

# Package ‘connectapi’

June 6, 2024

**Type** Package

**Title** Utilities for Interacting with the 'Posit Connect' Server API

**Version** 0.2.0

**Description** Provides a helpful 'R6' class and methods for interacting with the 'Posit Connect' Server API along with some meaningful utility functions for regular tasks. API documentation varies by 'Posit Connect' installation and version, but the latest documentation is also hosted publicly at <https://docs.posit.co/connect/api/>.

**License** MIT + file LICENSE

**URL** <https://pkgs.rstudio.com/connectapi/>, <https://github.com/rstudio/connectapi>

**BugReports** <https://github.com/rstudio/connectapi/issues>

**Imports** bit64,

fs,  
glue,  
httr,  
jsonlite,  
lifecycle,  
magrittr,  
purrr,  
R6,  
rlang (>= 0.4.2),  
tibble,  
uuid,  
vctrs (>= 0.3.0)

**Suggests** covr,

dbplyr,  
dplyr,  
ggplot2,  
gridExtra,  
httptest,  
knitr,  
lubridate,  
progress,  
rmarkdown,  
rprojroot,  
rsconnect,  
spelling,

testthat,  
 webshot2

**VignetteBuilder** knitr

**RdMacros** lifecycle

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.1

**Roxygen** list(markdown = TRUE)

**Config/testthat/edition** 3

## R topics documented:

connectapi-package . . . . .	3
audit_access_open . . . . .	4
audit_runas . . . . .	4
audit_r_versions . . . . .	5
browse_solo . . . . .	5
Bundle . . . . .	6
bundle_dir . . . . .	7
bundle_path . . . . .	7
bundle_static . . . . .	8
connect . . . . .	9
Content . . . . .	10
ContentTask . . . . .	15
content_delete . . . . .	16
content_item . . . . .	17
content_list_by_tag . . . . .	17
content_list_with_permissions . . . . .	18
content_title . . . . .	19
content_update . . . . .	19
create_random_name . . . . .	20
dashboard_url . . . . .	21
dashboard_url_chr . . . . .	21
delete_vanity_url . . . . .	22
deploy . . . . .	22
deploy_repo . . . . .	24
download_bundle . . . . .	25
Environment . . . . .	26
get_audit_logs . . . . .	27
get_bundles . . . . .	28
get_content . . . . .	29
get_environment . . . . .	31
get_groups . . . . .	32
get_group_members . . . . .	33
get_image . . . . .	34
get_jobs . . . . .	35
get_procs . . . . .	35
get_tags . . . . .	36
get_timezones . . . . .	37

get_usage_shiny . . . . .	38
get_usage_static . . . . .	39
get_users . . . . .	41
get_vanity_url . . . . .	42
get_variants . . . . .	43
get_variant_renderings . . . . .	43
get_variant_schedule . . . . .	44
git . . . . .	44
groups_create_remote . . . . .	45
page_cursor . . . . .	46
permissions . . . . .	46
poll_task . . . . .	47
PositConnect . . . . .	48
promote . . . . .	60
set_image_path . . . . .	61
set_run_as . . . . .	62
set_schedule . . . . .	63
set_vanity_url . . . . .	65
swap_vanity_url . . . . .	66
Task . . . . .	67
tbl_connect . . . . .	68
users_create_remote . . . . .	69
user_guid_from_username . . . . .	69
Vanity . . . . .	70
vanity_is_available . . . . .	71
Variant . . . . .	72
VariantSchedule . . . . .	75
VariantTask . . . . .	77
verify_content_name . . . . .	78

<b>Index</b>	<b>80</b>
--------------	-----------

---

connectapi-package	<i>connectapi: Utilities for Interacting with the 'Posit Connect' Server API</i>
--------------------	--

---

## Description

Provides a helpful 'R6' class and methods for interacting with the 'Posit Connect' Server API along with some meaningful utility functions for regular tasks. API documentation varies by 'Posit Connect' installation and version, but the latest documentation is also hosted publicly at <https://docs.posit.co/connect/api/>.

## Author(s)

**Maintainer:** Cole Arendt <cole@posit.co>

Authors:

- Sean Lopp

Other contributors:

- Posit, PBC [copyright holder, funder]

**See Also**

Useful links:

- <https://pkgs.rstudio.com/connectapi/>
- <https://github.com/rstudio/connectapi>
- Report bugs at <https://github.com/rstudio/connectapi/issues>

---

audit_access_open	<i>Audit Access Controls</i>
-------------------	------------------------------

---

**Description**

**[Experimental]**

**Usage**

```
audit_access_open(content, type = "all")
```

**Arguments**

content	data.frame of content information, as from <a href="#">get_content()</a>
type	One of "all" or "logged_in". If "all", return a list of apps whose access control is set to "Everyone". If "logged_in", return a list of apps whose access control is set to "All logged in users"

**See Also**

Other audit functions: [audit\\_r\\_versions\(\)](#), [audit\\_runas\(\)](#), [vanity\\_is\\_available\(\)](#)

---

audit_runas	<i>Audit Run As Settings</i>
-------------	------------------------------

---

**Description**

**[Experimental]**

**Usage**

```
audit_runas(content)
```

**Arguments**

content	data.frame of content information, as from <a href="#">get_content()</a>
---------	--

**Value**

A data frame with the app name and the Run As user if the Run As user is not the default

**See Also**

Other audit functions: [audit\\_access\\_open\(\)](#), [audit\\_r\\_versions\(\)](#), [vanity\\_is\\_available\(\)](#)

---

audit_r_versions	<i>Audit R Versions</i>
------------------	-------------------------

---

## Description

[Experimental]

## Usage

```
audit_r_versions(content)
```

## Arguments

content            data.frame of content information, as from [get\\_content\(\)](#)

## Value

A plot that shows the R version used by content over time and in aggregate.

## See Also

Other audit functions: [audit\\_access\\_open\(\)](#), [audit\\_runas\(\)](#), [vanity\\_is\\_available\(\)](#)

---

browse_solo	<i>Browse</i>
-------------	---------------

---

## Description

Browse to different locations on Connect via `utils::browseURL`

## Usage

```
browse_solo(content)
```

```
browse_dashboard(content)
```

```
browse_api_docs(connect)
```

```
browse_connect(connect)
```

## Arguments

content            A R6 Content object

connect            A R6 Connect object

## Value

The url that is opened in the browser

---

Bundle

*Bundle*

---

### Description

Bundle

Bundle

### Details

An R6 class that represents a bundle

### Public fields

path The bundle path on disk.

size The size of the bundle.

### Methods

#### Public methods:

- [Bundle\\$new\(\)](#)
- [Bundle\\$print\(\)](#)
- [Bundle\\$clone\(\)](#)

**Method** `new()`: Initialize this content bundle.

*Usage:*

```
Bundle$new(path)
```

*Arguments:*

path The bundle path on disk.

**Method** `print()`: Print this object.

*Usage:*

```
Bundle$print(...)
```

*Arguments:*

... Unused.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Bundle$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### See Also

Other R6 classes: [Content](#), [ContentTask](#), [Environment](#), [PositConnect](#), [Task](#), [Vanity](#), [Variant](#), [VariantSchedule](#), [VariantTask](#)

---

bundle_dir	<i>Define a bundle from a Directory</i>
------------	---

---

**Description**

Creates a bundle from a target directory.

**Usage**

```
bundle_dir(  
  path = ".",  
  filename = fs::file_temp(pattern = "bundle", ext = ".tar.gz")  
)
```

**Arguments**

path	The path to the directory to be bundled
filename	The output bundle path

**Value**

Bundle A bundle object

**See Also**

Other deployment functions: [bundle\\_path\(\)](#), [bundle\\_static\(\)](#), [deploy\(\)](#), [download\\_bundle\(\)](#), [poll\\_task\(\)](#)

**Examples**

```
bundle_dir(system.file("tests/testthat/examples/shiny/", package = "connectapi"))
```

---

bundle_path	<i>Define a bundle from a path (a path directly to a tar.gz file)</i>
-------------	---

---

**Description**

Define a bundle from a path (a path directly to a tar.gz file)

**Usage**

```
bundle_path(path)
```

**Arguments**

path	The path to a .tar.gz file
------	----------------------------

**Value**

Bundle A bundle object

**See Also**

Other deployment functions: [bundle\\_dir\(\)](#), [bundle\\_static\(\)](#), [deploy\(\)](#), [download\\_bundle\(\)](#), [poll\\_task\(\)](#)

**Examples**

```
bundle_path(system.file("tests/testthat/examples/static.tar.gz", package = "connectapi"))
```

---

bundle_static	<i>Define a bundle from a static file (or files)</i>
---------------	--

---

**Description**

Defines a bundle from static files. It copies all files to a temporary directory, generates a basic manifest file (using the first file as the "primary"), and bundles the directory.

**Usage**

```
bundle_static(
  path,
  filename = fs::file_temp(pattern = "bundle", ext = ".tar.gz")
)
```

**Arguments**

path	The path to a file (or files) that will be used for the static bundle
filename	The output bundle path

**Details**

NOTE: the `rsconnect` package is required for this function to work properly.

**Value**

Bundle A bundle object

**See Also**

Other deployment functions: [bundle\\_dir\(\)](#), [bundle\\_path\(\)](#), [deploy\(\)](#), [download\\_bundle\(\)](#), [poll\\_task\(\)](#)

**Examples**

```
bundle_static(system.file("logo.png", package = "connectapi"))
```



---

`connect`*Create a connection to Posit Connect*

---

### Description

Creates a connection to Posit Connect using the server URL and an api key. Validates the connection and checks that the version of the server is compatible with the current version of the package.

### Usage

```
connect(  
  server = Sys.getenv(paste0(prefix, "_SERVER"), NA_character_),  
  api_key = Sys.getenv(paste0(prefix, "_API_KEY"), NA_character_),  
  prefix = "CONNECT",  
  ...,  
  .check_is_fatal = TRUE  
)
```

### Arguments

<code>server</code>	The URL for accessing Posit Connect. Defaults to environment variable <code>CONNECT_SERVER</code>
<code>api_key</code>	The API Key to authenticate to Posit Connect with. Defaults to environment variable <code>CONNECT_API_KEY</code>
<code>prefix</code>	The prefix used to determine environment variables
<code>...</code>	Additional arguments. Not used at present
<code>.check_is_fatal</code>	Whether to fail if "check" requests fail. Useful in rare cases where more http request customization is needed for requests to succeed.

### Value

A Posit Connect R6 object that can be passed along to methods

### Examples

```
## Not run:  
connect()  
  
## End(Not run)  
  
# default is to read CONNECT_SERVER and CONNECT_API_KEY environment variables  
# this example will read TEST_1_SERVER and TEST_1_API_KEY  
connect(prefix = "TEST_1")
```

---

Content

*Content*

---

## Description

Content

Content

## Details

An R6 class that represents content

## Public fields

connect An R6 Connect object

content The content details from Posit Connect

## Methods

### Public methods:

- [Content\\$new\(\)](#)
- [Content\\$get\\_connect\(\)](#)
- [Content\\$get\\_content\(\)](#)
- [Content\\$get\\_content\\_remote\(\)](#)
- [Content\\$get\\_bundles\(\)](#)
- [Content\\$bundle\\_download\(\)](#)
- [Content\\$bundle\\_delete\(\)](#)
- [Content\\$internal\\_content\(\)](#)
- [Content\\$update\(\)](#)
- [Content\\$danger\\_delete\(\)](#)
- [Content\\$get\\_url\(\)](#)
- [Content\\$get\\_dashboard\\_url\(\)](#)
- [Content\\$jobs\(\)](#)
- [Content\\$job\(\)](#)
- [Content\\$variants\(\)](#)
- [Content\\$tag\\_set\(\)](#)
- [Content\\$tag\\_delete\(\)](#)
- [Content\\$tags\(\)](#)
- [Content\\$permissions\\_add\(\)](#)
- [Content\\$permissions\\_update\(\)](#)
- [Content\\$permissions\\_delete\(\)](#)
- [Content\\$permissions\(\)](#)
- [Content\\$environment\(\)](#)
- [Content\\$environment\\_set\(\)](#)
- [Content\\$environment\\_all\(\)](#)
- [Content\\$deploy\(\)](#)

- `Content$repo_enable()`
- `Content$repo_set()`
- `Content$print()`
- `Content$clone()`

**Method** `new()`: Initialize this content.

*Usage:*

```
Content$new(connect, content)
```

*Arguments:*

`connect` The Connect instance.

`content` The content data.

**Method** `get_connect()`: Returns the Connect instance.

*Usage:*

```
Content$get_connect()
```

**Method** `get_content()`: Returns the underlying content data.

*Usage:*

```
Content$get_content()
```

**Method** `get_content_remote()`: Obtain the content data from the Connect server.

*Usage:*

```
Content$get_content_remote()
```

**Method** `get_bundles()`: Return the set of content bundles.

*Usage:*

```
Content$get_bundles()
```

**Method** `bundle_download()`: Download the source archive for a content bundle.

*Usage:*

```
Content$bundle_download(  
  bundle_id,  
  filename = tempfile(pattern = "bundle", fileext = ".tar.gz"),  
  overwrite = FALSE  
)
```

*Arguments:*

`bundle_id` The bundle identifier.

`filename` Where to write the result.

`overwrite` Overwrite an existing filename.

**Method** `bundle_delete()`: Delete a content bundle.

*Usage:*

```
Content$bundle_delete(bundle_id)
```

*Arguments:*

`bundle_id` The bundle identifier.

**Method** `internal_content()`: Get this (remote) content item.

*Usage:*

`Content$internal_content()`

**Method** `update()`: Update this content item.

*Usage:*

`Content$update(...)`

*Arguments:*

... Content fields.

**Method** `danger_delete()`: Delete this content item.

*Usage:*

`Content$danger_delete()`

**Method** `get_url()`: Return the URL for this content.

*Usage:*

`Content$get_url()`

**Method** `get_dashboard_url()`: Return the URL for this content in the Posit Connect dashboard.

*Usage:*

`Content$get_dashboard_url(pane = "")`

*Arguments:*

pane The pane in the dashboard to link to.

**Method** `jobs()`: Return the jobs for this content.

*Usage:*

`Content$jobs()`

**Method** `job()`: Return a single job for this content.

*Usage:*

`Content$job(key)`

*Arguments:*

key The job key.

**Method** `variants()`: Return the variants for this content.

*Usage:*

`Content$variants()`

**Method** `tag_set()`: Set a tag for this content.

*Usage:*

`Content$tag_set(tag_id)`

*Arguments:*

tag\_id The tag identifier.

**Method** `tag_delete()`: Remove a tag for this content.

*Usage:*

`Content$tag_delete(tag_id)`

*Arguments:*

tag\_id The tag identifier.

**Method** tags(): The tags for this content.

*Usage:*

Content\$tags()

**Method** permissions\_add(): Add a principal to the ACL for this content.

*Usage:*

Content\$permissions\_add(principal\_guid, principal\_type, role)

*Arguments:*

principal\_guid GUID for the target user or group.

principal\_type Acting on user or group.

role The kind of content access.

**Method** permissions\_update(): Alter a principal in the ACL for this content.

*Usage:*

Content\$permissions\_update(id, principal\_guid, principal\_type, role)

*Arguments:*

id The target identifier.

principal\_guid GUID for the target user or group.

principal\_type Acting on user or group.

role The kind of content access.

**Method** permissions\_delete(): Remove an entry from the ACL for this content.

*Usage:*

Content\$permissions\_delete(id)

*Arguments:*

id The target identifier.

**Method** permissions(): Obtain some or all of the ACL for this content.

*Usage:*

Content\$permissions(id = NULL, add\_owner = FALSE)

*Arguments:*

id The target identifier.

add\_owner Include the content owner in the result set.

**Method** environment(): Return the environment variables set for this content.

*Usage:*

Content\$environment()

**Method** environment\_set(): Adjust the environment variables set for this content.

*Usage:*

Content\$environment\_set(...)

*Arguments:*

... Environment variable names and values.

**Method** environment\_all(): Overwrite the environment variables set for this content.

*Usage:*

```
Content$environment_all(...)
```

*Arguments:*

... Environment variable names and values.

**Method** `deploy()`: Deploy this content

*Usage:*

```
Content$deploy(bundle_id = NULL)
```

*Arguments:*

`bundle_id` Target bundle identifier.

**Method** `repo_enable()`: Adjust Git polling.

*Usage:*

```
Content$repo_enable(enabled = TRUE)
```

*Arguments:*

`enabled` Polling enabled.

**Method** `repo_set()`: Adjust Git repository.

*Usage:*

```
Content$repo_set(repository, branch, subdirectory)
```

*Arguments:*

`repository` Git repository URL

`branch` Git repository branch

`subdirectory` Git repository directory

**Method** `print()`: Print this object.

*Usage:*

```
Content$print(...)
```

*Arguments:*

... Unused.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Content$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**See Also**

Other R6 classes: [Bundle](#), [ContentTask](#), [Environment](#), [PositConnect](#), [Task](#), [Vanity](#), [Variant](#), [VariantSchedule](#), [VariantTask](#)

---

ContentTask

*ContentTask*

---

## Description

ContentTask

ContentTask

## Details

An R6 class that represents a Task for a piece of Content

## Super class

[connectapi::Content](#) -> ContentTask

## Public fields

task The task.

data The task data.

## Methods

### Public methods:

- [ContentTask\\$new\(\)](#)
- [ContentTask\\$get\\_task\(\)](#)
- [ContentTask\\$add\\_data\(\)](#)
- [ContentTask\\$get\\_data\(\)](#)
- [ContentTask\\$print\(\)](#)
- [ContentTask\\$clone\(\)](#)

**Method** `new()`: Initialize this task.

*Usage:*

```
ContentTask$new(connect, content, task)
```

*Arguments:*

connect The Connect instance.

content The Content instance.

task The task data.

**Method** `get_task()`: Return the underlying task.

*Usage:*

```
ContentTask$get_task()
```

**Method** `add_data()`: Set the data.

*Usage:*

```
ContentTask$add_data(data)
```

*Arguments:*

data The data.

**Method** `get_data()`: Get the data.

*Usage:*

`ContentTask$get_data()`

**Method** `print()`: Print this object.

*Usage:*

`ContentTask$print(...)`

*Arguments:*

... Unused.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

`ContentTask$clone(deep = FALSE)`

*Arguments:*

deep Whether to make a deep clone.

### See Also

Other R6 classes: [Bundle](#), [Content](#), [Environment](#), [PositConnect](#), [Task](#), [Vanity](#), [Variant](#), [VariantSchedule](#), [VariantTask](#)

---

content\_delete

*Delete Content*

---

### Description

Delete a content item. **WARNING:** This action deletes all history, configuration, logs, and resources about a content item. It *cannot* be undone.

### Usage

```
content_delete(content, force = FALSE)
```

### Arguments

content	an R6 content item
force	Optional. A boolean that determines whether we should prompt in interactive sessions

### Value

The R6 Content item. The item is deleted, but information about it is cached locally

### See Also

Other content functions: [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)



---

content_item	<i>Get Content Item</i>
--------------	-------------------------

---

**Description**

Returns a single content item based on guid

**Usage**

```
content_item(connect, guid)
```

**Arguments**

connect	A Connect object
guid	The GUID for the content item to be retrieved

**Value**

A Content object for use with other content endpoints

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git\\_permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

**Examples**

```
## Not run:
connect() %>%
  content_item("some-guid") %>%
  content_update_access_type("all")

## End(Not run)
```

---

content_list_by_tag	<i>Content List</i>
---------------------	---------------------

---

**Description**

**[Experimental]** Get a content list

**Usage**

```
content_list_by_tag(src, tag)
```

**Arguments**

src	An R6 Connect object
tag	A connect_tag_tree object or tag ID

**Details**

content\_list\_by\_tag() retrieves a content list by tag

---

content\_list\_with\_permissions

*Get Content List with Permissions*

---

**Description**

**[Experimental]** These functions are experimental placeholders until the API supports this behavior.

**Usage**

content\_list\_with\_permissions(src, ..., .p = NULL)

content\_list\_guid\_has\_access(content\_list, guid)

**Arguments**

src	A Connect R6 object
...	Extra arguments. Currently not used
.p	Optional. A predicate function, passed as-is to purrr::keep(). See get_content() for more details. Can greatly help performance by reducing how many items to get permissions for
content_list	A "content list with permissions" as returned by content_list_with_permissions()
guid	A user or group GUID to filter the content list by whether they have access

**Details**

content\_list\_with\_permissions loops through content and retrieves permissions for each item (with a progress bar). This can take a long time for lots of content! Make sure to use the optional .p argument as a predicate function that filters the content list before it is transformed.

content\_list\_guid\_has\_access works with a content\_list\_with\_permissions dataset by checking whether a given GUID (either user or group) has access to the content by:

- checking if the content has access\_type == "all"
- checking if the content has access\_type == "logged\_in"
- checking if the provided guid is the content owner
- checking if the provided guid is in the list of content permissions (in the "permissions" column)

---

content_title	<i>Get Content Title</i>
---------------	--------------------------

---

### Description

Return content title for a piece of content. If the content is missing (deleted) or not visible, then returns the default

### Usage

```
content_title(connect, guid, default = "Unknown Content")
```

### Arguments

connect	A Connect object
guid	The GUID for the content item to be retrieved
default	The default value returned for missing or not visible content

### Value

character. The title of the requested content

### See Also

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

content_update	<i>Update Content</i>
----------------	-----------------------

---

### Description

Update settings for a content item. For a list of all settings, see the [latest documentation](#) or the documentation for your server via `connectapi::browse_api_docs()`.

### Usage

```
content_update(content, ...)

content_update_access_type(content, access_type = c("all", "logged_in", "acl"))

content_update_owner(content, owner_guid)
```

**Arguments**

content	An R6 content item
...	Settings up update that are passed along to Posit Connect
access_type	One of "all", "logged_in", or "acl"
owner_guid	The GUID of a user who is a publisher, so that they can become the new owner of the content

**Details**

Popular selections are `content_update(access_type="all")`, `content_update(access_type="logged_in")` or `content_update(access_type="acl")`, process settings, title, description, etc.

- `content_update_access_type()` is a helper to make it easier to change `access_type`
- `content_update_owner()` is a helper to make it easier to change owner

**Value**

An R6 content item

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git\\_permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

create\_random\_name      *Create Random Name*

---

**Description**

Creates a random name from the LETTERS dataset

**Usage**

```
create_random_name(length = 25)
```

**Arguments**

length                      Optional. The length of the random name. Defaults to 25

**Value**

The randomly generated name

**See Also**

`connectapi::verify_content_name`

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git\\_permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

dashboard_url	<i>Build a Dashboard URL from a Content Item</i>
---------------	--

---

**Description**

Returns the URL for the content dashboard (opened to the selected pane).

**Usage**

```
dashboard_url(content, pane = "")
```

**Arguments**

content	<a href="#">Content</a> A Content object
pane	character The pane in the dashboard to link to

**Value**

character The dashboard URL for the content provided

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git\\_permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

dashboard_url_chr	<i>Build a Dashboard URL from Character Vectors</i>
-------------------	---

---

**Description**

Returns the URL for the content dashboard (opened to the selected pane). NOTE: this takes a character object for performance optimization.

**Usage**

```
dashboard_url_chr(connect_url, content_guid, pane = "")
```

**Arguments**

connect_url	character The base URL of the Connect server
content_guid	character The guid for the content item in question
pane	character The pane in the dashboard to link to

**Value**

character The dashboard URL for the content provided

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

`delete_vanity_url`      *Delete the Vanity URL*

---

**Description**

Deletes the Vanity URL for a piece of content.

**Usage**

```
delete_vanity_url(content)
```

**Arguments**

`content`              A Content object

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

`deploy`                      *Deploy a bundle*

---

**Description**

Deploys a bundle (tarball) to an Posit Connect server. If not provided, `name` (a unique identifier) will be an auto-generated alphabetic string. If deploying to an existing endpoint, you can set `name` or `guid` to the desired content.

**Usage**

```
deploy(
  connect,
  bundle,
  name = create_random_name(),
  title = name,
  guid = NULL,
  ...,
  .pre_deploy = {
  }
)

deploy_current(content)
```

## Arguments

<code>connect</code>	A Connect object
<code>bundle</code>	A Bundle object
<code>name</code>	The unique name for the content on the server
<code>title</code>	optional The title to be used for the content on the server
<code>guid</code>	optional The GUID if the content already exists on the server
<code>...</code>	Additional arguments passed along to the content creation
<code>.pre_deploy</code>	An expression to execute before deploying the new bundle. The variables <code>content</code> and <code>bundle_id</code> are supplied
<code>content</code>	A Content object

## Details

This function accepts the same arguments as `connectapi::content_update()`.

`deploy_current()` is a helper to easily redeploy the currently active bundle for an existing content item.

## Value

Task A task object

## See Also

`connectapi::content_update`

Other deployment functions: [bundle\\_dir\(\)](#), [bundle\\_path\(\)](#), [bundle\\_static\(\)](#), [download\\_bundle\(\)](#), [poll\\_task\(\)](#)

## Examples

```
## Not run:
client <- connect()

# beware bundling big directories, like `renv/`, `data/`, etc.
bnd <- bundle_dir(".")

deploy(client, bnd)

## End(Not run)

client <- connect(prefix = "TEST_1")
bnd <- bundle_path(system.file("tests/testthat/examples/static.tar.gz", package = "connectapi"))
deploy(client, bnd)
```

---

 deploy\_repo

*Deploy a Git Repository*


---

### Description

**[Experimental]** Deploy a git repository directly to Posit Connect, using Posit Connect's "pull-based" "git-polling" method of deployment.

### Usage

```

deploy_repo(
  client,
  repository,
  branch,
  subdirectory,
  name = create_random_name(),
  title = name,
  ...
)

deploy_repo_enable(content, enabled = TRUE)

deploy_repo_update(content)

```

### Arguments

client	A Connect R6 object
repository	The git repository to deploy
branch	The git branch to deploy
subdirectory	The subdirectory to deploy (must contain a manifest.json)
name	The "name" / unique identifier for the content. Defaults to a random character string
title	The "title" of the content
...	Additional options for defining / specifying content attributes
content	An R6 Content object (i.e. the result of <code>content_item()</code> )
enabled	Whether Connect will enable automatic polling for repository updates

### Details

- `deploy_repo_enable()` enables (or disables) Posit Connect's git polling for a piece of content
- `deploy_repo_update()` triggers an update of the content from its git repository, if any are present

### Value

A ContentTask object, for use with `poll_task()` (if you want to follow the logs)



**See Also**

connectapi::poll\_task, connectapi::repo\_check\_branches, connectapi::repo\_check\_manifest\_dirs

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

download\_bundle

*Download a Bundle from Deployed Connect Content*

---

**Description**

Downloads a Content item's active bundle, or (optionally) one of its other bundles.

**Usage**

```
download_bundle(
  content,
  filename = fs::file_temp(pattern = "bundle", ext = ".tar.gz"),
  bundle_id = NULL,
  overwrite = FALSE
)
```

**Arguments**

content	A Content object
filename	Optional. The output bundle path
bundle_id	Optional. A string representing the bundle_id to download. If NULL, will use the currently active bundle.
overwrite	Optional. Default FALSE. Whether to overwrite the target location if it already exists

**Value**

Bundle A bundle object

**See Also**

Other deployment functions: [bundle\\_dir\(\)](#), [bundle\\_path\(\)](#), [bundle\\_static\(\)](#), [deploy\(\)](#), [poll\\_task\(\)](#)

---

 Environment

*Environment*


---

## Description

Environment

Environment

## Details

An R6 class that represents a Content's Environment Variables

## Super class

`connectapi::Content` -> Environment

## Public fields

`env_raw` The (raw) set of environment variables.

`env_vars` The set of environment variables.

## Methods

### Public methods:

- `Environment$new()`
- `Environment$environment()`
- `Environment$environment_set()`
- `Environment$environment_all()`
- `Environment$env_refresh()`
- `Environment$print()`
- `Environment$clone()`

**Method** `new()`: Initialize this set of environment variables.

*Usage:*

```
Environment$new(connect, content)
```

*Arguments:*

`connect` The Connect instance.

`content` The Content instance.

**Method** `environment()`: Fetch the set of environment variables.

*Usage:*

```
Environment$environment()
```

**Method** `environment_set()`: Update the set of environment variables.

*Usage:*

```
Environment$environment_set(...)
```

*Arguments:*

... Environment variable names and values.

**Method** `environment_all()`: Overwrite the set of environment variables.

*Usage:*

```
Environment$environment_all(...)
```

*Arguments:*

... Environment variable names and values.

**Method** `env_refresh()`: Fetch the set o environment variables.

*Usage:*

```
Environment$env_refresh()
```

**Method** `print()`: Print this object.

*Usage:*

```
Environment$print(...)
```

*Arguments:*

... Unused.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Environment$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### See Also

Other R6 classes: [Bundle](#), [Content](#), [ContentTask](#), [PositConnect](#), [Task](#), [Vanity](#), [Variant](#), [VariantSchedule](#), [VariantTask](#)

---

get\_audit\_logs

*Get Audit Logs from Posit Connect Server*

---

### Description

Get Audit Logs from Posit Connect Server

### Usage

```
get_audit_logs(src, limit = 500, previous = NULL, nxt = NULL, asc_order = TRUE)
```

### Arguments

src	The source object
limit	The number of records to return.
previous	Retrieve the previous page of Shiny application usage logs relative to the provided value. This value corresponds to an internal reference within the server and should be sourced from the appropriate attribute within the paging object of a previous response.

nxt	Retrieve the next page of Shiny application usage logs relative to the provided value. This value corresponds to an internal reference within the server and should be sourced from the appropriate attribute within the paging object of a previous response.
asc_order	Defaults to TRUE; Determines if the response records should be listed in ascending or descending order within the response. Ordering is by the started timestamp field.

### Details

Please see <https://docs.posit.co/connect/api/#getAuditLogs> for more information.

### Value

A tibble with the following columns:

- id: ID of the audit action
- time: Timestamp in RFC3339 format when action was taken
- user\_id: User ID of the actor who made the audit action
- user\_description: Description of the actor
- action: Audit action taken
- event\_description: Description of action

### Examples

```
## Not run:
library(connectapi)
client <- connect()

# get the last 20 audit logs
get_audit_logs(client, limit = 20, asc_order = FALSE)

## End(Not run)
```

---

get\_bundles

*Get Bundles*

---

### Description

Lists bundles for a content item

### Usage

```
get_bundles(content)

delete_bundle(content, bundle_id)
```

### Arguments

content	A R6 Content item, as returned by <code>content_item()</code>
bundle_id	A specific bundle ID for a content item

**See Also**

Other content functions: `content_delete()`, `content_item()`, `content_title()`, `content_update()`, `create_random_name()`, `dashboard_url()`, `dashboard_url_chr()`, `delete_vanity_url()`, `deploy_repo()`, `get_environment()`, `get_image()`, `get_jobs()`, `get_vanity_url()`, `git`, `permissions`, `set_image_path()`, `set_run_as()`, `set_vanity_url()`, `swap_vanity_url()`, `verify_content_name()`

Other content functions: `content_delete()`, `content_item()`, `content_title()`, `content_update()`, `create_random_name()`, `dashboard_url()`, `dashboard_url_chr()`, `delete_vanity_url()`, `deploy_repo()`, `get_environment()`, `get_image()`, `get_jobs()`, `get_vanity_url()`, `git`, `permissions`, `set_image_path()`, `set_run_as()`, `set_vanity_url()`, `swap_vanity_url()`, `verify_content_name()`

---

get\_content

*Get information about content on the Posit Connect server*


---

**Description**

Get information about content on the Posit Connect server

**Usage**

```
get_content(src, guid = NULL, owner_guid = NULL, name = NULL, ..., .p = NULL)
```

**Arguments**

<code>src</code>	A Connect object
<code>guid</code>	The guid for a particular content item
<code>owner_guid</code>	The unique identifier of the user who owns the content
<code>name</code>	The content name specified when the content was created
<code>...</code>	Extra arguments. Currently not used
<code>.p</code>	Optional. A predicate function, passed as-is to <code>purrr::keep()</code> before turning the response into a tibble. Can be useful for performance

**Details**

Please see <https://docs.posit.co/connect/api/#get-v1/content> for more information.

**Value**

A tibble with the following columns:

- `guid`: The unique identifier of this content item.
- `name`: A simple, URL-friendly identifier. Allows alpha-numeric characters, hyphens ("-"), and underscores ("\_").
- `title`: The title of this content.
- `description`: A rich description of this content
- `access_type`: Access type describes how this content manages its viewers. It may have a value of `all`, `logged_in` or `acl`. The value `all` is the most permissive; any visitor to Posit Connect will be able to view this content. The value `logged_in` indicates that all Posit Connect accounts may view the content. The `acl` value lets specifically enumerated users and groups view the content. Users configured as collaborators may always view content.

- `connection_timeout`: Maximum number of seconds allowed without data sent or received across a client connection. A value of 0 means connections will never time-out (not recommended). When null, the default `Scheduler.ConnectionTimeout` is used. Applies only to content types that are executed on demand.
- `read_timeout`: Maximum number of seconds allowed without data received from a client connection. A value of 0 means a lack of client (browser) interaction never causes the connection to close. When null, the default `Scheduler.ReadTimeout` is used. Applies only to content types that are executed on demand.
- `init_timeout`: The maximum number of seconds allowed for an interactive application to start. Posit Connect must be able to connect to a newly launched Shiny application, for example, before this threshold has elapsed. When null, the default `Scheduler.InitTimeout` is used. Applies only to content types that are executed on demand.
- `idle_timeout`: The maximum number of seconds a worker process for an interactive application to remain alive after it goes idle (no active connections). When null, the default `Scheduler.IdleTimeout` is used. Applies only to content types that are executed on demand.
- `max_processes`: Specifies the total number of concurrent processes allowed for a single interactive application. When null, the default `Scheduler.MaxProcesses` setting is used. Applies only to content types that are executed on demand.
- `min_processes`: Specifies the minimum number of concurrent processes allowed for a single interactive application. When null, the default `Scheduler.MinProcesses` is used. Applies only to content types that are executed on demand.
- `max_conns_per_process`: Specifies the maximum number of client connections allowed to an individual process. Incoming connections which will exceed this limit are routed to a new process or rejected. When null, the default `Scheduler.MaxConnsPerProcess` is used. Applies only to content types that are executed on demand.
- `load_factor`: Controls how aggressively new processes are spawned. When null, the default `Scheduler.LoadFactor` is used. Applies only to content types that are executed on demand.
- `created_time`: The timestamp (RFC3339) indicating when this content was created.
- `last_deployed_time`: The timestamp (RFC3339) indicating when this content last had a successful bundle deployment performed.
- `bundle_id`: The identifier for the active deployment bundle. Automatically assigned upon the successful deployment of that bundle.
- `app_mode`: The runtime model for this content. Has a value of `unknown` before data is deployed to this item. Automatically assigned upon the first successful bundle deployment. Allowed: `api`, `jupyter-static`, `python-api`, `python-bokeh`, `python-dash`, `python-streamlit`, `rmd-shiny`, `rmd-static`, `shiny`, `static`, `tensorflow-saved-model`, `unknown`.
- `content_category`: Describes the specialization of the content runtime model. Automatically assigned upon the first successful bundle deployment.
- `parameterized`: True when R Markdown rendered content allows parameter configuration. Automatically assigned upon the first successful bundle deployment. Applies only to content with an `app_mode` of `rmd-static`.
- `r_version`: The version of the R interpreter associated with this content. The value null represents that an R interpreter is not used by this content or that the R package environment has not been successfully restored. Automatically assigned upon the successful deployment of a bundle.
- `py_version`: The version of the Python interpreter associated with this content. The value null represents that a Python interpreter is not used by this content or that the Python package environment has not been successfully restored. Automatically assigned upon the successful deployment of a bundle.

- `run_as`: The UNIX user that executes this content. When null, the default `Applications.RunAs` is used. Applies only to executable content types - not static.
- `run_as_current_user`: Indicates if this content is allowed to execute as the logged-in user when using PAM authentication. Applies only to executable content types - not static.
- `owner_guid`: The unique identifier for the owner
- `content_url`: The URL associated with this content. Computed from the associated vanity URL or GUID for this content.
- `dashboard_url`: The URL within the Connect dashboard where this content can be configured. Computed from the GUID for this content.
- `role`: The relationship of the accessing user to this content. A value of `owner` is returned for the content owner. `editor` indicates a collaborator. The `viewer` value is given to users who are permitted to view the content. A `none` role is returned for administrators who cannot view the content but are permitted to view its configuration. Computed at the time of the request.
- `id`: The internal numeric identifier of this content item

### Examples

```
## Not run:
library(connectapi)
client <- connect()

get_content(client)

## End(Not run)
```

---

get_environment	<i>Manage Environment Variables</i>
-----------------	-------------------------------------

---

### Description

Manage Environment Variables for a piece of content.

### Usage

```
get_environment(content)

set_environment_new(env, ...)

set_environment_remove(env, ...)

set_environment_all(env, ...)
```

### Arguments

<code>content</code>	An R6 Content object as returned by <code>content_item()</code>
<code>env</code>	An R6 Environment object as returned by <code>get_environment()</code>
<code>...</code>	<code>name = value</code> pairs of environment variable names and values

**Details**

get\_environment() returns an Environment object for use with "setter" methods

set\_environment\_new() updates environment values (either creating new values or updating existing). Set NA as the value to remove a variable.

set\_environment\_remove() is a wrapper on set\_environment\_new() that allows removing named / listed variables quickly

set\_environment\_all() sets *all* environment variable values (will remove variables not specified)

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

 get\_groups

*Get group information from the Posit Connect server*


---

**Description**

Get group information from the Posit Connect server

**Usage**

```
get_groups(src, page_size = 500, prefix = NULL, limit = Inf)
```

**Arguments**

src	The source object
page_size	the number of records to return per page (max 500)
prefix	Filters groups by prefix (group name). The filter is case insensitive.
limit	The max number of groups to return

**Details**

Please see <https://docs.posit.co/connect/api/#getGroups> for more information.

**Value**

A tibble with the following columns:

- guid: The unique identifier of the group
- name: The group name
- owner\_guid: The group owner's unique identifier. When using LDAP or Proxied authentication with group provisioning enabled this property will always be null.



## Examples

```
## Not run:
library(connectapi)
client <- connect()

# get all groups
get_groups(client, limit = Inf)

## End(Not run)
```

---

get_group_members	<i>Get users within a specific group</i>
-------------------	--

---

## Description

Get users within a specific group

## Usage

```
get_group_members(src, guid)
```

## Arguments

src	The source object
guid	A group GUID identifier

## Details

Please see <https://docs.posit.co/connect/api/#getGroupMembers> for more information.

## Value

A tibble with the following columns:

- email: The user's email
- username: The user's username
- first\_name: The user's first name
- last\_name: The user's last name
- user\_role: The user's role. It may have a value of administrator, publisher or viewer.
- created\_time: The timestamp (in RFC3339 format) when the user was created in the Posit Connect server
- updated\_time: The timestamp (in RFC3339 format) when the user was last updated in the Posit Connect server
- active\_time: The timestamp (in RFC3339 format) when the user was last active on the Posit Connect server
- confirmed: When false, the created user must confirm their account through an email. This feature is unique to password authentication.
- locked: Whether or not the user is locked
- guid: The user's GUID, or unique identifier, in UUID RFC4122 format

## Examples

```
## Not run:
library(connectapi)
client <- connect()

# get the first 20 groups
groups <- get_groups(client)

group_guid <- groups$guid[1]

get_group_members(client, guid = group_guid)

## End(Not run)
```

---

get\_image

*Get the Content Image*

---

## Description

**[Experimental]** `get_image` saves the content image to the given path (default: temp file). `delete_image` removes the image (optionally saving to the given path) `has_image` returns whether the content has an image

## Usage

```
get_image(content, path = NULL)

delete_image(content, path = NULL)

has_image(content)
```

## Arguments

content	A content object
path	optional. The path to the image on disk

## See Also

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

get_jobs	<i>Get Jobs</i>
----------	-----------------

---

### Description

**[Experimental]** Retrieve details about jobs associated with a `content_item`. "Jobs" in Posit Connect are content executions

### Usage

```
get_jobs(content)
```

```
get_job(content, key)
```

### Arguments

content	A Content object, as returned by <code>content_item()</code>
key	The key for a job

### See Also

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

get_procs	<i>Get Real-Time Process Data</i>
-----------	-----------------------------------

---

### Description

**[Experimental]** This returns real-time process data from the Posit Connect API. It requires administrator privileges to use. NOTE that this only returns data for the server that responds to the request (i.e. in a Highly Available cluster)

### Usage

```
get_procs(src)
```

### Arguments

src	The source object
-----	-------------------

**Value**

A tibble with the following columns:

- pid: The PID of the current process
- appId: The application ID
- appId: The application GUID
- appName: The application name
- appUrl: The application URL
- appRunAs: The application RunAs user
- type: The type of process
- cpuCurrent: The current CPU usage
- cpuTotal: The total CPU usage
- ram: The current RAM usage

---

get\_tags

*Get all Tags on the server*

---

**Description**

Tag manipulation and assignment functions

**Usage**

get\_tags(src)

get\_tag\_data(src)

create\_tag(src, name, parent = NULL)

create\_tag\_tree(src, ...)

delete\_tag(src, tag)

get\_content\_tags(content)

set\_content\_tag\_tree(content, ...)

set\_content\_tags(content, ...)

filter\_tag\_tree\_id(tags, ids)

filter\_tag\_tree\_chr(tags, pattern)

**Arguments**

src	The source object
name	The name of the tag to create
parent	optional. A connect_tag_tree object (as returned by get_tags()) pointed at the parent tag
...	Additional arguments
	Manage tags (requires Administrator role):
	<ul style="list-style-type: none"> <li>• get_tags() - returns a "tag tree" object that can be traversed with tag_tree\$tag1\$childtag</li> <li>• get_tag_data() - returns a tibble of tag data</li> <li>• create_tag() - create a tag by specifying the Parent directly</li> <li>• create_tag_tree() - create tag(s) by specifying the "desired" tag tree hierarchy</li> <li>• delete_tag() - delete a tag (and its children). WARNING: will disassociate any content automatically</li> </ul>
	Manage content tags:
	<ul style="list-style-type: none"> <li>• get_content_tags() - return a connect_tag_tree object corresponding to the tags for a piece of content.</li> <li>• set_content_tag_tree() - attach a tag to content by specifying the desired tag tree</li> <li>• set_content_tags() - Set multiple tags at once by providing connect_tag_tree objects</li> </ul>
	Search a tag tree:
	<ul style="list-style-type: none"> <li>• filter_tag_tree_chr() - filters a tag tree based on a regex</li> <li>• filter_tag_tree_id() - filters a tag tree based on an id</li> </ul>
tag	A connect_tag_tree object (as returned by get_tags())
content	An R6 Content object, as returned by content_item()
tags	A connect_tag_tree object (as returned by get_tags())
ids	A list of ids to filter the tag tree by
pattern	A regex to filter the tag tree by (it is passed to grepl)

get\_timezones

*Get TimeZones***Description**

Get the available timezones from the server.

**Usage**

```
get_timezones(connect)
```

**Arguments**

connect	An R6 Connect object
---------	----------------------

**Value**

A TimeZone vector to be used for setting time zones

**See Also**

Other schedule functions: [get\\_variant\\_schedule\(\)](#), [set\\_schedule\(\)](#)

---

get\_usage\_shiny

*Get usage information for deployed shiny applications*


---

**Description**

Get usage information for deployed shiny applications

**Usage**

```
get_usage_shiny(
  src,
  content_guid = NULL,
  min_data_version = NULL,
  from = NULL,
  to = NULL,
  limit = 500,
  previous = NULL,
  nxt = NULL,
  asc_order = TRUE
)
```

**Arguments**

src	the source object
content_guid	Filter results by content GUID
min_data_version	Filter by data version. Records with a data version lower than the given value will be excluded from the set of results.
from	The timestamp that starts the time window of interest. Any usage information that ends prior to this timestamp will not be returned. Individual records may contain a starting time that is before this if they end after it or have not finished. Must be of class Date or POSIX
to	The timestamp that ends the time window of interest. Any usage information that starts after this timestamp will not be returned. Individual records may contain an ending time that is after this (or no ending time) if they start before it. Must be of class Date or POSIX
limit	The number of records to return.
previous	Retrieve the previous page of Shiny application usage logs relative to the provided value. This value corresponds to an internal reference within the server and should be sourced from the appropriate attribute within the paging object of a previous response.

nxt	Retrieve the next page of Shiny application usage logs relative to the provided value. This value corresponds to an internal reference within the server and should be sourced from the appropriate attribute within the paging object of a previous response.
asc_order	Defaults to TRUE; Determines if the response records should be listed in ascending or descending order within the response. Ordering is by the started timestamp field.

## Details

Please see <https://docs.posit.co/connect/api/#getShinyAppUsage> for more information.

## Value

A tibble with the following columns:

- `content_guid`: The GUID, in RFC4122 format, of the Shiny application this information pertains to.
- `user_guid`: The GUID, in RFC4122 format, of the user that visited the application.
- `started`: The timestamp, in RFC3339 format, when the user opened the application.
- `ended`: The timestamp, in RFC3339 format, when the user left the application.
- `data_version`: The data version the record was recorded with. The Shiny Application Events section of the Posit Connect Admin Guide explains how to interpret `data_version` values.

## Examples

```
## Not run:
library(connectapi)
client <- connect()

from <- Sys.Date() - lubridate::days(5)
get_usage_shiny(client, limit = 20, from = from)

## End(Not run)
```

---

<code>get_usage_static</code>	<i>Get usage information from deployed static content</i>
-------------------------------	---

---

## Description

This function retrieves usage information from static content on the Posit Connect server (e.g. Rmarkdown, Jupyter Notebooks)

## Usage

```
get_usage_static(
  src,
  content_guid = NULL,
  min_data_version = NULL,
  from = NULL,
```

```

to = NULL,
limit = 500,
previous = NULL,
nxt = NULL,
asc_order = TRUE
)

```

### Arguments

<code>src</code>	the source object
<code>content_guid</code>	Filter results by content GUID
<code>min_data_version</code>	Filter by data version. Records with a data version lower than the given value will be excluded from the set of results.
<code>from</code>	The timestamp that starts the time window of interest. Any usage information that ends prior to this timestamp will not be returned. Individual records may contain a starting time that is before this if they end after it or have not finished. Must be of class <code>Date</code> or <code>POSIX</code>
<code>to</code>	The timestamp that ends the time window of interest. Any usage information that starts after this timestamp will not be returned. Individual records may contain an ending time that is after this (or no ending time) if they start before it. Must be of class <code>Date</code> or <code>POSIX</code>
<code>limit</code>	The number of records to return.
<code>previous</code>	Retrieve the previous page of Shiny application usage logs relative to the provided value. This value corresponds to an internal reference within the server and should be sourced from the appropriate attribute within the paging object of a previous response.
<code>nxt</code>	Retrieve the next page of Shiny application usage logs relative to the provided value. This value corresponds to an internal reference within the server and should be sourced from the appropriate attribute within the paging object of a previous response.
<code>asc_order</code>	Defaults to <code>TRUE</code> ; Determines if the response records should be listed in ascending or descending order within the response. Ordering is by the started timestamp field.

### Details

Please see <https://docs.posit.co/connect/api/#getContentVisits> for more information.

### Value

A tibble with the following columns:

- `content_guid`: The GUID, in RFC4122 format, of the Shiny application this information pertains to.
- `user_guid`: The GUID, in RFC4122 format, of the user that visited the application.
- `variant_key`: The key of the variant the user visited. This will be null for static content.
- `time`: The timestamp, in RFC3339 format, when the user visited the content.
- `rendering_id`: The ID of the rendering the user visited. This will be null for static content.
- `bundle_id`: The ID of the particular bundle used.



- `data_version`: The data version the record was recorded with. The Rendered and Static Content Visit Events section of the Posit Connect Admin Guide explains how to interpret `data_version` values.

### Examples

```
## Not run:
library(connectapi)
client <- connect()

from <- Sys.Date() - lubridate::days(5)
get_usage_static(client, limit = 20, from = from)

## End(Not run)
```

---

get\_users

*Get user information from the Posit Connect server*

---

### Description

Get user information from the Posit Connect server

### Usage

```
get_users(src, page_size = 500, prefix = NULL, limit = Inf)
```

### Arguments

<code>src</code>	The source object
<code>page_size</code>	the number of records to return per page (max 500)
<code>prefix</code>	Filters users by prefix (username, first name, or last name). The filter is case insensitive.
<code>limit</code>	The max number of records to return

### Details

Please see <https://docs.posit.co/connect/api/#get-v1/users> for more information.

### Value

A tibble with the following columns:

- `email`: The user's email
- `username`: The user's username
- `first_name`: The user's first name
- `last_name`: The user's last name
- `user_role`: The user's role. It may have a value of administrator, publisher or viewer.
- `created_time`: The timestamp (in RFC3339 format) when the user was created in the Posit Connect server

- `updated_time`: The timestamp (in RFC3339 format) when the user was last updated in the Posit Connect server
- `active_time`: The timestamp (in RFC3339 format) when the user was last active on the Posit Connect server
- `confirmed`: When false, the created user must confirm their account through an email. This feature is unique to password authentication.
- `locked`: Whether or not the user is locked
- `guid`: The user's GUID, or unique identifier, in UUID RFC4122 format

### Examples

```
## Not run:
library(connectapi)
client <- connect()

# get all users
get_users(client, limit = Inf)

## End(Not run)
```

---

<code>get_vanity_url</code>	<i>Get the Vanity URL</i>
-----------------------------	---------------------------

---

### Description

Gets the Vanity URL for a piece of content.

### Usage

```
get_vanity_url(content)
```

### Arguments

`content`            A Content object

### Value

A character string (or NULL if not defined)

### See Also

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

get_variants	<i>Get Variant</i>
--------------	--------------------

---

**Description**

**[Experimental]** Work with variants

**Usage**

```
get_variants(content)
```

```
get_variant_default(content)
```

```
get_variant(content, key)
```

**Arguments**

content	An R6 Content object. Returned from <code>content_item()</code>
key	The Variant key for a specific variant

**Details**

- `get_variants()` returns a tibble with variant data for a `content_item`
- `get_default_variant()` returns the default variant for a `content_item`
- `get_variant()` returns a specific variant for a `content_item` (specified by key)

**See Also**

Other variant functions: [get\\_variant\\_renderings\(\)](#)

Other variant functions: [get\\_variant\\_renderings\(\)](#)

Other variant functions: [get\\_variant\\_renderings\(\)](#)

---

get_variant_renderings
------------------------

*Render a Variant*

---

**Description**

**[Experimental]** Get details about renderings (i.e. render history) or execute a variant on demand

**Usage**

```
get_variant_renderings(variant)
```

```
variant_render(variant)
```

**Arguments**

variant	An R6 Variant object. As returned by <code>get_variant()</code> or <code>get_variant_default()</code>
---------	---

**Details**

- `get_variant_renderings()` returns all renderings / content for a particular variant. Returns a tibble
- `variant_render()` executes a variant on demand. Returns a `VariantTask` object

**See Also**

Other variant functions: [get\\_variants\(\)](#)

---

`get_variant_schedule`    *Get a Variant Schedule*

---

**Description**

**[Experimental]** Gets the schedule associated with a `Variant`.

**Usage**

```
get_variant_schedule(variant)
```

**Arguments**

`variant`            A `Variant` object, as returned by `get_variant()` or `get_variant_default()`

**Value**

A `VariantSchedule` object

**See Also**

Other schedule functions: [get\\_timezones\(\)](#), [set\\_schedule\(\)](#)

---

`git`                    *Git Repository Helpers*

---

**Description**

**[Experimental]** These functions help use Posit Connect's configured authorization to query available branches and subdirectories for deployment using `deploy_repo()`

**Usage**

```
repo_check_account(client, host)
```

```
repo_check_branches(client, repository)
```

```
repo_check_branches_ref(client, repository)
```

```
repo_check_manifest_dirs(client, repository, branch)
```

**Arguments**

client	A Connect R6 object
host	The git repository host (with schema). For example, "https://github.com"
repository	The git repository to explore or consider deploying
branch	The git branch to explore for subdirectories

**Details**

- `repo_check_account()` messages whether an account is in use, and then returns that account
- `repo_check_branches()` retrieves which branches are available, returning in a named list
- `repo_check_manifest_dirs()` retrieves which directories contain a `manifest.json`, returning in a named list

**See Also**

`connectapi::deploy_repo`

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

groups\_create\_remote    *Create a Remote Group*

---

**Description**

Create a Remote Group

**Usage**

```
groups_create_remote(connect, prefix, expect = 1, check = TRUE)
```

**Arguments**

connect	A R6 Connect object
prefix	character. The prefix of the group name to search for
expect	number. The number of responses to expect for this search
check	boolean. Whether to check for local existence first

**Value**

The results of creating the groups

---

page_cursor	<i>Paging</i>
-------------	---------------

---

**Description**

Helper functions that make paging easier in the Posit Connect Server API.

Helper functions that make paging easier in the Posit Connect Server API.

**Usage**

```
page_cursor(client, req, limit = Inf)
```

```
page_offset(client, req, limit = Inf)
```

**Arguments**

client	A Connect client object
req	The request that needs to be paged
limit	A row limit

**Value**

The aggregated results from all requests

The aggregated results from all requests

---

permissions	<i>Content permissions</i>
-------------	----------------------------

---

**Description**

Get or set content permissions for a content item

**Usage**

```
content_add_user(content, guid, role = c("viewer", "owner"))
```

```
content_add_group(content, guid, role = c("viewer", "owner"))
```

```
content_delete_user(content, guid)
```

```
content_delete_group(content, guid)
```

```
get_user_permission(content, guid, add_owner = TRUE)
```

```
get_my_permission(content, add_owner = TRUE)
```

```
get_group_permission(content, guid)
```

```
get_content_permissions(content, add_owner = TRUE)
```

**Arguments**

content	An R6 content object
guid	The guid associated with either a user (for content_add_user) or group (for content_add_group)
role	The role to assign to a user. Either "viewer" or "owner." Defaults to "viewer"
add_owner	Optional. Whether to include the owner in returned permission sets. Default is TRUE. The owner will have an NA_character_ permission "id"

**Details**

Permission modification:

- content\_add\_\* adds a permission to the content
- content\_delete\_\* removes a permission from the content

Permission retrieval:

- get\_content\_permissions() lists permissions
- get\_my\_permission() gets the permission associated with the caller.
- get\_user\_permission() gets the permissions associated with a given user. It does not evaluate group memberships
- get\_group\_permission() gets the permissions associated with a given group.

NOTE: by default, the owner is injected with an "NA\_character\_" permission id. This makes it easier to find / isolate this record.

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

poll\_task

*Poll Task*

---

**Description**

Polls a task, waiting for information about a deployment. If the task has results, the output will be a modified "Task" object with task\$get\_data() available to retrieve the results.

**Usage**

```
poll_task(task, wait = 1, callback = message)
```

**Arguments**

task	A Task object
wait	The interval to wait between polling
callback	A function to be called for each message received. Set to NULL for no callback

**Details**

For a simple way to silence messages, set `callback = NULL`

**Value**

Task The Task object that was input

**See Also**

Other deployment functions: [bundle\\_dir\(\)](#), [bundle\\_path\(\)](#), [bundle\\_static\(\)](#), [deploy\(\)](#), [download\\_bundle\(\)](#)

---

PositConnect	<i>Class representing a Connect API client</i>
--------------	--

---

**Description**

Class representing a Connect API client

Class representing a Connect API client

**Usage**

```
client <- Connect$new(server = 'connect.example.com',
  apiKey = 'mysecretkey')
client$get_apps()
client$get_tags()
```

**Details**

This class allows a user to interact with a Connect server via the Connect API. Authentication is done by providing an API key.

**Public fields**

`server` The base URL of your Posit Connect server.

`api_key` Your Posit Connect API key.

`tags` The initial set of tags.

`tag_map` The initial tag map.

`httr_additions` An initial set of httr configuration added to each HTTP call.

`using_auth` Indicates that the API key is added to each HTTP call.

**Methods****Public methods:**

- [Connect\\$get\\_connect\(\)](#)
- [Connect\\$new\(\)](#)
- [Connect\\$httr\\_config\(\)](#)
- [Connect\\$print\(\)](#)
- [Connect\\$raise\\_error\(\)](#)
- [Connect\\$add\\_auth\(\)](#)



- `Connect$GET()`
- `Connect$GET_RESULT()`
- `Connect$GET_URL()`
- `Connect$GET_RESULT_URL()`
- `Connect$PUT()`
- `Connect$HEAD()`
- `Connect$DELETE()`
- `Connect$PATCH()`
- `Connect$POST()`
- `Connect$me()`
- `Connect$get_dashboard_url()`
- `Connect$get_tags()`
- `Connect$get_tag_id()`
- `Connect$get_tag_tree()`
- `Connect$tag_create_safe()`
- `Connect$tag_create()`
- `Connect$tag()`
- `Connect$tag_delete()`
- `Connect$get_apps()`
- `Connect$get_schedule()`
- `Connect$content_create()`
- `Connect$content_upload()`
- `Connect$content_deploy()`
- `Connect$content()`
- `Connect$task()`
- `Connect$set_content_tag()`
- `Connect$remove_content_tag()`
- `Connect$user()`
- `Connect$users()`
- `Connect$users_remote()`
- `Connect$users_create()`
- `Connect$users_create_remote()`
- `Connect$users_lock()`
- `Connect$users_unlock()`
- `Connect$users_update()`
- `Connect$groups()`
- `Connect$group_members()`
- `Connect$group_member_add()`
- `Connect$group_member_remove()`
- `Connect$groups_create()`
- `Connect$groups_create_remote()`
- `Connect$groups_remote()`
- `Connect$inst_content_visits()`
- `Connect$inst_shiny_usage()`
- `Connect$procs()`

- `Connect$repo_account()`
- `Connect$repo_branches()`
- `Connect$repo_manifest_dirs()`
- `Connect$schedules()`
- `Connect$docs()`
- `Connect$audit_logs()`
- `Connect$server_settings_r()`
- `Connect$server_settings()`
- `Connect$clone()`

**Method** `get_connect()`: Return this connect.

*Usage:*

```
Connect$get_connect()
```

**Method** `new()`: Initialize a new connect.

*Usage:*

```
Connect$new(server, api_key)
```

*Arguments:*

`server` The base URL of your Posit Connect server.

`api_key` Your Posit Connect API key.

**Method** `httr_config()`: Set additional httr configuration that is added to each HTTP call.

*Usage:*

```
Connect$httr_config(...)
```

*Arguments:*

... Set of httr configurations.

**Method** `print()`: Print details about this instance.

*Usage:*

```
Connect$print(...)
```

*Arguments:*

... Ignored.

**Method** `raise_error()`: Raise an error when the HTTP result is an HTTP error.

*Usage:*

```
Connect$raise_error(res)
```

*Arguments:*

`res` HTTP result.

**Method** `add_auth()`: Returns HTTP authorization headers, or NULL when none are used.

*Usage:*

```
Connect$add_auth()
```

**Method** `GET()`: Perform an HTTP GET request of the named API path. Returns an object parsed from the HTTP response.

*Usage:*

```
Connect$GET(path, writer = httr::write_memory(), parser = "parsed", ...)
```

*Arguments:*

path API path.  
writer Controls where the response is written.  
parser How the response is parsed.  
... Arguments to the httr::GET.

**Method GET\_RESULT():** Perform an HTTP GET request of the named API path. Returns the HTTP response object.

*Usage:*

```
Connect$GET_RESULT(path, writer = httr::write_memory(), ...)
```

*Arguments:*

path API path.  
writer Controls where the response is written.  
... Arguments to the httr::GET.

**Method GET\_URL():** Perform an HTTP GET request of the named URL. Returns an object parsed from the HTTP response.

*Usage:*

```
Connect$GET_URL(url, writer = httr::write_memory(), parser = "parsed", ...)
```

*Arguments:*

url Target URL.  
writer Controls where the response is written.  
parser How the response is parsed.  
... Arguments to the httr::GET.

**Method GET\_RESULT\_URL():** Perform an HTTP GET request of the named URL. Returns the HTTP response object.

*Usage:*

```
Connect$GET_RESULT_URL(url, writer = httr::write_memory(), ...)
```

*Arguments:*

url Target URL.  
writer Controls where the response is written.  
... Arguments to the httr::GET.

**Method PUT():** Perform an HTTP PUT request of the named API path. Returns an object parsed from the HTTP response.

*Usage:*

```
Connect$PUT(path, body, encode = "json", ..., .empty_object = TRUE)
```

*Arguments:*

path API path.  
body The HTTP payload.  
encode How the payload is encoded.  
... Arguments to the httr::PUT.  
.empty\_object Indicates that an empty JSON object is sent when the body is empty.

**Method HEAD():** Perform an HTTP HEAD request of the named API path. Returns the HTTP response object.

*Usage:*

```
Connect$HEAD(path, ...)
```

*Arguments:*

path API path.

... Arguments to the httr::HEAD.

**Method DELETE():** Perform an HTTP DELETE request of the named API path. Returns the HTTP response object.

*Usage:*

```
Connect$DELETE(path, ...)
```

*Arguments:*

path API path.

... Arguments to the httr::DELETE.

**Method PATCH():** Perform an HTTP PATCH request of the named API path. Returns an object parsed from the HTTP response.

*Usage:*

```
Connect$PATCH(
  path,
  body,
  encode = "json",
  prefix = "/__api__/",
  ...,
  .empty_object = TRUE
)
```

*Arguments:*

path API path.

body The HTTP payload.

encode How the payload is encoded.

prefix API path prefix.

... Arguments to the httr::PATCH.

.empty\_object Indicates that an empty JSON object is sent when the body is empty.

**Method POST():** Perform an HTTP POST request of the named API path. Returns an object parsed from the HTTP response.

*Usage:*

```
Connect$POST(
  path,
  body,
  encode = "json",
  prefix = "/__api__/",
  ...,
  .empty_object = TRUE
)
```

*Arguments:*

path API path.

body The HTTP payload.

encode How the payload is encoded.

prefix API path prefix.  
... Arguments to the http::POST.  
.empty\_object Indicates that an empty JSON object is sent when the body is empty.

**Method** me(): Perform an HTTP GET request of the "me" server endpoint.

*Usage:*

Connect\$me()

**Method** get\_dashboard\_url(): Return the base URL of the Connect server.

*Usage:*

Connect\$get\_dashboard\_url()

**Method** get\_tags(): Return all tags.

*Usage:*

Connect\$get\_tags(use\_cache = FALSE)

*Arguments:*

use\_cache Indicates that a cached set of tags is used.

**Method** get\_tag\_id(): Get the identifier for the named tag.

*Usage:*

Connect\$get\_tag\_id(tagname)

*Arguments:*

tagname The name of the tag.

**Method** get\_tag\_tree(): Get the tag tree.

*Usage:*

Connect\$get\_tag\_tree()

**Method** tag\_create\_safe(): Create a tag.

*Usage:*

Connect\$tag\_create\_safe(name, parent\_id = NULL)

*Arguments:*

name The tag name.

parent\_id The parent identifier.

**Method** tag\_create(): Create a tag.

*Usage:*

Connect\$tag\_create(name, parent\_id = NULL)

*Arguments:*

name The tag name.

parent\_id The parent identifier.

**Method** tag(): Get a tag.

*Usage:*

Connect\$tag(id = NULL)

*Arguments:*

id The tag identifier.

**Method** tag\_delete(): Delete a tag.

*Usage:*

```
Connect$tag_delete(id)
```

*Arguments:*

id The tag identifier.

**Method** get\_apps(): Get content items.

*Usage:*

```
Connect$get_apps(filter = NULL, .collapse = "&", .limit = Inf, page_size = 25)
```

*Arguments:*

filter Named list containing filter conditions.

.collapse How multiple filters are combined.

.limit The limit.

page\_size The page size.

**Method** get\_schedule(): Get a schedule.

*Usage:*

```
Connect$get_schedule(schedule_id)
```

*Arguments:*

schedule\_id The schedule identifier.

**Method** content\_create(): Create content.

*Usage:*

```
Connect$content_create(name, title = name, ...)
```

*Arguments:*

name The content name.

title The content title.

... Other content fields.

**Method** content\_upload(): Upload a content bundle.

*Usage:*

```
Connect$content_upload(bundle_path, guid)
```

*Arguments:*

bundle\_path The path to the bundle archive.

guid The content GUID.

**Method** content\_deploy(): Deploy a content bundle.

*Usage:*

```
Connect$content_deploy(guid, bundle_id)
```

*Arguments:*

guid The content GUID.

bundle\_id The bundle identifier.

**Method** content(): Get a content item.

*Usage:*

```
Connect$content(  
  guid = NULL,  
  owner_guid = NULL,  
  name = NULL,  
  include = "tags,owner"  
)
```

*Arguments:*

guid The content GUID.  
owner\_guid The target content owner.  
name The target name.  
include Additional response fields.

**Method** task(): Get a task.

*Usage:*

```
Connect$task(task_id, first = 0, wait = 5)
```

*Arguments:*

task\_id The task identifier.  
first The initial status position.  
wait Maximum time to wait for update.

**Method** set\_content\_tag(): Set a tag for a content item.

*Usage:*

```
Connect$set_content_tag(content_id, tag_id)
```

*Arguments:*

content\_id The content identifier.  
tag\_id The tag identifier.

**Method** remove\_content\_tag(): Remove a tag from a content item.

*Usage:*

```
Connect$remove_content_tag(content_id, tag_id)
```

*Arguments:*

content\_id The content identifier.  
tag\_id The tag identifier.

**Method** user(): Get user details.

*Usage:*

```
Connect$user(guid)
```

*Arguments:*

guid The user GUID.

**Method** users(): Get users.

*Usage:*

```
Connect$users(page_number = 1, prefix = NULL, page_size = 500)
```

*Arguments:*

page\_number The page number.  
prefix The search term.

page\_size The page size.

**Method** users\_remote(): Get remote users.

*Usage:*

```
Connect$users_remote(prefix)
```

*Arguments:*

prefix The search term.

**Method** users\_create(): Create a user.

*Usage:*

```
Connect$users_create(
  username,
  email,
  first_name = NULL,
  last_name = NULL,
  password = NULL,
  user_must_set_password = NULL,
  user_role = NULL,
  unique_id = NULL
)
```

*Arguments:*

username The username.

email Email address.

first\_name First name.

last\_name Last name.

password The password.

user\_must\_set\_password Indicates that user sets password on first login.

user\_role Role for user.

unique\_id Identifier for user.

**Method** users\_create\_remote(): Create a remote user.

*Usage:*

```
Connect$users_create_remote(temp_ticket)
```

*Arguments:*

temp\_ticket Ticket identifying target remote user.

**Method** users\_lock(): Lock a user.

*Usage:*

```
Connect$users_lock(user_guid)
```

*Arguments:*

user\_guid User GUID.

**Method** users\_unlock(): Unlock a user.

*Usage:*

```
Connect$users_unlock(user_guid)
```

*Arguments:*

user\_guid User GUID.



**Method** `users_update()`: Update a user.

*Usage:*

```
Connect$users_update(user_guid, ...)
```

*Arguments:*

`user_guid` User GUID.

... User fields.

**Method** `groups()`: Get groups.

*Usage:*

```
Connect$groups(page_number = 1, prefix = NULL, page_size = 500)
```

*Arguments:*

`page_number` The page number.

`prefix` The search term.

`page_size` The page size.

**Method** `group_members()`: Get group members.

*Usage:*

```
Connect$group_members(guid)
```

*Arguments:*

`guid` The group GUID.

**Method** `group_member_add()`: Add a group member.

*Usage:*

```
Connect$group_member_add(group_guid, user_guid)
```

*Arguments:*

`group_guid` The group GUID.

`user_guid` The user GUID.

**Method** `group_member_remove()`: Remove a group member.

*Usage:*

```
Connect$group_member_remove(group_guid, user_guid)
```

*Arguments:*

`group_guid` The group GUID.

`user_guid` The user GUID.

**Method** `groups_create()`: Create a group.

*Usage:*

```
Connect$groups_create(name)
```

*Arguments:*

`name` The group name.

**Method** `groups_create_remote()`: Create a remote group.

*Usage:*

```
Connect$groups_create_remote(temp_ticket)
```

*Arguments:*

temp\_ticket Ticket identifying target remote group.

**Method** groups\_remote(): Get remote groups.

*Usage:*

```
Connect$groups_remote(prefix = NULL, limit = 500)
```

*Arguments:*

prefix The search term.

limit The maximal result set size.

**Method** inst\_content\_visits(): Get (non-interactive) content visits.

*Usage:*

```
Connect$inst_content_visits(
  content_guid = NULL,
  min_data_version = NULL,
  from = NULL,
  to = NULL,
  limit = 500,
  previous = NULL,
  nxt = NULL,
  asc_order = TRUE
)
```

*Arguments:*

content\_guid Content GUID.

min\_data\_version Data version for request.

from Start of range.

to End of range.

limit Result set size.

previous Previous item.

nxt Next item.

asc\_order Indicates ascending result order.

**Method** inst\_shiny\_usage(): Get interactive content visits.

Get (non-interactive) content visits.

*Usage:*

```
Connect$inst_shiny_usage(
  content_guid = NULL,
  min_data_version = NULL,
  from = NULL,
  to = NULL,
  limit = 500,
  previous = NULL,
  nxt = NULL,
  asc_order = TRUE
)
```

*Arguments:*

content\_guid Content GUID.

min\_data\_version Data version for request.

from Start of range.

to End of range.  
limit Result set size.  
previous Previous item.  
nxt Next item.  
asc\_order Indicates ascending result order.

**Method** `procs()`: Get running processes.

*Usage:*

```
Connect$procs()
```

**Method** `repo_account()`: Determine if Git repository is associated with authorization.

*Usage:*

```
Connect$repo_account(host)
```

*Arguments:*

host Repository URL.

**Method** `repo_branches()`: Get Git repository branches.

*Usage:*

```
Connect$repo_branches(repo)
```

*Arguments:*

repo Repository URL.

**Method** `repo_manifest_dirs()`: Get Git repository directories.

*Usage:*

```
Connect$repo_manifest_dirs(repo, branch)
```

*Arguments:*

repo Repository URL.

branch Repository branch.

**Method** `schedules()`: Get schedules.

*Usage:*

```
Connect$schedules(  
  start = Sys.time(),  
  end = Sys.time() + 60 * 60 * 24 * 7,  
  detailed = FALSE  
)
```

*Arguments:*

start Starting time.

end Ending time.

detailed Indicates detailed schedule information.

**Method** `docs()`: Get documentation.

*Usage:*

```
Connect$docs(docs = "api", browse = TRUE)
```

*Arguments:*

docs Named document.

browse Open a browser.

**Method** `audit_logs()`: Get auditing.

*Usage:*

```
Connect$audit_logs(limit = 500, previous = NULL, nxt = NULL, asc_order = TRUE)
```

*Arguments:*

`limit` Result set size.

`previous` Previous item.

`nxt` Next item.

`asc_order` Indicates ascending result order.

**Method** `server_settings_r()`: Get R installations.

*Usage:*

```
Connect$server_settings_r()
```

**Method** `server_settings()`: Get server settings.

*Usage:*

```
Connect$server_settings()
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Connect$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

Other R6 classes: [Bundle](#), [Content](#), [ContentTask](#), [Environment](#), [Task](#), [Vanity](#), [Variant](#), [VariantSchedule](#), [VariantTask](#)

---

promote

*Promote content from one Connect server to another*

---

## Description

Promote content from one Connect server to another

## Usage

```
promote(from, to, to_key, from_key, name)
```

## Arguments

<code>from</code>	The url for the server containing the content (the originating server)
<code>to</code>	The url for the server where the content will be deployed (the destination server)
<code>to_key</code>	An API key on the destination "to" server. If the destination content is going to be updated, the API key must belong to a user with collaborator access on the content that will be updated. If the destination content is to be created new, the API key must belong to a user with publisher privileges.

from_key	An API key on the originating "from" server. The API key must belong to a user with collaborator access to the content to be promoted.
name	The name of the content on the originating "from" server. If content with the same name is found on the destination server, the content will be updated. If no content on the destination server has a matching name, a new endpoint will be created.

**Value**

The URL for the content on the destination "to" server

---

set_image_path	<i>Set the Content Image</i>
----------------	------------------------------

---

**Description**

**[Experimental]**

**Usage**

```
set_image_path(content, path)
```

```
set_image_url(content, url)
```

```
set_image_webshot(content, ...)
```

**Arguments**

content	A content object
path	The path to an image on disk
url	The url for an image
...	Additional arguments passed on to <a href="#">webshot2::webshot()</a>

**Details**

Set the Content Image using a variety of methods.

NOTE: `set_image_webshot()` requires [webshot2::webshot\(\)](#), but currently skips and warns for any content that requires authentication until the `webshot2` package supports authentication.

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git\\_permissions](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

set_run_as	<i>Set RunAs User</i>
------------	-----------------------

---

### Description

Set the RunAs user for a piece of content. The run\_as\_current\_user flag only does anything if:

### Usage

```
set_run_as(content, run_as, run_as_current_user = FALSE)
```

### Arguments

content	an R6 Content item
run_as	The RunAs user to use for this content
run_as_current_user	Whether to run this content as the viewer of the application

### Details

- PAM is the authentication method
- Applications.RunAsCurrentUser is enabled on the server

Also worth noting that the run\_as user must exist on the Posit Connect server (as a linux user) and have appropriate group memberships, or you will get a 400: Bad Request. Set to NULL to use the default RunAs user / unset any current configuration.

To "read" the current RunAs user, use the Content object or get\_content() function.

### Value

a Content object, updated with new details

### See Also

connectapi::content\_update

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git\\_permissions](#), [set\\_image\\_path\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

set_schedule	<i>Set a Schedule</i>
--------------	-----------------------

---

### Description

**[Experimental]** Sets the schedule for a given Variant. Requires a Schedule object (as returned by `get_variant_schedule()`)

### Usage

```
set_schedule(.schedule, ...)
```

```
set_schedule_minute(  
  .schedule,  
  n = 30,  
  start_time = Sys.time(),  
  activate = TRUE,  
  email = FALSE,  
  timezone = Sys.timezone()  
)
```

```
set_schedule_hour(  
  .schedule,  
  n = 1,  
  start_time = Sys.time(),  
  activate = TRUE,  
  email = FALSE,  
  timezone = Sys.timezone()  
)
```

```
set_schedule_day(  
  .schedule,  
  n = 1,  
  start_time = Sys.time(),  
  activate = TRUE,  
  email = FALSE,  
  timezone = Sys.timezone()  
)
```

```
set_schedule_weekday(  
  .schedule,  
  start_time = Sys.time(),  
  activate = TRUE,  
  email = FALSE,  
  timezone = Sys.timezone()  
)
```

```
set_schedule_week(  
  .schedule,  
  n = 1,  
  start_time = Sys.time(),
```

```
        activate = TRUE,  
        email = FALSE,  
        timezone = Sys.timezone()  
    )  
  
    set_schedule_dayofweek(  
        .schedule,  
        days,  
        start_time = Sys.time(),  
        activate = TRUE,  
        email = FALSE,  
        timezone = Sys.timezone()  
    )  
  
    set_schedule_semimonth(  
        .schedule,  
        first = TRUE,  
        start_time = Sys.time(),  
        activate = TRUE,  
        email = FALSE,  
        timezone = Sys.timezone()  
    )  
  
    set_schedule_dayofmonth(  
        .schedule,  
        n = 1,  
        day = 1,  
        start_time = Sys.time(),  
        activate = TRUE,  
        email = FALSE,  
        timezone = Sys.timezone()  
    )  
  
    set_schedule_dayweekofmonth(  
        .schedule,  
        n = 1,  
        day = 1,  
        week = 1,  
        start_time = Sys.time(),  
        activate = TRUE,  
        email = FALSE,  
        timezone = Sys.timezone()  
    )  
  
    set_schedule_year(  
        .schedule,  
        n = 1,  
        start_time = Sys.time(),  
        activate = TRUE,  
        email = FALSE,  
        timezone = Sys.timezone()  
    )  
)
```



```
set_schedule_remove(.schedule)
```

```
schedule_describe(.schedule)
```

### Arguments

<code>.schedule</code>	A schedule object. As returned by <code>get_variant_schedule()</code>
<code>...</code>	Scheduling parameters
<code>n</code>	The "number of" iterations
<code>start_time</code>	The start time of the schedule
<code>activate</code>	Whether to publish the output of this schedule
<code>email</code>	Whether to send emails on this schedule
<code>timezone</code>	The timezone to use for setting the schedule. Defaults to <code>Sys.timezone()</code>
<code>days</code>	The days of the week (0-6)
<code>first</code>	<a href="#">logical</a> Whether to execute on the 1st and 15th (TRUE) or 14th and last (FALSE)
<code>day</code>	The day of the week (0-6) or day of the month (0-31)
<code>week</code>	The week of the month (0-5)
<code>schedule</code>	A JSON blob (as a string) describing the schedule. See "More Details"

### Details

- `set_schedule()` is a raw interface to Posit Connect's schedule API
- `set_schedule_*` functions provide handy wrappers around `set_schedule()`
- `set_schedule_remove()` removes a schedule / un-schedules a variant

Beware, using `set_schedule()` currently uses the Posit Connect schedule API directly, and so can be a little clunky. Using the `set_schedule_*` is generally recommended.

### Value

An updated Schedule object

### See Also

Other schedule functions: [get\\_timezones\(\)](#), [get\\_variant\\_schedule\(\)](#)

---

set\_vanity\_url

*Set the Vanity URL*

---

### Description

Sets the Vanity URL for a piece of content.

### Usage

```
set_vanity_url(content, url, force = FALSE)
```

**Arguments**

content	A Content object
url	The path component of the URL
force	optional. Default FALSE. Whether to force-reassign a vanity URL that already exists

**Value**

An updated Content object

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [swap\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

**Examples**

```
## Not run:
bnd <- bundle_dir("~/my/directory")
connect() %>%
  deploy(bnd) %>%
  set_vanity_url("a/vanity/url")

## End(Not run)
```

---

swap_vanity_url	<i>Swap the Vanity URL</i>
-----------------	----------------------------

---

**Description**

Swaps the Vanity URLs between two pieces of content

**Usage**

```
swap_vanity_url(from_content, to_content)
```

**Arguments**

from_content	A Content object
to_content	A Content object

**See Also**

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [verify\\_content\\_name\(\)](#)

---

Task

*Task*

---

## Description

Task

Task

## Details

An R6 class that represents a Task

## Public fields

connect The Connect instance.

task The task.

data The task data.

## Methods

### Public methods:

- [Task\\$new\(\)](#)
- [Task\\$get\\_connect\(\)](#)
- [Task\\$get\\_task\(\)](#)
- [Task\\$add\\_data\(\)](#)
- [Task\\$get\\_data\(\)](#)
- [Task\\$print\(\)](#)
- [Task\\$clone\(\)](#)

**Method** `new()`: Initialize this task.

*Usage:*

```
Task$new(connect, task)
```

*Arguments:*

connect The Connect instance.

task The task data.

**Method** `get_connect()`: Return the associated Connect instance.

*Usage:*

```
Task$get_connect()
```

**Method** `get_task()`: Return the underlying task.

*Usage:*

```
Task$get_task()
```

**Method** `add_data()`: Set the data.

*Usage:*

```
Task$add_data(data)
```

*Arguments:*

data The data.

**Method** `get_data()`: Get the data.

*Usage:*

```
Task$get_data()
```

**Method** `print()`: Print this object.

*Usage:*

```
Task$print(...)
```

*Arguments:*

... Unused.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Task$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### See Also

Other R6 classes: [Bundle](#), [Content](#), [ContentTask](#), [Environment](#), [PositConnect](#), [Vanity](#), [Variant](#), [VariantSchedule](#), [VariantTask](#)

---

tbl\_connect

*Connect Tibble*


---

### Description

**[Experimental]** A lazy tibble that automatically pages through API requests when collected.

### Usage

```
tbl_connect(
  src,
  from = c("users", "groups", "content", "usage_shiny", "usage_static", "audit_logs"),
  ...
)
```

### Arguments

src	The source object
from	The type of tibble
...	Additional arguments that are not yet implemented

### Value

A `tbl_connect` object

---

users\_create\_remote     *Create a Remote User*

---

### Description

The remote user creation workflow involves authentication providers like LDAP that involve a queryable identity store. This helper wraps the API calls necessary to retrieve information about and then create such a user. It functions with a "fuzzy match" prefix by default, but if you want to instantiate users directly, you should set `exact = TRUE`.

### Usage

```
users_create_remote(connect, prefix, expect = 1, check = TRUE, exact = FALSE)
```

### Arguments

connect	A R6 Connect object
prefix	character. The prefix of the user name to search for
expect	number. Optional. The number of responses to expect for this search
check	boolean. Optional. Whether to check for local existence first
exact	boolean. Optional. Whether to only create users whose username exactly matches the provided prefix.

### Details

NOTE: there can be problems with usernames that are not unique. Please open an issue if you run into any problems.

### Value

The results of creating the users

---

user\_guid\_from\_username  
*User*

---

### Description

Get user details

### Usage

```
user_guid_from_username(client, username)
```

### Arguments

client	A Connect R6 object
username	The user's username

**Details**

`user_guid_from_username()` is a helper to retrieve a user GUID, given the user's username. It is useful in Shiny applications for using `session$user`

---

Vanity

*Vanity*


---

**Description**

Vanity

Vanity

**Details**

An R6 class that represents a Vanity URL

**Super class**

`connectapi::Content` -> Vanity

**Public fields**

`vanity` The vanity.

**Methods****Public methods:**

- `Vanity$new()`
- `Vanity$get_vanity()`
- `Vanity$print()`
- `Vanity$clone()`

**Method** `new()`: Initialize this vanity.

*Usage:*

`Vanity$new(connect, content, vanity)`

*Arguments:*

`connect` The Connect instance.

`content` The Content instance.

`vanity` The vanity data.

**Method** `get_vanity()`: Return the underlying vanity.

*Usage:*

`Vanity$get_vanity()`

**Method** `print()`: Print this object.

*Usage:*

`Vanity$print(...)`

*Arguments:*

... Unused.

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
Vanity$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### See Also

Other R6 classes: [Bundle](#), [Content](#), [ContentTask](#), [Environment](#), [PositConnect](#), [Task](#), [Variant](#), [VariantSchedule](#), [VariantTask](#)

---

vanity\_is\_available    *Check to see if a vanity URL is currently in use*

---

### Description

[Experimental]

### Usage

```
vanity_is_available(connect, vanity)
```

### Arguments

connect	A Connect R6 object
vanity	string of the vanity URL to check

### Value

logical indicating if the vanity URL is available.

### See Also

Other audit functions: [audit\\_access\\_open\(\)](#), [audit\\_r\\_versions\(\)](#), [audit\\_runas\(\)](#)

---

 Variant

*Variant*


---

**Description**

Variant

Variant

**Details**

An R6 class that represents a Variant

**Super class**`connectapi::Content` -> Variant**Public fields**

key The variant key.

variant The variant.

**Methods****Public methods:**

- `Variant$get_variant()`
- `Variant$get_variant_remote()`
- `Variant$new()`
- `Variant$send_mail()`
- `Variant$get_schedule()`
- `Variant$get_schedule_remote()`
- `Variant$get_subscribers()`
- `Variant$remove_subscriber()`
- `Variant$add_subscribers()`
- `Variant$render()`
- `Variant$renderings()`
- `Variant$update_variant()`
- `Variant$jobs()`
- `Variant$job()`
- `Variant$get_url()`
- `Variant$get_url_rev()`
- `Variant$get_dashboard_url()`
- `Variant$print()`
- `Variant$clone()`

**Method** `get_variant()`: Get the underlying variant data.*Usage:*`Variant$get_variant()`



**Method** `get_variant_remote()`: Get and store the (remote) variant data.

*Usage:*

```
Variant$get_variant_remote()
```

**Method** `new()`: Initialize this variant.

*Usage:*

```
Variant$new(connect, content, key)
```

*Arguments:*

`connect` The Connect instance.

`content` The Content instance.

`key` The variant key.

**Method** `send_mail()`: Mail previously rendered content.

*Usage:*

```
Variant$send_mail(to = c("me", "collaborators", "collaborators_viewers"))
```

*Arguments:*

`to` Targeting.

**Method** `get_schedule()`: Get the (remote) schedule data.

*Usage:*

```
Variant$get_schedule()
```

**Method** `get_schedule_remote()`: Get the (remote) schedule data.

*Usage:*

```
Variant$get_schedule_remote()
```

**Method** `get_subscribers()`: Get the subscribers.

*Usage:*

```
Variant$get_subscribers()
```

**Method** `remove_subscriber()`: Remove a named subscriber.

*Usage:*

```
Variant$remove_subscriber(guid)
```

*Arguments:*

`guid` User GUID.

**Method** `add_subscribers()`: Add named subscribers.

*Usage:*

```
Variant$add_subscribers(guids)
```

*Arguments:*

`guids` User GUIDs.

**Method** `render()`: Render this variant.

*Usage:*

```
Variant$render()
```

**Method** `renderings()`: List the renderings of this variant.

*Usage:*  
Variant\$renderings()

**Method** update\_variant(): Update this variant.

*Usage:*  
Variant\$update\_variant(...)

*Arguments:*  
... Target fields and values.

**Method** jobs(): Jobs for this variant.

*Usage:*  
Variant\$jobs()

**Method** job(): Return single job for this variant.

*Usage:*  
Variant\$job(key)

*Arguments:*  
key The job key.

**Method** get\_url(): Return the URL for this variant.

*Usage:*  
Variant\$get\_url()

**Method** get\_url\_rev(): Return the URL associated with one rendering for this variant.

*Usage:*  
Variant\$get\_url\_rev(rev)

*Arguments:*  
rev Rendering identifier.

**Method** get\_dashboard\_url(): Return the URL for this variant in the Posit Connect dashboard.

*Usage:*  
Variant\$get\_dashboard\_url(pane = "access")

*Arguments:*  
pane The pane in the dashboard to link to.

**Method** print(): Print this object.

*Usage:*  
Variant\$print(...)

*Arguments:*  
... Unused.

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*  
Variant\$clone(deep = FALSE)

*Arguments:*  
deep Whether to make a deep clone.

## See Also

Other R6 classes: [Bundle](#), [Content](#), [ContentTask](#), [Environment](#), [PositConnect](#), [Task](#), [Vanity](#), [VariantSchedule](#), [VariantTask](#)

---

VariantSchedule	<i>VariantSchedule</i>
-----------------	------------------------

---

## Description

VariantSchedule

VariantSchedule

## Details

An R6 class that represents a Schedule

## Super classes

`connectapi::Content` -> `connectapi::Variant` -> VariantSchedule

## Public fields

`schedule_data` The schedule data.

## Methods

### Public methods:

- `VariantSchedule$new()`
- `VariantSchedule$GET()`
- `VariantSchedule$POST()`
- `VariantSchedule$DELETE()`
- `VariantSchedule$set_schedule()`
- `VariantSchedule$is_empty()`
- `VariantSchedule$print()`
- `VariantSchedule$get_schedule()`
- `VariantSchedule$get_schedule_remote()`
- `VariantSchedule$describe_schedule()`
- `VariantSchedule$clone()`

**Method** `new()`: Initialize this schedule.

*Usage:*

```
VariantSchedule$new(connect, content, key, schedule)
```

*Arguments:*

`connect` The Connect instance.

`content` The Content instance.

`key` The variant key.

`schedule` The schedule data.

**Method** `GET()`: Perform an HTTP GET request of the named API path. Returns an object parsed from the HTTP response.

*Usage:*

```
VariantSchedule$GET(path)
```

*Arguments:*

path API path.

**Method POST():** Perform an HTTP POST request of the named API path. Returns an object parsed from the HTTP response.

*Usage:*

VariantSchedule\$POST(path, body)

*Arguments:*

path API path.

body The HTTP payload.

**Method DELETE():** Perform an HTTP DELETE request of the named API path. Returns the HTTP response object.

*Usage:*

VariantSchedule\$DELETE(path)

*Arguments:*

path API path.

**Method set\_schedule():** Set the schedule for this variant

*Usage:*

VariantSchedule\$set\_schedule(...)

*Arguments:*

... Schedule fields.

**Method is\_empty():** Return if this variant has a schedule.

*Usage:*

VariantSchedule\$is\_empty()

**Method print():** Print this object.

*Usage:*

VariantSchedule\$print(...)

*Arguments:*

... Unused.

**Method get\_schedule():** Get the schedule data.

*Usage:*

VariantSchedule\$get\_schedule()

**Method get\_schedule\_remote():** Get and store the (remote) schedule data.

*Usage:*

VariantSchedule\$get\_schedule\_remote()

**Method describe\_schedule():** Description of the associated schedule.

*Usage:*

VariantSchedule\$describe\_schedule()

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

VariantSchedule\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

**See Also**

Other R6 classes: [Bundle](#), [Content](#), [ContentTask](#), [Environment](#), [PositConnect](#), [Task](#), [Vanity](#), [Variant](#), [VariantTask](#)

---

VariantTask

*VariantTask*

---

**Description**

VariantTask

VariantTask

**Details**

An R6 class that represents a Variant Task

**Super classes**

[connectapi::Content](#) -> [connectapi::Variant](#) -> VariantTask

**Public fields**

task The task.

data The variant data.

**Methods****Public methods:**

- [VariantTask\\$new\(\)](#)
- [VariantTask\\$get\\_task\(\)](#)
- [VariantTask\\$add\\_data\(\)](#)
- [VariantTask\\$get\\_data\(\)](#)
- [VariantTask\\$print\(\)](#)
- [VariantTask\\$clone\(\)](#)

**Method** `new()`: Initialize this variant task.

*Usage:*

```
VariantTask$new(connect, content, key, task)
```

*Arguments:*

connect The Connect instance.

content The Content instance.

key The variant key.

task The task data.

**Method** `get_task()`: Return the underlying task.

*Usage:*

```
VariantTask$get_task()
```

**Method** `add_data()`: Set the data.

*Usage:*

```
VariantTask$add_data(data)
```

*Arguments:*

`data` The data.

**Method** `get_data()`: Get the data.

*Usage:*

```
VariantTask$get_data()
```

**Method** `print()`: Print this object.

*Usage:*

```
VariantTask$print(...)
```

*Arguments:*

`...` Unused.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
VariantTask$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

### See Also

Other R6 classes: [Bundle](#), [Content](#), [ContentTask](#), [Environment](#), [PositConnect](#), [Task](#), [Vanity](#), [Variant](#), [VariantSchedule](#)

---

`verify_content_name`     *Verify Content Name*

---

### Description

Ensures that a content name fits the specifications / requirements of Posit Connect. Throws an error if content name is invalid. Content names (as of the time of writing) must be between 3 and 64 alphanumeric characters, dashes, and underscores

### Usage

```
verify_content_name(name)
```

### Arguments

`name`                     The proposed content name

### Value

The name (or an error if invalid)

**See Also**

`connectapi::create_random_name`

Other content functions: [content\\_delete\(\)](#), [content\\_item\(\)](#), [content\\_title\(\)](#), [content\\_update\(\)](#), [create\\_random\\_name\(\)](#), [dashboard\\_url\(\)](#), [dashboard\\_url\\_chr\(\)](#), [delete\\_vanity\\_url\(\)](#), [deploy\\_repo\(\)](#), [get\\_bundles\(\)](#), [get\\_environment\(\)](#), [get\\_image\(\)](#), [get\\_jobs\(\)](#), [get\\_vanity\\_url\(\)](#), [git](#), [permissions](#), [set\\_image\\_path\(\)](#), [set\\_run\\_as\(\)](#), [set\\_vanity\\_url\(\)](#), [swap\\_vanity\\_url\(\)](#)

# Index

## \* R6 classes

- Bundle, 6
- Content, 10
- ContentTask, 15
- Environment, 26
- PositConnect, 48
- Task, 67
- Vanity, 70
- Variant, 72
- VariantSchedule, 75
- VariantTask, 77

## \* audit functions

- audit\_access\_open, 4
- audit\_r\_versions, 5
- audit\_runas, 4
- vanity\_is\_available, 71

## \* content functions

- content\_delete, 16
- content\_item, 17
- content\_title, 19
- content\_update, 19
- create\_random\_name, 20
- dashboard\_url, 21
- dashboard\_url\_chr, 21
- delete\_vanity\_url, 22
- deploy\_repo, 24
- get\_bundles, 28
- get\_environment, 31
- get\_image, 34
- get\_jobs, 35
- get\_vanity\_url, 42
- git, 44
- permissions, 46
- set\_image\_path, 61
- set\_run\_as, 62
- set\_vanity\_url, 65
- swap\_vanity\_url, 66
- verify\_content\_name, 78

## \* deployment functions

- bundle\_dir, 7
- bundle\_path, 7
- bundle\_static, 8
- deploy, 22

- download\_bundle, 25
- poll\_task, 47

## \* schedule functions

- get\_timezones, 37
- get\_variant\_schedule, 44
- set\_schedule, 63

## \* variant functions

- get\_variant\_renderings, 43
- get\_variants, 43

- audit\_access\_open, 4, 4, 5, 71

- audit\_r\_versions, 4, 5, 71

- audit\_runas, 4, 4, 5, 71

- browse\_api\_docs (browse\_solo), 5

- browse\_connect (browse\_solo), 5

- browse\_dashboard (browse\_solo), 5

- browse\_solo, 5

- Bundle, 6, 14, 16, 27, 60, 68, 71, 74, 77, 78

- bundle\_dir, 7, 8, 23, 25, 48

- bundle\_path, 7, 7, 8, 23, 25, 48

- bundle\_static, 7, 8, 8, 23, 25, 48

- Connect (PositConnect), 48

- connect, 9

- connectapi (connectapi-package), 3

- connectapi-package, 3

- connectapi::Content, 15, 26, 70, 72, 75, 77

- connectapi::Variant, 75, 77

- Content, 6, 10, 16, 21, 27, 60, 68, 71, 74, 77, 78

- content\_add\_group (permissions), 46

- content\_add\_user (permissions), 46

- content\_delete, 16, 17, 19–22, 25, 29, 32, 34, 35, 42, 45, 47, 61, 62, 66, 79

- content\_delete\_group (permissions), 46

- content\_delete\_user (permissions), 46

- content\_item, 16, 17, 19–22, 25, 29, 32, 34, 35, 42, 45, 47, 61, 62, 66, 79

- content\_list\_by\_tag, 17

- content\_list\_guid\_has\_access (content\_list\_with\_permissions), 18

- content\_list\_with\_permissions, 18



- content\_title, [16](#), [17](#), [19](#), [20–22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- content\_update, [16](#), [17](#), [19](#), [19](#), [20–22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- content\_update\_access\_type  
(content\_update), [19](#)
- content\_update\_owner (content\_update), [19](#)
- ContentTask, [6](#), [14](#), [15](#), [27](#), [60](#), [68](#), [71](#), [74](#), [77](#), [78](#)
- create\_random\_name, [16](#), [17](#), [19](#), [20](#), [20](#), [21](#), [22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- create\_tag (get\_tags), [36](#)
- create\_tag\_tree (get\_tags), [36](#)
- dashboard\_url, [16](#), [17](#), [19](#), [20](#), [21](#), [22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- dashboard\_url\_chr, [16](#), [17](#), [19–21](#), [21](#), [22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- delete\_bundle (get\_bundles), [28](#)
- delete\_image (get\_image), [34](#)
- delete\_tag (get\_tags), [36](#)
- delete\_vanity\_url, [16](#), [17](#), [19–22](#), [22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- deploy, [7](#), [8](#), [22](#), [25](#), [48](#)
- deploy\_current (deploy), [22](#)
- deploy\_repo, [16](#), [17](#), [19–22](#), [24](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- deploy\_repo\_enable (deploy\_repo), [24](#)
- deploy\_repo\_update (deploy\_repo), [24](#)
- download\_bundle, [7](#), [8](#), [23](#), [25](#), [48](#)
- Environment, [6](#), [14](#), [16](#), [26](#), [60](#), [68](#), [71](#), [74](#), [77](#), [78](#)
- filter\_tag\_tree\_chr (get\_tags), [36](#)
- filter\_tag\_tree\_id (get\_tags), [36](#)
- get\_audit\_logs, [27](#)
- get\_bundles, [16](#), [17](#), [19–22](#), [25](#), [28](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- get\_content, [29](#)
- get\_content(), [4](#), [5](#)
- get\_content\_permissions (permissions), [46](#)
- get\_content\_tags (get\_tags), [36](#)
- get\_environment, [16](#), [17](#), [19–22](#), [25](#), [29](#), [31](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- get\_group\_members, [33](#)
- get\_group\_permission (permissions), [46](#)
- get\_groups, [32](#)
- get\_image, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- get\_job (get\_jobs), [35](#)
- get\_jobs, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- get\_my\_permission (permissions), [46](#)
- get\_procs, [35](#)
- get\_tag\_data (get\_tags), [36](#)
- get\_tags, [36](#)
- get\_timezones, [37](#), [44](#), [65](#)
- get\_usage\_shiny, [38](#)
- get\_usage\_static, [39](#)
- get\_user\_permission (permissions), [46](#)
- get\_users, [41](#)
- get\_vanity\_url, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- get\_variant (get\_variants), [43](#)
- get\_variant\_default (get\_variants), [43](#)
- get\_variant\_renderings, [43](#), [43](#)
- get\_variant\_schedule, [38](#), [44](#), [65](#)
- get\_variants, [43](#), [44](#)
- git, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [44](#), [47](#), [61](#), [62](#), [66](#), [79](#)
- groups\_create\_remote, [45](#)
- has\_image (get\_image), [34](#)
- logical, [65](#)
- page\_cursor, [46](#)
- page\_offset (page\_cursor), [46](#)
- permissions, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#), [34](#), [35](#), [42](#), [45](#), [46](#), [61](#), [62](#), [66](#), [79](#)
- poll\_task, [7](#), [8](#), [23](#), [25](#), [47](#)
- PositConnect, [6](#), [14](#), [16](#), [27](#), [48](#), [68](#), [71](#), [74](#), [77](#), [78](#)
- promote, [60](#)
- repo\_check\_account (git), [44](#)
- repo\_check\_branches (git), [44](#)
- repo\_check\_branches\_ref (git), [44](#)
- repo\_check\_manifest\_dirs (git), [44](#)
- schedule\_describe (set\_schedule), [63](#)
- set\_content\_tag\_tree (get\_tags), [36](#)
- set\_content\_tags (get\_tags), [36](#)
- set\_environment\_all (get\_environment), [31](#)
- set\_environment\_new (get\_environment), [31](#)
- set\_environment\_remove  
(get\_environment), [31](#)

`set_image_path`, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#),  
[34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)  
`set_image_url` (`set_image_path`), [61](#)  
`set_image_webshot` (`set_image_path`), [61](#)  
`set_run_as`, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#), [34](#), [35](#),  
[42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [79](#)  
`set_schedule`, [38](#), [44](#), [63](#)  
`set_schedule_day` (`set_schedule`), [63](#)  
`set_schedule_dayofmonth` (`set_schedule`),  
[63](#)  
`set_schedule_dayofweek` (`set_schedule`),  
[63](#)  
`set_schedule_dayweekofmonth`  
(`set_schedule`), [63](#)  
`set_schedule_hour` (`set_schedule`), [63](#)  
`set_schedule_minute` (`set_schedule`), [63](#)  
`set_schedule_remove` (`set_schedule`), [63](#)  
`set_schedule_semimonth` (`set_schedule`),  
[63](#)  
`set_schedule_week` (`set_schedule`), [63](#)  
`set_schedule_weekday` (`set_schedule`), [63](#)  
`set_schedule_year` (`set_schedule`), [63](#)  
`set_vanity_url`, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#),  
[34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [65](#), [66](#), [79](#)  
`swap_vanity_url`, [16](#), [17](#), [19–22](#), [25](#), [29](#), [32](#),  
[34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [66](#), [79](#)

`Task`, [6](#), [14](#), [16](#), [27](#), [60](#), [67](#), [71](#), [74](#), [77](#), [78](#)  
`tbl_connect`, [68](#)

`user_guid_from_username`, [69](#)  
`users_create_remote`, [69](#)

`Vanity`, [6](#), [14](#), [16](#), [27](#), [60](#), [68](#), [70](#), [74](#), [77](#), [78](#)  
`vanity_is_available`, [4](#), [5](#), [71](#)  
`Variant`, [6](#), [14](#), [16](#), [27](#), [60](#), [68](#), [71](#), [72](#), [77](#), [78](#)  
`variant_render`  
(`get_variant_renderings`), [43](#)  
`VariantSchedule`, [6](#), [14](#), [16](#), [27](#), [60](#), [68](#), [71](#),  
[74](#), [75](#), [78](#)  
`VariantTask`, [6](#), [14](#), [16](#), [27](#), [60](#), [68](#), [71](#), [74](#), [77](#),  
[77](#)  
`verify_content_name`, [16](#), [17](#), [19–22](#), [25](#), [29](#),  
[32](#), [34](#), [35](#), [42](#), [45](#), [47](#), [61](#), [62](#), [66](#), [78](#)

`webshot2::webshot()`, [61](#)