types. This depends on the value of ta
dimensions	Additional dimensions and/or columns. This depends on values of ta and disaggs
property	Type of name
metadata	Should metadata be included
hierarchy	Should additional hirarchy level be included
baseurl	DATIM API End point url
verbose	Display all notifications

Value

data as tibble

Examples

```
## Not run:
 library(grabr)

 datim_query(ou = "Nigeria", ta = "PLHIV")

 datim_query(ou = "Nigeria", ta = "POP_EST")

 datim_query(ou = "Nigeria",
             level = "country",
             ta = "TX_CURR",
             disaggs = "Age/Sex/HIVStatus",
             dimensions = c("Age: <15/15+  (Coarse)", "Sex"))

## End(Not run)
```

`datim_sqlviews` *Query Datim SQLViews*

Description

Query Datim SQLViews

Usage

```
datim_sqlviews(
  username,
  password,
  view_name = NULL,
  dataset = FALSE,
  datauid = NULL,
  query = NULL,
  baseurl = NULL
)
```

Arguments

`username`	Datim username
`password`	Datim password
`view_name`	Datim SQLView name
`dataset`	Return SQLView dataset or uid? Default is false
`datauid`	Data UID
`query`	SQLView Query params, a list containing type and params key value pairs
`baseurl`	Datim API Base URL

Value

SQLView uid or dataset as data frame

Note

This function should be used to identify Datim SQLView and Extract Data

Examples

```
## Not run:
  library(grabr)

  datim_sqlviews(
    username =glamr::datim_user(),
    password =glamr::datim_pwd(),
    view_name = "A list of OUs",
```

```
    dataset = TRUE
  )

## End(Not run)
```

---

gen_url *Generate a API URL*

Description

Generate a API URL

Usage

```
gen_url(
  ou_uid,
  org_lvl,
  org_type = "facility",
  value_type = "results",
  is_hts = FALSE,
  fy_pd = NULL,
  baseurl = "https://final.datim.org/"
)
```

Arguments

ou_uid	UID for the country, recommend using 'identify_ouuids()'
org_lvl	org hierarchy level, eg facility is level 7 in country X, recommend using 'identify_levels()'
org_type	organization type, either facility (default) or community
value_type	results (default) or targets
is_hts	is the API for HTS indicators (HTS_TST or HTS_TST_POS), default = FALSE
fy_pd	fiscal year(s) to cover, default will be current FY if not provided
baseurl	API base url, default = https://final.datim.org/

Examples

```
## Not run:
 #get OU UID
  ouuid <- identify_ouuids() %>% dplyr::filter(ou == "Ghana")
 #get facility level
  faclvl <- identify_levels("facility",
                            username = myuser, password = mypwd()) %>%
                            dplyr::filter(ou == "Ghana")
 #gen url
  myurl <- gen_url(ouuid, faclvl, org_type = facility)
## End(Not run)
```

get_baseurl *Get base url from a link*

Description

Get base url from a link

Usage

```
get_baseurl(url)
```

Arguments

url	DATIM API end points

Value

Base url without trailing slash

See Also

Other utility: var_exists()

get_datim_data *DATIM API Call for Targets*

Description

DATIM API Call for Targets

Usage

```
get_datim_data(url, username, password)
```

Arguments

url	supply url forAPI call, recommend using ‘gen_url()‘
username	DATIM username
password	DATIM password, recommend using ‘mypwd()‘

Examples

```
## Not run:
 myurl <- paste0(baseurl, "api/29/analytics.json?
                 dimension=LxhLO68FcXm:udCop657yzi&
                 dimension=ou:LEVEL-4;HfVjCurKxh2&
                 filter=pe:2018Oct&
                 displayProperty=SHORTNAME&outputIdScheme=CODE")
 myuser <- "UserX"
 df_datim <- get_datim_data(myurl, myuser, mypwd(myuser))
## End(Not run)
```

`get_datim_targets` *DATIM API Call for Targets*

Description

DATIM API Call for Targets

Usage

```
get_datim_targets(url, username, password)
```

Arguments

`url`	supply url forAPI call, recommend using‘gen_url()‘
`username`	DATIM username
`password`	DATIM password, recommend using ‘mypwd()‘

Examples

```
## Not run:
 myurl <- paste0(baseurl, "api/29/analytics.json?
                 dimension=LxhLO68FcXm:udCop657yzi&
                 dimension=ou:LEVEL-4;HfVjCurKxh2&
                 filter=pe:2018Oct&
                 displayProperty=SHORTNAME&outputIdScheme=CODE")
 myuser <- "UserX"
 df_targets <- get_datim_targets(myurl, myuser, mypwd(myuser))
## End(Not run)
```

`get_levels` *Get all orgunits levels in org hierarchy*

Description

Get all orgunits levels in org hierarchy

Usage

```
get_levels(
  username,
  password,
  expand = FALSE,
  reshape = FALSE,
  baseurl = "https://final.datim.org/"
)
```

Arguments

`username`	DATIM username, recommed using glamr::datim_user()‘
`password`	DATIM password, recommend using glamr::datim_pwd()‘
`expand`	Fill in missing snu1 level? default is FALSE
`reshape`	Reshape data as long? default is FALSE
`baseurl`	base API url, default = https://final.datim.org/

Value

df

Note

Similar to ‘grabr::identify_levels()‘ and ‘grabr::get_outable()‘

Examples

```
## Not run:
  library(grabr)

  # Get PEPFAR Org Levels
  get_levels()

## End(Not run)
```

`get_orguids` *Get Org UIDS*

Description

Get Org UIDS

Usage

```
get_orguids(
  level = 3,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

`level`	Org level
`username`	DATIM Username, recommend using glamr::datim_user()‘
`password`	DATIM password, recommend using glamr::datim_pwd()‘
`baseurl`	base url for the API, default = https://final.datim.org/

Value

ORG UIDS as tibble

Note

Use with caution. Use ‘get_ouorguids()‘ for levels below 3

Examples

```
## Not run:
 library(grabr)

 # All orgunit level 3 uids + names
 orgs <- get_orguids(level = 3)

## End(Not run)
```

`get_ouorglabel` *Identify OU/Org Label*

Description

Identify OU/Org Label

Usage

```
get_ouorglabel(
  operatingunit,
  country = NULL,
  org_level = 4,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

`operatingunit`	Operating unit
`country`	Country name
`org_level`	OU Org level, default is set to 4, PSNU
`username`	Datim account username
`password`	Datim account password
`baseurl`	Datim base url

Value

Org level label

Examples

```
## Not run:
  library(grabr)

  get_ouorglabel(operatingunit = "Zambia", org_level = 5)

## End(Not run)
```

`get_ouorglevel` *Get OU Org level*

Description

Get OU Org level

Usage

```
get_ouorglevel(
  operatingunit,
  country = NULL,
  org_type = "prioritization",
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

`operatingunit`	Operatingunit name
`country`	Country name (default = Operatingunit)
`org_type`	Orgunit type (country_lvl, prioritization, community, facility_lvl)
`username`	Datim Account username
`password`	Datim Account Password
`baseurl`	Datim Base URL

Value

uid

Examples

```
## Not run:
 library(grabr)

 cntry <- "Zambia"

 # Get country org level
 get_ouorglevel(cntry)

 # Get community org level
 get_ouorglevel(cntry, org_type = "community")

## End(Not run)
```

get_ouorgs *Get list of OU Orgs at specific level*

Description

Get list of OU Orgs at specific level

Usage

```
get_ouorgs(
  ouuid,
  level = 4,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

ouuid	OU uid
level	org level
username	DATIM Username
password	DATIM password, recommend using 'mypwd()'
baseurl	base url for the API, default = https://final.datim.org/

Value

ORG UIDS as tibble

Note

Use 'get_orguids()' for levels above 4

Examples

```
## Not run:

 library(grabr)

 cntry <- "Zambia"

 uid <- get_ouuid(cntry)

 lvl <- get_ouorglevel(cntry, org_type = "prioritization")

 orgs <- get_ouorgs(ouuid = uid, level = lvl)

## End(Not run)
```

get_ouorguids *Get Orgs uids by level*

Description

Get Orgs uids by level

Usage

```
get_ouorguids(
  ouuid,
  level,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

ouuid	Operatingunit uid
level	Orgunit level
username	Datim Account username
password	Datim Account Password
baseurl	Datim base url

Value

list of uids

Examples

```
## Not run:
 library(grabr)

 # Set country of interest
 cntry <- "Zambia"

 # Get OU/Country orgunit uid
 uid <- get_ouuid(cntry)

 # Get org level for psnu
 lvl <- get_ouorglevel(cntry, org_type = "prioritization")

 # Retreived all uids for level 4 (SNU1)
 get_ouorguids(ouuid = uid, level = 4)

## End(Not run)
```

`get_outable` *Pull Table of OUs/Countries, UIDs, ISO codes and levels*

Description

'get_outable' pulls from DATIM to return a dataframe with all PEPFAR Operating Units and countries along with useful information for merging, eg ISO codes, and use in DATIM APIs, eg UIDs and hierarchy levels.

Usage

```
get_outable(username, password, baseurl = "https://final.datim.org/")
```

Arguments

`username`	DATIM Username, defaults to using glamr::datim_user()' if blank
`password`	DATIM password, defaults to using glamr::datim_pwd()' if blank
`baseurl`	base url for the API, default = https://final.datim.org/

Details

'get_outtable' is a wrapper around 'identify_ouuids' and 'identify_levels' that pulls this information directly from DATIM. The user will need to have a DATIM account to access this data. You can take advantage of storing you credentials locally in a secure way using 'set_datim'.

Value

data frame with all PEPFAR OUs, countries, their UIDs, ISO codes and different levels in the DATIM hierarchy

See Also

[set_datim()] to store DATIM authentication; [load_secrets()] to load credentials into session

Examples

```
## Not run:
 load_secrets()
 ou_table <- datim_outable()
## End(Not run)
```

get_ouuid *Get Operatingunit / Country Org UID*

Description

Get Operatingunit / Country Org UID

Usage

```
get_ouuid(
  operatingunit,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

operatingunit	Operatingunit name
username	Datim Account username, recommend using glamr::datim_user()‘
password	Datim Account Password, recommend using glamr::datim_pwd()‘
baseurl	base url for the API, default = https://final.datim.org/

Value

uid

Examples

```
## Not run:
  library(grabr)

  # get orgunit for specific OU/Country: kenya
  get_ouuid(operatingunit = "Kenya")

## End(Not run)
```

get_ouuids *Get OU Org UIDS*

Description

Get OU Org UIDS

Usage

```
get_ouuids(
  add_details = FALSE,
  username,
  password,
  baseurl = "https://final.datim.org/"
)
```

Arguments

add_details	Add countries for regional ou, default is false
username	DATIM Username, recommend using glamr::datim_user()‘
password	DATIM password, recommend using glamr::datim_pwd()‘
baseurl	base url for the API, default = https://final.datim.org/

Value

OU UIDS as tibble

Examples

```
## Not run:
 library(grabr)

 # OU Org UIDs
 ous <- get_ouuids()

## End(Not run)
```

identify_levels *Identify Facility/Community levels in org hierarchy*

Description

'identify_levels' pulls from DATIM to return a dataframe with all PEPFAR Operating Units and countries with their ISO codes and hierarhcy levels. This is one of two components that feeds into 'get_outable'.

Usage

```
identify_levels(username, password, baseurl = "https://final.datim.org/")
```

Arguments

username	DATIM Username, defaults to using glamr::datim_user()' if blank
password	DATIM password, defaults to using glamr::datim_pwd()' if blank
baseurl	base url for the API, default = https://final.datim.org/

Details

To access the UIDs, the user will need to have a DATIM account. You can take advantage of storing you credentials locally in a secure way using 'set_datim'.

See Also

[set_datim()] to store DATIM authentication; [load_secrets()] to load credentials into session

Examples

```
## Not run:
 #table for all OUs
  load_secrets()
  identify_levels()
## End(Not run)
```

identify_ouuids *Pull OU UIDS*

Description

'identify_ouuids' pulls from DATIM to return a dataframe with all PEPFAR Operating Units and countries and their UIDs. This is one of two components that feeds into 'get_outable'.

Usage

```
identify_ouuids(username, password, baseurl = "https://final.datim.org/")
```

Arguments

username	DATIM Username, defaults to using glamr::datim_user()‘ if blank
password	DATIM password, defaults to using glamr::datim_pwd()‘ if blank
baseurl	base url for the API, default = https://final.datim.org/

Details

To access the UIDs, the user will need to have a DATIM account. You can take advantage of storing you credentials locally in a secure way using ‘set_datim‘.

Value

Datim country names

See Also

[set_datim()] to store DATIM authentication; [load_secrets()] to load credentials into session

Examples

```
## Not run:
 load_secrets()
 ous <- identify_ouuids()
## End(Not run)
```

---

lazy_secrets *Lazy checking/loading of credentials*

Description

This function is useful within another function. It check whether a username or password has been provided by the user and, if not, checks if they were stored via ‘glamr‘ or prompts user to provide credentials through interactive prompt.

Usage

```
lazy_secrets(service = c("datim", "pano", "pdap", "s3"), username, password)
```

Arguments

service	account, either datim", "pano", "pdap", or "s3"
username	account username or s3 access key
password	account password or s3 secret key

Value

returns a list of 2 - username/access and password/secret

Examples

```
## Not run:
  accnt <- lazy_secrets("datim", username = username, password = password)
  datim_dimensions(accnt$username, accnt$password)

## End(Not run)
```

`pano_download` *Download file from PEPFAR Panorama*

Description

Download file from PEPFAR Panorama

Usage

```
pano_download(
  item_url,
  username,
  password,
  session = NULL,
  dest_path = NULL,
  uncompress = FALSE
)
```

Arguments

`item_url`	URL for the item to be downlaoded
`username`	Username for PEPFAR Panorama Account. Recommend using ‘pano_user()‘
`password`	Password for PEPFAR Panorama Account. Recommend using ‘pano_pwd()‘
`session`	Login session, only used within other ‘pano_extract_*()‘
`dest_path`	Location and name of the destination file
`uncompress`	If yes, the downloaded zip file will be decompressed. Default is FALSE

Value

file content as binary

Examples

```
## Not run:
  library(tidyverse)
  library(grabr)
  library(glamr)

  url <- "https://pepfar-panorama.org/forms/downloads"

  elts <- pano_items(page_html = url,
                     username = pano_user(), password = pano_pwd())

  f_url <- elts %>% filter(type == "file zipfile") %>% pull(path) %>% first()

  pano_download(item_url = url, session = s, dest = "Data")
## End(Not run)
```

---

pano_extract *Extract data outputs from Panorama*

Description

Extract data outputs from Panorama

Usage

```
pano_extract(
  item = "mer",
  version,
  fiscal_year,
  quarter,
  unpack = FALSE,
  username,
  password,
  session = NULL,
  baseurl = "https://pepfar-panorama.org"
)
```

Arguments

item	Panorama data type. Eg: mer, financial, sims, narratives
version	Data release version: Initial or Clean, defaults to current version if blank
fiscal_year	Reporting Fiscal year, defaults to current FY if blank
quarter	Reporting Quarter, defaults to current quarter if blank
unpack	If TRUE, unpack nested directories
username	Panorama username, recommend using ‘glamr::pano_user()‘

password	Panorama password, recommend using ‘glamr::pano_pwd()‘
session	Login session, only used within other ‘pano_extract_*()‘
baseurl	Panorama base url

Value

list of output files as data frame

Note

This function combines ‘pano_session()‘, ‘pano_content()‘, ‘pano_elements()‘, and in some cases ‘pano_unpack()‘

Examples

```
## Not run:
  library(tidyverse)
  library(grabr)

  pano_extract(item = "mer")

## End(Not run)
```

---

pano_extract_msd *Downloads Country Specific MSDs*

Description

Downloads Country Specific MSDs

Usage

```
pano_extract_msd(
  operatingunit = NULL,
  version,
  fiscal_year,
  quarter,
  level = c("psnu", "ou", "site", "nat"),
  dest_path,
  username,
  password,
  baseurl = "https://pepfar-panorama.org"
)
```

Arguments

operatingunit	PEPFAR Operating Unit. Default is set to NULL for to return global datasets
version	Data release version: "initial" or "clean", defaults to current version
fiscal_year	Reporting Fiscal year, defaults to current version
quarter	Reporting Quarter: Single digit quarters, defaults to current version
level	Org level, options are psnu" (default), "ou", "site", or "nat"
dest_path	Directory path to download file. Default set to ‘si_path()‘
username	Panorama username, recommend using ‘glamr::pano_user()‘, which is the default if left blank
password	Panorama password, recommend using ‘glamr::pano_pwd()‘, which is the default if left blank
baseurl	Pano base url

Examples

```
## Not run:
 pano_extract_msd(operatingunit = "Zambia", level = "site")

## End(Not run)
```

---

pano_extract_msds *Downloads All Global + OU Specific MSDs*

Description

Downloads All Global + OU Specific MSDs

Usage

```
pano_extract_msds(
  operatingunit,
  add_global = TRUE,
  items = "mer",
  archive = FALSE,
  dest_path,
  username,
  password,
  baseurl = "https://pepfar-panorama.org"
)
```

Arguments

operatingunit	PEPFAR Operating Unit. Default is set to NULL for global datasets
add_global	Add global datasets in this extract? Default is TRUE
items	Panorama data set, default option is ‘mer‘
archive	Logical, should the old files be archived? default=FALSE
dest_path	Directory path to download file. Default set to ‘glamr::si_path()‘
username	Panorama username, recommend using ‘glamr::pano_user()‘
password	Panorama password, recommend using ‘glamr::pano_pwd()‘
baseurl	Panorama base url, default="https://pepfar-panorama.org"

Examples

```
## Not run:
 dir_mer <- si_path()

 pano_extract_msds(operatingunit = "Zambia",
                   archive = TRUE,
                   dest_path = dir_mer)

## End(Not run)
```

---

pano_items *Extract data items from url*

Description

Extract data items from url

Usage

```
pano_items(page_url, username, password, session = NULL)
```

Arguments

page_url	Current html page url
username	Username for PEPFAR Panorama Account. Recommend using ‘pano_user()‘
password	Password for PEPFAR Panorama Account. Recommend using ‘pano_pwd()‘
session	Login session, only used within other ‘pano_extract_*()‘

Value

data items as data frame

Examples

```
## Not run:
  library(grabr)

  s <- pano_session("<my-pano-user>", "<my-password>")
  url <- "https://pepfar-panorama.org/forms/downloads"

  items <- pano_items(page_url = url, session = s)

## End(Not run)
```

---

pano_session *Create an active session for PEPFAR Panorama*

Description

Create an active session for PEPFAR Panorama

Usage

```
pano_session(username, password, baseurl = "https://pepfar-panorama.org")
```

Arguments

username	Username for PEPFAR Panorama Account. Recommend using ‘pano_user()‘
password	Password for PEPFAR Panorama Account. Recommend using ‘pano_pwd()‘
baseurl	PEPFAR Panorama base url

Value

login session

Examples

```
## Not run:
  library(grabr)

  s <- pano_session("<my-pano-user>", "<my-password>")

## End(Not run)
```

pull_mech *Pull Partner/Mechanism Info from DATIM*

Description

Pull Partner/Mechanism Info from DATIM

Usage

```
pull_mech(usaid_only = TRUE, ou_sel = NULL, folderpath_output = NULL)
```

Arguments

usaid_only	specify if only USAID mechansism should be returned, default = TRUE
ou_sel	option to specify an operating unit, default = NULL
folderpath_output	provide the full path to the folder for saving

Examples

```
## Not run:
#pull mechanism/partner information
df <- pull_mech()
## End(Not run)
```

pull_mer *Extract DATIM Results and Targets (DATIM API Call)*

Description

Extract DATIM Results and Targets (DATIM API Call)

Usage

```
pull_mer(
  ou_name = NULL,
  username,
  password,
  baseurl = "https://final.datim.org/",
  fy_pd = NULL,
  quarters_complete = NULL,
  folderpath_output = NULL
)
```

Arguments

ou_name	Operating Unit name, if mechanism is not specified
username	DATIM username
password	DATIM password, recommend using ‘mypwd()‘
baseurl	API base url, default = https://final.datim.org/
fy_pd	fiscal year(s) to cover, default will be current FY if not provided
quarters_complete	no. of quarters completed through FY to determine weeks left in year
folderpath_output	folder path to store DATIM output, default = NULL

Examples

```
## Not run:
 #ou mer data
 myuser <- "UserX"
 mech_x_dta <- pull_mer(ou_name = "Namibia", username = myuser, password = mypwd(myuser))

## End(Not run)
```

---

s3_buckets *Get S3 Buckets list*

Description

Get S3 Buckets list

Usage

```
s3_buckets(access_key, secret_key, ...)
```

Arguments

access_key	S3 Access Key ID
secret_key	S3 Secret Access Key
...	Additional arguments passed to s3HTTP

Value

S3 Backets list as tibble

Examples

```
## Not run:
s3_buckets()
## End(Not run)
```

s3_download *Download S3 Objects*

Description

Download S3 Objects

Usage

```
s3_download(bucket, object, filepath = NULL, access_key, secret_key, ...)
```

Arguments

bucket	S3 Bucket name
object	S3 Object key (id)
filepath	Full path of destination file
access_key	S3 Access key id
secret_key	S3 Secret Access key
...	Additional aws.S3::save_object() options

Value

file name

Examples

```
## Not run:
s3_objects("sample-bucket") %>%
  filter(str_detect(key, "^HFR")) %>%
  pull(key) %>%
  first() %>%
  s3_download()
## End(Not run)
```

s3_excel_sheets *Read sheets from S3 Objects / Excel*

Description

Read sheets from S3 Objects / Excel

Usage

```
s3_excel_sheets(bucket, object_key, access_key, secret_key)
```

Arguments

bucket	S3 Bucket
object_key	S3 Object Key
access_key	S3 Access Key
secret_key	S3 Secret Key

Value

Excel sheets as data frame

Examples

```
## Not run:
s3_objects(
    bucket = "sample-bucket",
    prefix = "ddc/xyz/ABC") %>%
  filter(str_detect(key, "^HFR")) %>%
  pull(key) %>%
  first() %>%
  s3_excel_sheets(bucket = "<sample-bucket>", object_key = .)
## End(Not run)
```

---

s3_objects *Get S3 Bucket objects list*

Description

Get S3 Bucket objects list

Usage

```
s3_objects(
  bucket,
  prefix = NULL,
  n = 1000,
  unpack_keys = FALSE,
  access_key,
  secret_key,
  ...
)
```

Arguments

bucket	S3 Bucket name
prefix	Limits response by key. Default set to NULL
n	Max number of record, default = 1000
unpack_keys	Separate key column, default is false
access_key	S3 Access Key ID
secret_key	S3 Secret Access Key
...	Additional aws.s3::get_bucket_df() options

Value

S3 Objects list as tibble

Examples

```
## Not run:
s3_objects("sample-bucket")
## End(Not run)
```

---

s3_object_type *Idenfity S3 Object type*

Description

Idenfity S3 Object type

Usage

```
s3_object_type(object)
```

Arguments

object	S3 Object key

Value

file type: text, csv, excel, json, python, shell

Examples

```
## Not run:
s3_objects("sample-bucket") %>%
  filter(str_detect(key, "^HFR")) %>%
  pull(key) %>%
  first() %>%
  s3_object_type()
## End(Not run)
```

s3_read_object *Read content of S3 Objects*

Description

Read content of S3 Objects

Usage

```
s3_read_object(bucket, object, sheet = NULL, access_key, secret_key, ...)
```

Arguments

bucket	S3 Bucket name
object	S3 Object key (id)
sheet	S3 Excel object sheet name / index
access_key	S3 Access key id
secret_key	S3 Secret Access key
...	Additional arguments passed to s3HTTP

Value

df

Examples

```
## Not run:
s3_objects("sample-bucket") %>%
  filter(str_detect(key, "^HFR")) %>%
  pull(key) %>%
  first() %>%
  s3_read_object(bucket = "sample-bucket", object_key = .)
## End(Not run)
```

s3_remove *Remove objects from S3 bucket*

Description

Remove objects from S3 bucket

Usage

```
s3_remove(objects, bucket, access_key, secret_key, ...)
```

Arguments

objects	S3 object keys (full path)
bucket	S3 backet name
access_key	S3 Access Key
secret_key	S3 Secret Key
...	Additional aws.S3::delete_object() options

Value

boolean

Examples

```
## Not run:
df_objects %>%
  pull(key) %>%
  first() %>%
  s3_remove(objects = .,
            bucket = "test-bkt")
## End(Not run)
```

---

s3_unpack_keys *Unpack Objects Key*

Description

Unpack Objects Key

Usage

```
s3_unpack_keys(df_objects, rmv_sysfiles = TRUE, rmv_hidden = TRUE)
```

Arguments

df_objects	S3 objects as df
rmv_sysfiles	Remove System object (logs, git folders, etc.)?
rmv_hidden	Remove Hidden objects (folders/files starting with dot) ?

Value

S3 Cleaned Objects list as tibble

Examples

```
## Not run:
s3_objects("sample-bucket") %>% s3_unpack_keys()
## End(Not run)
```

`s3_upload` *Upload file to S3 Bucket*

Description

Upload file to S3 Bucket

Usage

```
s3_upload(
  filepath,
  bucket,
  prefix = NULL,
  object = NULL,
  access_key,
  secret_key,
  ...
)
```

Arguments

`filepath`	Source file path
`bucket`	S3 backet name
`prefix`	S3 Prefix (folder structure). Default set to NULL
`object`	Destination S3 object name (with file extention)
`access_key`	S3 Access Key
`secret_key`	S3 Secret Key
`...`	Additional aws.S3::put_object() options

Value

boolean

Examples

```
## Not run:
filepath %>% s3_upload(bucket = "test-bkt")
## End(Not run)
```

var_exists *Check if variable exist*

Description

Check if variable exist

Usage

```
var_exists(df, var)
```

Arguments

df	data frame to check against
var	quoted variable of interest

See Also

Other utility: `get_baseurl()`

Examples

```
## Not run:
var_exists(df, "val")
## End(Not run)
```

wave_process_query *Pull Genie Data from PDAP Wave*

Description

This function provides streamlined API access to PDAP Wave data, the successor to DATIM Genie. PDAP Wave API simplifies the requests that previously needed to be made with DATIM and returns a dataset back that matches the MSD structure. Further documentation can be found at https://wave.test.pdap.pepfar.net/api/docs#/.

Usage

```
wave_process_query(
  request_body,
  folderpath_dwnld = "Data",
  psd_type = c("psnu_im", "ou_im", "site_im"),
  request_type = c("POST", "GET"),
  username,
  password
)
```

Arguments

request_body	elements to pass into the PDAP Wave POST API
folderpath_dwnld	where to download, default = "Data"
psd_type	Type of PEPFAR Structured dataset: "psnu_im" (default), "ou_im", or "site_im"
request_type	API request type: "POST" (default) or "GET
username	DATIM username, if blank looks for stored credentials (glamr::set_datim()) and then prompts for credentials if not found
password	DATIM password, if blank looks for stored credentials (glamr::set_datim()) and then prompts for credentials if not found

Details

Users must pass their query filter in a list form into request_body, which matches what you would manually do in Genie previously. You can proceed with either POST or GET requests to access PSNUxIM, OUxIM, and SitexIM data.

This function was adapted from code developed and shared by Derek Wood (GHSD/PRIME).

Value

list of request and stored data in zip

Examples

```
## Not run:
 library(tidyverse)
 library(glamr)

 #get country uid for API
cntry_uid <- pepfar_country_list %>%
 filter(country == "Tanzania") %>%
 pull(country_uid)

 #establish parameters to pass into POST API
 post_body <- list(
   daily_frozen='daily',
   fiscal_year=list(2023, 2024),
   funding_agency = list("USAID"),
   indicator=list("TX_CURR","TX_ML","TX_CURR_LAG2", "TX_NET_NEW","TX_NEW",
                  "TX_RTT","PMTCT_STAT", "PMTCT_STAT_POS", "PMTCT_ART"),
   uid_hierarchy_list=list(str_glue('-|-|{cntry_uid}')))

 #run POST API
 wave_process_query(post_body)

 #load data
 df_wave <- return_latest("Data") %>%
    read_psd()

## End(Not run)
```

Index