

# Package ‘arctgisutils’

September 3, 2026

**Title** R-ArcGIS Bridge Utility Functions

**Version** 0.6.1

**Description** Developer oriented utility functions designed to be used as the building blocks of R packages that work with ArcGIS Location Services. It provides functionality for authorization, Esri JSON construction and parsing, as well as other utilities pertaining to geometry and Esri type conversions. To support 'ArcGIS Pro' users, authorization can be done via 'arctgisbinding'. Installation instructions for 'arctgisbinding' can be found at <https://developers.arcgis.com/r-bridge/installation/>.

**License** Apache License (>= 2)

**URL** <https://github.com/R-ArcGIS/arctgisutils>,  
<https://developers.arcgis.com/r-bridge/api-reference/arctgisutils/>

**BugReports** <https://github.com/R-ArcGIS/arctgisutils/issues>

**Depends** R (>= 4.2)

**Imports** cli, httr2 (>= 1.0.5), R6, RcppSimdJson, rlang, S7, sf, utils, yjjsonr, lifecycle

**Suggests** arctgisbinding, collapse (>= 2.0.0), data.table, jsonify, testthat (>= 3.0.0), vctrs, curl, shinyOAuth

**Config/rextendr/version** 0.4.2

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en

**SystemRequirements** Cargo (Rust's package manager), rustc >= 1.67, xz

**Config/roxygen2/version** 8.1.0

**NeedsCompilation** yes

**Author** Josiah Parry [aut, cre] (ORCID: <https://orcid.org/0000-0001-9910-865X>),  
Kenneth Vernon [ctb] (ORCID: <https://orcid.org/0000-0003-0098-5092>),  
Martha Bass [ctb] (ORCID: <https://orcid.org/0009-0004-0268-5426>),

Eli Pousson [ctb] (ORCID: <<https://orcid.org/0000-0001-8280-1706>>),  
 Ryan Zomorodi [ctb] (ORCID: <<https://orcid.org/0009-0003-6417-5985>>)

**Maintainer** Josiah Parry <josiah.parry@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-09-03 05:21:01 UTC

## Contents

|                                   |    |
|-----------------------------------|----|
| arc_agent . . . . .               | 3  |
| arc_base_req . . . . .            | 3  |
| arc_form_params . . . . .         | 4  |
| arc_gp_job . . . . .              | 5  |
| arc_group . . . . .               | 7  |
| arc_group_users . . . . .         | 8  |
| arc_host . . . . .                | 9  |
| arc_item . . . . .                | 10 |
| arc_item_data . . . . .           | 11 |
| arc_job_status . . . . .          | 12 |
| arc_paginate_req . . . . .        | 12 |
| arc_portal_resources . . . . .    | 13 |
| arc_portal_servers . . . . .      | 14 |
| arc_portal_urls . . . . .         | 15 |
| arc_portal_users . . . . .        | 15 |
| arc_token . . . . .               | 17 |
| arc_user . . . . .                | 19 |
| arc_user_self . . . . .           | 19 |
| as_esri_geometry . . . . .        | 20 |
| as_extent . . . . .               | 22 |
| as_features . . . . .             | 23 |
| as_featureset . . . . .           | 25 |
| as_fields . . . . .               | 27 |
| as_layer . . . . .                | 29 |
| auth_code . . . . .               | 32 |
| auth_shiny . . . . .              | 34 |
| compact . . . . .                 | 36 |
| content . . . . .                 | 37 |
| detect_errors . . . . .           | 38 |
| determine_dims . . . . .          | 39 |
| determine_esri_geo_type . . . . . | 40 |
| fetch_layer_metadata . . . . .    | 41 |
| gp_job_from_url . . . . .         | 42 |
| gp_params . . . . .               | 42 |
| is_date . . . . .                 | 46 |
| opts . . . . .                    | 47 |
| parse_esri_json . . . . .         | 47 |
| portal_types . . . . .            | 48 |
| rbind_results . . . . .           | 49 |

arc\_agent

3

search\_items

50

self

53

url

54

validate\_crs

55

Index

57

|           |                                       |
|-----------|---------------------------------------|
| arc_agent | <i>Set user-agent for arcgisutils</i> |
|-----------|---------------------------------------|

**Description**

Override the default user-agent set by httr2 to indicate that a request came from arcgisutils.

**Usage**

```
arc_agent(req)
```

**Arguments**

req                    an httr2 request

**Value**

an httr2 request object

**Examples**

```
req <- httr2::request("http://example.com")
arc_agent(req)
```

|              |                              |
|--------------|------------------------------|
| arc_base_req | <i>Generate base request</i> |
|--------------|------------------------------|

**Description**

This function takes a url and creates a basic httr2 request that adds the user-agent and adds an authorization token to the X-Esri-Authorization header.

**Usage**

```
arc_base_req(
  url,
  token = NULL,
  path = NULL,
  query = NULL,
  error_call = rlang::caller_env()
)
```

Arguments

|            |                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------|
| url        | a valid url that is passed to <code>httr2::request()</code>                                                          |
| token      | an object of class <code>httr2_token</code> as generated by <code>auth_code()</code> or related function             |
| path       | a character vector of paths to be appended to url using <code>httr2::req_url_path_append()</code>                    |
| query      | a named vector or named list of query parameters to be appended to the url using <code>httr2::req_url_query()</code> |
| error_call | the caller environment to be used when propagating errors.                                                           |

Value

an `httr2_request` with the `X-Esri-Authorization` header and `User-Agent` set.

Examples

```
arc_base_req("https://arcgis.com")
```

---

|                 |                                |
|-----------------|--------------------------------|
| arc_form_params | <i>Form request parameters</i> |
|-----------------|--------------------------------|

---

Description

ArcGIS endpoints make extensive use of form encoded data for the body of http requests. Form requests require that each element has a name and is encoded as a single string—often as json.

Usage

```
arc_form_params(params = list())  
  
as_form_params(x)
```

Arguments

|        |                                                                                |
|--------|--------------------------------------------------------------------------------|
| params | a named list with scalar character elements                                    |
| x      | for <code>as_form_params()</code> , a named list to convert to form parameters |

Details

The `arc_form_params` class provides validation of form body parameters ensuring that each element is a scalar string. It uses a named list internally to store the parameters.

The helper function `as_form_params()` converts a named list to form parameters by automatically JSON-encoding each element using `yyjsonr::write_json_str()` with `auto_unbox = TRUE`.

Value

an object of class `arc_form_params`

See Also

Other geoprocessing: [arc\\_gp\\_job](#), [arc\\_job\\_status\(\)](#), [gp\\_job\\_from\\_url\(\)](#), [gp\\_params](#)

Examples

```
arc_form_params(  
    list(f = "json", outFields = "*", where = "1 = 1")  
)
```

---

|            |                                    |
|------------|------------------------------------|
| arc_gp_job | Create a Geoprocessing Service Job |
|------------|------------------------------------|

---

Description

The `arc_gp_job` class is used to interact with Geoprocessing Services in ArcGIS Online and Enterprise.

Usage

```
new_gp_job(base_url, params = list(), token = arc_token())
```

Arguments

- `base_url` the URL of the job service (without `/submitJob`)
- `params` a named list where each element is a scalar character
- `token` default [arc\\_token\(\)](#). The token to be used with the job.

Details

The `arc_gp_job` uses S7 classes for the job request parameters and job status via [arc\\_form\\_params\(\)](#) and [arc\\_job\\_status\(\)](#) respectively. Importantly, [arc\\_form\\_params\(\)](#) ensures that parameters provided to a geoprocessing service are all character scalars as required by the form body.

Value

An object of class `arc_gp_job`.

Associated Functions

```
from_url(url, token = arc_token()) Create a GP Job object from an existing job URL
```

Public fields

- `base_url` the URL of the job service (without `/submitJob`)
- `id` the ID of the started job. NULL `self$start()` has not been called.

## Active bindings

`params` returns an S7 object of class `arc_form_params` (see [arc\\_form\\_params\(\)](#)) the list can be accessed via `self$params@params`.

`status` returns the status of the geoprocessing job as an S7 object of class `gp_job_status` (see [arc\\_job\\_status\(\)](#)) by querying the `/jobs/{job-id}` endpoint.

`results` returns the current results of the job by querying the `/jobs/{job-id}/results` endpoint.

## Methods

### Public methods:

- [arc\\_gp\\_job\\$new\(\)](#)
- [arc\\_gp\\_job\\$start\(\)](#)
- [arc\\_gp\\_job\\$cancel\(\)](#)
- [arc\\_gp\\_job\\$await\(\)](#)
- [arc\\_gp\\_job\\$messages\(\)](#)
- [arc\\_gp\\_job\\$clone\(\)](#)

`arc_gp_job$new()`:

*Usage:*

```
arc_gp_job$new(
  base_url,
  params = list(),
  result_fn = NULL,
  token = arc_token(),
  error_call = rlang::caller_call()
)
```

*Arguments:*

`base_url` the URL of the job service (without `/submitJob`)

`params` a named list where each element is a scalar character

`result_fn` Default NULL. An optional function to apply to the results JSON. By default parses results using `RcppSimdJson::fparse()`.

`token` default [arc\\_token\(\)](#). The token to be used with the job.

`error_call` default `rlang::caller_call()` the calling environment.

`arc_gp_job$start()`: Starts the job by calling the `/submitJob` endpoint. This also sets the public field `id`.

*Usage:*

```
arc_gp_job$start()
```

`arc_gp_job$cancel()`: Cancels a job by calling the `/cancel` endpoint.

*Usage:*

```
arc_gp_job$cancel()
```

`arc_gp_job$await()`: Waits for job completion and returns results.

*Usage:*

```
arc_gp_job$await(interval = 0.1, verbose = FALSE)
```

*Arguments:*

interval polling interval in seconds (default 0.1)

verbose whether to print status messages (default FALSE)

arc\_gp\_job\$messages(): Fetches Geoprocessing Messages

*Usage:*

```
arc_gp_job$messages()
```

arc\_gp\_job\$clone(): The objects of this class are cloneable with this method.

*Usage:*

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

*Arguments:*

deep Whether to make a deep clone.

**See Also**

Other geoprocessing: [arc\\_form\\_params\(\)](#), [arc\\_job\\_status\(\)](#), [gp\\_job\\_from\\_url\(\)](#), [gp\\_params](#)

**Examples**

```
url <- paste0(
  "https://logistics.arcgis.com/arcgis/",
  "rest/services/World/ServiceAreas/",
  "GPService/GenerateServiceAreas"
)
job <- new_gp_job(url, list(f = "json"))
job

# extract params S7 class
params <- job$params
params

# view underlying list
params@params
```

---

arc\_group

*Fetch Group Information*


---

**Description**

Fetches metadata about a group based on a provided group\_id.

**Usage**

```
arc_group(group_id, host = arc_host(), token = arc_token())
```

Arguments

- group\_id            the unique group identifier. A scalar character.
- host                default "https://www.arcgis.com". The host of your ArcGIS Portal.
- token               an http2\_token as created by auth\_code() or similar

Details

[Experimental]

Value

a list with group metadata

See Also

Other portal organization: [arc\\_user\(\)](#)

Examples

```
arc_group("2f0ec8cb03574128bd673cefab106f39")
```

---

|                 |                              |
|-----------------|------------------------------|
| arc_group_users | <i>List users in a group</i> |
|-----------------|------------------------------|

---

Description

List all users in a given group.

Usage

```
arc_group_users(  
  group,  
  name = NULL,  
  member_type = NULL,  
  joined_before = NULL,  
  joined_after = NULL,  
  sort_field = c("username", "membertype", "joined"),  
  sort_order = c("asc", "desc"),  
  page_size = 50,  
  max_pages = Inf,  
  .progress = TRUE,  
  host = arc_host(),  
  token = arc_token()  
)
```

Arguments

|               |                                                                                                 |
|---------------|-------------------------------------------------------------------------------------------------|
| group         | a group ID as a scalar string or PortalGroup created via <a href="#">arc_group()</a>            |
| name          | a scalar string of a user to search for.                                                        |
| member_type   | default NULL. If provided must be one of "admin" or "member".                                   |
| joined_before | default NULL. A scalar date to search for users who joined before this date.                    |
| joined_after  | default NULL. A scalar date to search for users who joined after this date.                     |
| sort_field    | default "username". The field to sort by. Must be one of "username", "membertype", or "joined". |
| sort_order    | optional string. One of either asc or desc for ascending or descending order respectively.      |
| page_size     | a scalar integer between 1 and 100 indicating the number of responses per page.                 |
| max_pages     | the maximum number of pages to fetch. By default fetches all pages.                             |
| .progress     | default TRUE. Whether to display a progress bar for requests.                                   |
| host          | default "https://www.arcgis.com". The host of your ArcGIS Portal.                               |
| token         | an object of class httr2_token as generated by <a href="#">auth_code()</a> or related function  |

Value

a data.frame

References

[API Reference](#)

Examples

```
## Not run:
set_arc_token(auth_user())
groups <- arc_user("r-bridge-docs")$groups
arc_group_users(groups$id[1])

## End(Not run)
```

---

|          |                               |
|----------|-------------------------------|
| arc_host | <i>Determines Portal Host</i> |
|----------|-------------------------------|

---

Description

Returns a scalar character indicating the host to make requests to.

Usage

```
arc_host()
```

**Details**

By default, the host is ArcGIS Online <<https://www.arcgis.com>>. If the environment variable ARCGIS\_HOST is set, it will be returned.

**Value**

A scalar character, "https://www.arcgis.com" by default.

**Examples**

```
arc_host()
```

---

|          |                             |
|----------|-----------------------------|
| arc_item | <i>Portal Item Metadata</i> |
|----------|-----------------------------|

---

**Description**

Given the unique ID of a content item, fetches the item's metadata from a portal.

**Usage**

```
arc_item(item_id, host = arc_host(), token = arc_token())
```

**Arguments**

|         |                                                                   |
|---------|-------------------------------------------------------------------|
| item_id | the ID of the item to fetch. A scalar character.                  |
| host    | default "https://www.arcgis.com". The host of your ArcGIS Portal. |
| token   | an http2_token as created by auth_code() or similar               |

**Details**

See [API Reference](#) for more information.

**[Experimental]**

**Value**

an object of class PortalItem which is a list with the item's metadata.

**See Also**

Other portal item: [arc\\_item\\_data\(\)](#)

**Examples**

```
arc_item("9df5e769bfe8412b8de36a2e618c7672")
```

---

|               |                                |
|---------------|--------------------------------|
| arc_item_data | <i>Download an Item's Data</i> |
|---------------|--------------------------------|

---

## Description

Download the data backing a portal item. This function always returns a raw vector as the type of the data that is downloaded cannot always be known.

## Usage

```
arc_item_data(item, host = arc_host(), token = arc_token())
```

## Arguments

|       |                                                                               |
|-------|-------------------------------------------------------------------------------|
| item  | the item ID or the result of <code>arc_item()</code> .                        |
| host  | default "https://www.arcgis.com". The host of your ArcGIS Portal.             |
| token | an <code>httr2_token</code> as created by <code>auth_code()</code> or similar |

## Details

[Experimental]

## Value

a raw vector containing the bytes of the data associated with the item. If the response is `application/json` then the json string is returned without parsing.

## See Also

Other portal item: [arc\\_item\(\)](#)

## Examples

```
arc_item_data("9df5e769bfe8412b8de36a2e618c7672")
```

---

|                |                                 |
|----------------|---------------------------------|
| arc_job_status | <i>Geoprocessing Job Status</i> |
|----------------|---------------------------------|

---

**Description**

Represents the status of a geoprocessing job.

**Usage**

```
arc_job_status(status = character(0))
```

**Arguments**

status            a scalar character. Must be one of "esriJobSubmitted", "esriJobWaiting", "esriJobExecuting", "esriJobSucceeded", "esriJobFailed", "esriJobTimedOut", "esriJobCancelling", or "esriJobCancelled".

**Value**

an object of class arc\_job\_status

**See Also**

Other geoprocessing: [arc\\_form\\_params\(\)](#), [arc\\_gp\\_job](#), [gp\\_job\\_from\\_url\(\)](#), [gp\\_params](#)

**Examples**

```
arc_job_status("esriJobSubmitted")
```

---

|                  |                                 |
|------------------|---------------------------------|
| arc_paginate_req | <i>Paginate ArcGIS Requests</i> |
|------------------|---------------------------------|

---

**Description**

Many API endpoints provide common **pagination properties**. `arc_paginate_request()` automatically applies pagination to an input request.

**Usage**

```
arc_paginate_req(req, page_size = 10, max_pages = Inf, .progress = TRUE)
```

**Arguments**

req            an http2\_request ideally created with arc\_base\_req  
page\_size    a scalar integer between 1 and 100 indicating the number of responses per page.  
max\_pages    the maximum number of pages to fetch. By default fetches all pages.  
.progress    default TRUE. Whether to display a progress bar for requests.

**Value**

a list of `httr2_response`.

**References**

[API Documentation](#)

**See Also**

[arc\\_base\\_req\(\)](#)

---

arc\_portal\_resources    *Portal File Resources*

---

**Description**

The resources endpoint lists all file resources for the organization.

**Usage**

```
arc_portal_resources(  
  id = arc_portal_self(token)[["id"]],  
  page_size = 50,  
  max_pages = Inf,  
  .progress = TRUE,  
  host = arc_host(),  
  token = arc_token()  
)
```

**Arguments**

|                        |                                                                                                             |
|------------------------|-------------------------------------------------------------------------------------------------------------|
| <code>id</code>        | the portal ID. By default it fetches the <code>id</code> from <a href="#">arc_portal_self()</a> .           |
| <code>page_size</code> | a scalar integer between 1 and 100 indicating the number of responses per page.                             |
| <code>max_pages</code> | the maximum number of pages to fetch. By default fetches all pages.                                         |
| <code>.progress</code> | default <code>TRUE</code> . Whether to display a progress bar for requests.                                 |
| <code>host</code>      | default <code>"https://www.arcgis.com"</code> . The host of your ArcGIS Portal.                             |
| <code>token</code>     | an object of class <code>httr2_token</code> as generated by <a href="#">auth_code()</a> or related function |

**Value**

a `data.frame` of resources available to your portal.

**References**

[API Reference](#)

**See Also**

Other portal: [arc\\_portal\\_urls\(\)](#), [arc\\_portal\\_users\(\)](#), [self](#)

**Examples**

```
## Not run:
  set_arc_token(auth_user())
  arc_portal_resources()

## End(Not run)
```

---

|                    |                                       |
|--------------------|---------------------------------------|
| arc_portal_servers | <i>List ArcGIS Enterprise Servers</i> |
|--------------------|---------------------------------------|

---

**Description**

The servers resource lists the ArcGIS Server sites that have been federated with the portal.

**Usage**

```
arc_portal_servers(
  id = arc_portal_self(token)[["id"]],
  host = arc_host(),
  token = arc_token()
)
```

**Arguments**

|       |                                                                                                |
|-------|------------------------------------------------------------------------------------------------|
| id    | the portal ID. By default it fetches the id from <a href="#">arc_portal_self()</a> .           |
| host  | default "https://www.arcgis.com". The host of your ArcGIS Portal.                              |
| token | an object of class httr2_token as generated by <a href="#">auth_code()</a> or related function |

**Value**

a data.frame of servers

**Examples**

```
## Not run:
  set_arc_token(auth_user())
  arc_portal_servers()

## End(Not run)
```

---

|                 |                            |
|-----------------|----------------------------|
| arc_portal_urls | <i>Organization's URLs</i> |
|-----------------|----------------------------|

---

**Description**

Returns the URLs of an organizations services.

**Usage**

```
arc_portal_urls(host = arc_host(), token = arc_token())
```

**Arguments**

- host                    default "https://www.arcgis.com". The host of your ArcGIS Portal.
- token                  an httr2\_token as created by auth\_code() or similar

**Details**

See [API Reference](#) for more information. **[Experimental]**

**Value**

a data.frame

**See Also**

Other portal: [arc\\_portal\\_resources\(\)](#), [arc\\_portal\\_users\(\)](#), [self](#)

**Examples**

```
arc_portal_urls()
```

---

|                  |                     |
|------------------|---------------------|
| arc_portal_users | <i>Portal Users</i> |
|------------------|---------------------|

---

**Description**

This function lists all users in a portal.

**Usage**

```

arc_portal_users(
  id = arc_portal_self(token)[["id"]],
  sort_field = NULL,
  provider = NULL,
  sort_order = NULL,
  role = NULL,
  fullname = NULL,
  username = NULL,
  firstname = NULL,
  lastname = NULL,
  filter_intersection = NULL,
  page_size = 50,
  max_pages = Inf,
  .progress = TRUE,
  host = arc_host(),
  token = arc_token()
)

```

**Arguments**

|                     |                                                                                                                                            |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| id                  | the portal ID. By default it fetches the id from <a href="#">arc_portal_self()</a> .                                                       |
| sort_field          | optional field to sort by. It must be one of "username", "fullname", "created", "lastlogin", "mfaenabled", "level", "role".                |
| provider            | optional filter users based on their identity provider. Must be one of "arcgis", "enterprise", "facebook", "google", "apple", or "github". |
| sort_order          | optional order to sort by. It must be one of "asc" or "desc".                                                                              |
| role                | optional role to filter down to. It must be one of "org_admin", "org_publisher", "org_user".                                               |
| fullname            | optional string of the user's fullanme to search for.                                                                                      |
| username            | optional string of the user's user name to search for.                                                                                     |
| firstname           | optional string of the user's first name to search for.                                                                                    |
| lastname            | optional string of the user's last name to search for.                                                                                     |
| filter_intersection | optional boolean value. If TRUE mutiple filters are treated as an "and" condition. If FALSE, treated as an "or".                           |
| page_size           | a scalar integer between 1 and 100 indicating the number of responses per page.                                                            |
| max_pages           | the maximum number of pages to fetch. By default fetches all pages.                                                                        |
| .progress           | default TRUE. Whether to display a progress bar for requests.                                                                              |
| host                | default "https://www.arcgis.com". The host of your ArcGIS Portal.                                                                          |
| token               | an object of class httr2_token as generated by <a href="#">auth_code()</a> or related function                                             |

**Value**

a data.frame of users.

**References**

[API Reference](#)

**See Also**

Other portal: [arc\\_portal\\_resources\(\)](#), [arc\\_portal\\_urls\(\)](#), [self](#)

**Examples**

```
## Not run:
set_arc_token(auth_user())
arc_portal_users()

## End(Not run)
```

---

arc\_token

*Manage authorization tokens*


---

**Description**

These functions are used to set, fetch, and check authorization tokens.

**Usage**

```
arc_token(token = "ARCGIS_TOKEN")

set_arc_token(token, ...)

unset_arc_token(token = NULL)

obj_check_token(token, call = rlang::caller_env())

check_token_has_user(token, call = rlang::caller_env())
```

**Arguments**

|       |                                                                                                                                                                                                                                         |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| token | for <code>arc_token()</code> , the name of a token to fetch. For <code>set_arc_token()</code> , it is an <code>httr2_token</code> that will be set. For <code>unset_arc_token()</code> , a character vector of token names to be unset. |
| ...   | named arguments to set <code>httr2_token</code> . Must be valid names and must be an <code>httr2_token</code> .                                                                                                                         |

`call`            The execution environment of a currently running function, e.g. `call = caller_env()`. The corresponding function call is retrieved and mentioned in error messages as the source of the error.

You only need to supply `call` when throwing a condition from a helper function which wouldn't be relevant to mention in the message.

Can also be `NULL` or a [defused function call](#) to respectively not display any call or hard-code a code to display.

For more information about error calls, see [Including function calls in error messages](#).

## Details

It is possible to have multiple authorization tokens in one session. These functions assist you in managing them.

`arc_token()` is used to fetch tokens by name. The default token is `ARCGIS_TOKEN`. However, they can be any valid character scalar. `set_arc_token()` will create store a token with the name `ARCGIS_TOKEN`. However, you can alternatively set the tokens by name using a key-value pair. The key is what you would pass to `arc_token()` to fetch the `httr2_token` object. To remove a token that has been set, use `unset_arc_token()`.

`obj_check_token()` is a developer oriented function that can be used to check if an object is indeed an `httr2_token`. To check if a token has expired, [validate\\_or\\_refresh\\_token\(\)](#) will do so.

`check_token_has_user()` is a developer oriented function that checks to see if a token has a `username` field associated with it.

For developers:

`set_arc_token()` uses a package level environment to store the tokens. The tokens are fetched from the environment using `arc_token()`.

## Examples

```
# create fake tokens
token_a <- httr2::oauth_token("1234", arcgis_host = arc_host())
token_b <- httr2::oauth_token("abcd", arcgis_host = arc_host())

# set token to the default location
set_arc_token(token_a)

# fetch token from the default location
arc_token()

# set token by name
set_arc_token(org_a = token_a, org_b = token_b)

# fetch token by name
arc_token("org_a")
arc_token("org_b")

# unset tokens
unset_arc_token()
unset_arc_token(c("org_a", "org_b"))
```

---

|          |                         |
|----------|-------------------------|
| arc_user | <i>User Information</i> |
|----------|-------------------------|

---

**Description**

Fetch a user’s metadata based on username.

**Usage**

arc\_user(username, host = arc\_host(), token = arc\_token())

**Arguments**

- username            the username to fetch. A scalar character.
- host                default "https://www.arcgis.com". The host of your ArcGIS Portal.
- token               an http2\_token as created by auth\_code() or similar

**Details**

**[Experimental]**

**Value**

a list of class PortalUser

**See Also**

Other portal organization: [arc\\_group\(\)](#)

**Examples**

```
arc_user("esri_en")
```

---

|               |                                             |
|---------------|---------------------------------------------|
| arc_user_self | <i>Discover Authenticated User Metadata</i> |
|---------------|---------------------------------------------|

---

**Description**

Given an authentication token, return a list of user-specific information such as the user ID, user-name, available credits, email, groups, last login date and more.

**Usage**

```
arc_user_self(
  host = arc_host(),
  token = arc_token(),
  error_call = rlang::caller_call()
)
```

**Arguments**

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| host       | default "https://www.arcgis.com". The host of your ArcGIS Portal. |
| token      | an http2_token as created by auth_code() or similar               |
| error_call | the caller environment to be used when propagating errors.        |

**Value**

a list of the authenticated user's metadata

**References**

[API Reference](#)

**Examples**

```
## Not run:
if (interactive()) {
  arc_user_self(token = auth_user())
}

## End(Not run)
```

---

as\_esri\_geometry

---

*Create Esri JSON Geometry Objects*


---

**Description**

as\_esri\_geometry() converts an sfg object to a EsriJSON Geometry object as a string.

**Usage**

```
as_esri_geometry(x, crs = NULL, call = rlang::caller_env())
```

**Arguments**

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | an object of class sfg. Must be one of "POINT", "MULTIPOINT", "LINESTRING", "MULTILINESTRING", "POLYGON", or "MULTIPOLYGON".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| crs  | the coordinate reference system. It must be interpretable by <code>sf::st_crs()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| call | <p>The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be <code>NULL</code> or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |

**Details**

See `as_featureset()` and `as_features()` for converting sfc and sf objects into EsriJSON.

**Value**

a scalar string

**References**

[API Reference](#)

**Examples**

```
library(sf)
# POINT
# create sfg points
xy <- st_point(c(1, 2))
xyz <- st_point(c(1, 2, 3))
xym <- st_point(c(1, 2, 3), dim = "XYM")
xyzm <- st_point(c(1, 2, 3, 4))

as_esri_geometry(xy)
as_esri_geometry(xyz)
as_esri_geometry(xym)
as_esri_geometry(xyzm)

# MULTIPOINT
# vector to create matrix points
set.seed(0)
x <- rnorm(12)

xy <- st_multipoint(matrix(x, ncol = 2))
xyz <- st_multipoint(matrix(x, ncol = 3))
xym <- st_multipoint(matrix(x, ncol = 3), dim = "XYM")
xyzm <- st_multipoint(matrix(x, ncol = 4), dim = "XYM")
```

```

as_esri_geometry(xy)
as_esri_geometry(xyz)
as_esri_geometry(xym)
as_esri_geometry(xyzm)

# LINESTRING
xy <- st_linestring(matrix(x, ncol = 2))
xyz <- st_linestring(matrix(x, ncol = 3))
xym <- st_linestring(matrix(x, ncol = 3), dim = "XYM")
xyzm <- st_linestring(matrix(x, ncol = 4), dim = "XYM")

as_esri_geometry(xy)
as_esri_geometry(xyz)
as_esri_geometry(xym)
as_esri_geometry(xyzm)

# MULTILINESTRING
as_esri_geometry(st_multilinestring(list(xy, xy)))
as_esri_geometry(st_multilinestring(list(xyz, xyz)))
as_esri_geometry(st_multilinestring(list(xym, xym)))
as_esri_geometry(st_multilinestring(list(xyzm, xyzm)))

# POLYGON
coords <- rbind(
  c(0, 0, 0, 1),
  c(0, 1, 0, 1),
  c(1, 1, 1, 1),
  c(1, 0, 1, 1),
  c(0, 0, 0, 1)
)

xy <- st_polygon(list(coords[, 1:2]))
xyz <- st_polygon(list(coords[, 1:3]))
xym <- st_polygon(list(coords[, 1:3]), dim = "XYM")
xyzm <- st_polygon(list(coords))

as_esri_geometry(xy)
as_esri_geometry(xyz)
as_esri_geometry(xym)
as_esri_geometry(xyzm)

# MULTIPOLYGON
as_esri_geometry(st_multipolygon(list(xy, xy)))
as_esri_geometry(st_multipolygon(list(xyz, xyz)))
as_esri_geometry(st_multipolygon(list(xym, xym)))
as_esri_geometry(st_multipolygon(list(xyzm, xyzm)))

```

**Description**

Given an sf or sfc object create a list that represents the extent of the object. The result of this function can be parsed directly into json using `jsonify::to_json(x, unbox = TRUE)` or included into a list as the extent component that will be eventually converted into json using the above function.

**Usage**

```
as_extent(x, crs = sf::st_crs(x), call = rlang::caller_env())
```

**Arguments**

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | an sf or sfc object                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| crs  | the CRS of the object. Must be parsable by <code>sf::st_crs()</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| call | <p>The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply call when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be NULL or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |

**Value**

An extent json object. Use `jsonify::to_json(x, unbox = TRUE)` to convert to json.

**Examples**

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
as_extent(nc)
```

---

as\_features

---

*Create Esri Features*


---

**Description**

These functions create an array of Esri Feature objects. Each feature consists of a geometry and attribute field. The result of `as_esri_features()` is a JSON array of Features whereas `as_features()` is a list that represents the same JSON array. Using `jsonify::to_json(as_features(x), unbox = TRUE)` will result in the same JSON array.

**Usage**

```
as_features(x, crs = sf::st_crs(x), call = rlang::caller_env())
```

```
as_esri_features(x, crs = sf::st_crs(x), call = rlang::caller_env())
```

**Arguments**

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>    | an object of class <code>sf</code> , <code>data.frame</code> , or <code>sfc</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>crs</code>  | the coordinate reference system. It must be interpretable by <code>sf::st_crs()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>call</code> | <p>The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be <code>NULL</code> or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |

**Value**

Either a scalar string or a named list.

**References**

[API Reference](#)

**Examples**

```
library(sf)
# POINT
# create sfg points
xy <- st_sfc(st_point(c(1, 2)))
xyz <- st_sfc(st_point(c(1, 2, 3)))
xym <- st_sfc(st_point(c(1, 2, 3), dim = "XYM"))

as_esri_features(xy)
as_esri_features(xyz)
as_esri_features(xym)

# MULTIPOINT
# vector to create matrix points
set.seed(0)
x <- rnorm(12)

xy <- st_sfc(st_multipoint(matrix(x, ncol = 2)))
xyz <- st_sfc(st_multipoint(matrix(x, ncol = 3)))
xym <- st_sfc(st_multipoint(matrix(x, ncol = 3), dim = "XYM"))

as_esri_features(xy)
as_esri_features(xyz)
as_esri_features(xym)

# LINESTRING
xy <- st_sfc(st_linestring(matrix(x, ncol = 2)))
xyz <- st_sfc(st_linestring(matrix(x, ncol = 3)))
```

```

xym <- st_sfc(st_linestring(matrix(x, ncol = 3), dim = "XYM"))

as_esri_features(xy)
as_esri_features(xyz)
as_esri_features(xym)

# MULTILINESTRING
as_esri_features(st_sfc(st_multilinestring(list(xy[[1]], xy[[1]]))))
as_esri_features(st_sfc(st_multilinestring(list(xyz[[1]], xyz[[1]]))))
as_esri_features(st_sfc(st_multilinestring(list(xym[[1]], xym[[1]]))))

# POLYGON
coords <- rbind(
  c(0, 0, 0, 1),
  c(0, 1, 0, 1),
  c(1, 1, 1, 1),
  c(1, 0, 1, 1),
  c(0, 0, 0, 1)
)

xy <- st_sfc(st_polygon(list(coords[, 1:2])))
xyz <- st_sfc(st_polygon(list(coords[, 1:3])))
xym <- st_sfc(st_polygon(list(coords[, 1:3]), dim = "XYM"))

as_esri_features(xy)
as_esri_features(xyz)
as_esri_features(xym)

# MULTIPOLYGON
as_esri_features(st_sfc(st_multipolygon(list(xy[[1]], xy[[1]]))))
as_esri_features(st_sfc(st_multipolygon(list(xyz[[1]], xyz[[1]]))))
as_esri_features(st_sfc(st_multipolygon(list(xym[[1]], xym[[1]]))))

```

---

as\_featureset

Create Esri FeatureSet Objects

---

## Description

These functions create an Esri FeatureSet object. A FeatureSet contains an inner array of features as well as additional metadata about the the collection such as the geometry type, spatial reference, and object ID field.

## Usage

```
as_featureset(x, crs = sf::st_crs(x), call = rlang::caller_env())
```

```
as_esri_featureset(x, crs = sf::st_crs(x), call = rlang::caller_env())
```

**Arguments**

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>    | an object of class <code>sf</code> , <code>data.frame</code> , or <code>sfc</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>crs</code>  | the coordinate reference system. It must be interpretable by <code>sf::st_crs()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>call</code> | <p>The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be <code>NULL</code> or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |

**Value**

a list or a json string

**References**

[API Reference](#)

**Examples**

```
library(sf)
# POINT
# create sfg points
xy <- st_sfc(st_point(c(1, 2)))
xyz <- st_sfc(st_point(c(1, 2, 3)))
xym <- st_sfc(st_point(c(1, 2, 3), dim = "XYM"))

as_esri_featureset(xy)
as_esri_featureset(xyz)
as_esri_featureset(xym)

# MULTIPOINT
# vector to create matrix points
set.seed(0)
x <- rnorm(12)

xy <- st_sfc(st_multipoint(matrix(x, ncol = 2)))
xyz <- st_sfc(st_multipoint(matrix(x, ncol = 3)))
xym <- st_sfc(st_multipoint(matrix(x, ncol = 3), dim = "XYM"))

as_esri_featureset(xy)
as_esri_featureset(xyz)
as_esri_featureset(xym)

# LINESTRING
xy <- st_sfc(st_linestring(matrix(x, ncol = 2)))
xyz <- st_sfc(st_linestring(matrix(x, ncol = 3)))
```

```

xym <- st_sfc(st_linestring(matrix(x, ncol = 3), dim = "XYM"))

as_esri_featureset(xy)
as_esri_featureset(xyz)
as_esri_featureset(xym)

# MULTILINESTRING
as_esri_featureset(st_sfc(st_multilinestring(list(xy[[1]], xy[[1]]))))
as_esri_featureset(st_sfc(st_multilinestring(list(xyz[[1]], xyz[[1]]))))
as_esri_featureset(st_sfc(st_multilinestring(list(xym[[1]], xym[[1]]))))

# POLYGON
coords <- rbind(
  c(0, 0, 0, 1),
  c(0, 1, 0, 1),
  c(1, 1, 1, 1),
  c(1, 0, 1, 1),
  c(0, 0, 0, 1)
)

xy <- st_sfc(st_polygon(list(coords[, 1:2])))
xyz <- st_sfc(st_polygon(list(coords[, 1:3])))
xym <- st_sfc(st_polygon(list(coords[, 1:3]), dim = "XYM"))

as_esri_featureset(xy)
as_esri_featureset(xyz)
as_esri_featureset(xym)

# MULTIPOLYGON
as_esri_featureset(st_sfc(st_multipolygon(list(xy[[1]], xy[[1]]))))
as_esri_featureset(st_sfc(st_multipolygon(list(xyz[[1]], xyz[[1]]))))
as_esri_featureset(st_sfc(st_multipolygon(list(xym[[1]], xym[[1]]))))

```

as\_fields

*Esri Field Type Mapping***Description**

Infers Esri field types from R objects. Use `as_fields()` to create a data.frame of valid **Esri Field Types** from an sf object or data.frame.

**Usage**

```

as_fields(.data, arg = rlang::caller_arg(.data), call = rlang::caller_env())

infer_esri_type(
  .data,
  arg = rlang::caller_arg(.data),
  call = rlang::caller_env()
)

```

```
fields_as_ptype_df(fields, n = 0, call = rlang::caller_env())
```

```
ptype_tbl(fields, n = 0, call = rlang::caller_env())
```

### Arguments

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.data</code>  | an object of class <code>data.frame</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>arg</code>    | An argument name in the current function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>call</code>   | <p>The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be <code>NULL</code> or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |
| <code>fields</code> | a list or <code>data.frame</code> of field types. Requires the fields type and name to be present.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>n</code>      | the number of rows to create in the prototype table                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

### Details

#### [Experimental]

- `as_fields()` takes a data frame-like object and infers the Esri field type from it.
- `fields_as_ptype_df()` takes a list with type and name and creates an empty `data.frame` with the corresponding column names and types.
- `get_ptype()` takes a scalar character containing the Esri field type and returns a prototype of the pertinent R type

#### Field type mapping::

Esri field types are mapped as

- `esriFieldTypeSmallInteger`: integer
- `esriFieldTypeSingle`: double
- `esriFieldTypeGUID`: integer
- `esriFieldTypeOID`: integer
- `esriFieldTypeInteger`: integer
- `esriFieldTypeBigInteger`: double
- `esriFieldTypeDouble`: double
- `esriFieldTypeString`: character
- `esriFieldTypeDate`: date

R types are mapped as

- `double`: `esriFieldTypeDouble`

- integer: esriFieldTypeInteger
- character: esriFieldTypeString
- date: esriFieldTypeDate
- raw: esriFieldTypeBlob

### Value

- fields\_as\_ptype\_df() takes a data.frame with columns name and type and creates an empty data.frame with the corresponding columns and R types
- as\_fields() returns a data.frame with columns name, type, alias, nullable, and editable columns
  - This resembles that of the fields returned by a FeatureService

### Examples

```
inferred <- as_fields(iris)
inferred

fields_as_ptype_df(inferred)
```

---

|          |                           |
|----------|---------------------------|
| as_layer | Create Esri layer objects |
|----------|---------------------------|

---

### Description

These functions are used to generate list objects that can be converted into json objects that are used in REST API requests. Notably they are used for adding R objects as items to a portal.

### Usage

```
as_layer(
  x,
  name,
  title,
  layer_definition = as_layer_definition(x, name, "object_id", as_fields(x)),
  id = NULL,
  layer_url = NULL,
  legend_url = NULL,
  popup_info = NULL,
  call = rlang::caller_env()
)

as_layer_definition(
  x,
  name,
  object_id_field,
```

```

    fields = as_fields(x),
    display_field = NULL,
    drawing_info = NULL,
    has_attachments = FALSE,
    max_scale = 0,
    min_scale = 0,
    templates = NULL,
    type_id_field = NULL,
    types = NULL,
    call = rlang::caller_env()
  )

  as_feature_collection(
    layers = list(),
    show_legend = TRUE,
    call = rlang::caller_env()
  )

```

### Arguments

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                | an object of class <code>data.frame</code> . This can be an <code>sf</code> object or <code>tibble</code> or any other subclass of <code>data.frame</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>name</code>             | a scalar character of the name of the layer. Must be unique.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>title</code>            | A user-friendly string title for the layer that can be used in a table of contents.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>layer_definition</code> | a layer definition list as created by <code>as_layer_definition()</code> . A default is derived from <code>x</code> and the name object.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>id</code>               | A number indicating the index position of the layer in the WMS or map service.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>layer_url</code>        | default <code>NULL</code> . A string URL to a service that should be used for all queries against the layer. Used with hosted tiled map services on ArcGIS Online when there is an associated feature service that allows for queries.                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>legend_url</code>       | default <code>NULL</code> . A string URL to a legend graphic for the layer. Used with WMS layers. The URL usually contains a <code>GetLegendGraphic</code> request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>popup_info</code>       | default <code>NULL</code> . A list that can be converted into a <b>popupInfo</b> object defining the pop-up window content for the layer. There is no helper for <code>popupInfo</code> objects.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>call</code>             | The execution environment of a currently running function, e.g. <code>call = caller_env()</code> . The corresponding function call is retrieved and mentioned in error messages as the source of the error.<br><br>You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message.<br><br>Can also be <code>NULL</code> or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.<br><br>For more information about error calls, see <a href="#">Including function calls in error messages</a> . |
| <code>object_id_field</code>  | a scalar character vector indicating the name of the object ID field in the dataset.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|                 |                                                                                                                                                                                                                                                                                                      |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fields          | a data.frame describing the fields in x. These values are inferred by default via <a href="#">as_fields()</a> .                                                                                                                                                                                      |
| display_field   | default NULL. A scalar character containing the name of the field that best summarizes the feature. Values from this field are used by default as the titles for pop-up windows.                                                                                                                     |
| drawing_info    | default NULL. See REST documentation in details for more. There are no helpers or validators for drawingInfo objects.                                                                                                                                                                                |
| has_attachments | default FALSE.                                                                                                                                                                                                                                                                                       |
| max_scale       | default NULL. A number representing the maximum scale at which the layer definition will be applied. The number is the scale's denominator; thus, a value of 2400 represents a scale of 1/2,400. A value of 0 indicates that the layer definition will be applied regardless of how far you zoom in. |
| min_scale       | default NULL. A number representing the minimum scale at which the layer definition will be applied.                                                                                                                                                                                                 |
| templates       | default NULL. See REST documentation in details for more.                                                                                                                                                                                                                                            |
| type_id_field   | default NULL. See REST documentation in details for more.                                                                                                                                                                                                                                            |
| types           | An array of type objects available for the dataset. This is used when the type_id_field is populated. NOTE there are no helper functions to create type objects. Any type list objects must match the json structure when passed to <code>jsonify::to_json(x, unbox = TRUE)</code> .                 |
| layers          | a list of layers as created by <code>as_layer()</code> .                                                                                                                                                                                                                                             |
| show_legend     | default FALSE. Logical scalar indicating if this layer should be shown in the legend in client applications.                                                                                                                                                                                         |

## Details

A `FeatureCollection` defines a layer of features that will be stored on a web map. It consists of an array of layers. The layer contains the features (attributes and geometries) as a `FeatureSet` (see [as\\_esri\\_featureset\(\)](#)) and additional metadata which is stored in the `LayerDefinition` object. The `LayerDefinition` most importantly documents the fields in the object, the object ID, and additional metadata such as name, title, and display scale.

Additional documentation for these json object:

- [layer](#)
- [layerDefinition](#)
- [featureCollection](#)

## Value

A list object containing the required fields for each respective json type. The results can be converted to json using `jsonify::to_json(x, unbox = TRUE)`

**Examples**

```
ld <- as_layer_definition(iris, "iris", "objectID")
l <- as_layer(iris, "iris name", "Iris Title")
fc <- as_feature_collection(layers = list(l))
```

auth\_code

*Authorization***Description**

Authorize your R session to connect to an ArcGIS Portal. See details.

**Usage**

```
auth_code(client = Sys.getenv("ARCGIS_CLIENT"), host = arc_host())

auth_client(
  client = Sys.getenv("ARCGIS_CLIENT"),
  secret = Sys.getenv("ARCGIS_SECRET"),
  host = arc_host(),
  expiration = 120
)

auth_binding()

auth_user(
  username = Sys.getenv("ARCGIS_USER"),
  password = Sys.getenv("ARCGIS_PASSWORD"),
  host = arc_host(),
  expiration = 60
)

auth_key(api_key = Sys.getenv("ARCGIS_API_KEY"), host = arc_host())

refresh_token(token, client = Sys.getenv("ARCGIS_CLIENT"), host = arc_host())

validate_or_refresh_token(
  token,
  client = Sys.getenv("ARCGIS_CLIENT"),
  host = arc_host(),
  refresh_threshold = 10,
  call = rlang::caller_env()
)
```

**Arguments**

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| client            | an OAuth 2.0 developer application client ID. By default uses the environment variable ARCGIS_CLIENT.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| host              | default "https://www.arcgis.com". The host of your ArcGIS Portal.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| secret            | an OAuth 2.0 developer application secret. By default uses the environment variable ARCGIS_SECRET.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| expiration        | the duration of the token in minutes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| username          | default Sys.getenv("ARCGIS_USER"). Your username to login. <b>Do not</b> hard code this value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| password          | default Sys.getenv("ARCGIS_PASSWORD"). Your password to login. <b>Do not</b> hard code this value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| api_key           | default Sys.getenv("ARCGIS_API_KEY"). A character scalar of an ArcGIS Developer API key.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| token             | an httr2_token as created by auth_code() or similar                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| refresh_threshold | default 10. If token expiry is within this threshold (in seconds) the token will be refreshed only if a refresh_token is available. Token refreshing is only possible with auth_code() flow.                                                                                                                                                                                                                                                                                                                                                                                                           |
| call              | <p>The execution environment of a currently running function, e.g. call = caller_env(). The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply call when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be NULL or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |

**Details**

ArcGIS Online and Enterprise Portals utilize OAuth2 authorization via their REST APIs.

- auth\_code() is the recommend OAuth2 workflow for interactive sessions
- auth\_client() is the recommended OAuth2 workflow for non-interactive sessions
- auth\_user() uses legacy username and password authorization using the generateToken endpoint. It is only recommended for legacy systems that do not implement OAuth2.
- auth\_binding() fetches a token from the active portal set by arcgisbinding. Uses arcgisbinding::arc.check\_portal to extract the authorization token. Recommended if using arcgisbinding.

**Value**

an httr2\_token

## Examples

```
## Not run:
auth_code()
auth_client()
auth_user()
auth_key()
auth_binding()

## End(Not run)
```

---

auth\_shiny

*Authenticate with Shiny*


---

## Description

oauth\_provider\_arcgis() and auth\_shiny() provide an integrated authentication experience for {shiny} applications via the package [shinyOAuth](#). Applications that use auth\_shiny() will enable user-specific behavior and authentications.

## Usage

```
oauth_provider_arcgis(host = arc_host())

auth_shiny(
  client = Sys.getenv("ARCGIS_CLIENT"),
  secret = Sys.getenv("ARCGIS_SECRET"),
  redirect_uri = Sys.getenv("ARCGIS_REDIRECT_URI"),
  host = arc_host(),
  ...
)
```

## Arguments

|              |                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------|
| host         | default "https://www.arcgis.com". The host of your ArcGIS Portal.                                     |
| client       | an OAuth 2.0 developer application client ID. By default uses the environment variable ARCGIS_CLIENT. |
| secret       | an OAuth 2.0 developer application secret. By default uses the environment variable ARCGIS_SECRET.    |
| redirect_uri | default Sys.getenv("ARCGIS_REDIRECT_URI"). The redirect URL after completing authentication flow.     |
| ...          | additional arguments passed to <a href="#">shinyOAuth::oauth_client()</a>                             |

## Details

Authentication with `auth_shiny()` requires the package `shinyOAuth`. When creating an OAuth app in ArcGIS Online / Enterprise, ensure that the app has a valid redirect URI. This **must be** the same redirect URI that is used by `shinyOAuth` and **must be** the same port as the shiny application is served on.

`oauth_provider_arcgis()` is a low-level provider function that mimics other `oauth_provider_*`() functions from `shinyOAuth`. Use `auth_shiny()` to create an OAuth client.

**Important:** The client object returned by `auth_shiny()` must be created outside of the server function to ensure a shared state across the OAuth flow. Creating the client inside the server function will result in authentication failures.

The user info returned from `auth$token@userinfo` has the same structure as `arc_portal_self()`. See that function's documentation for available fields.

The below example is derived from the `shinyOAuth` documentation.

```
library(shiny)
library(shinyOAuth)
library(arcgisutils)

# Simple UI
ui <- fluidPage(
  use_shinyOAuth(),
  uiOutput("login_information")
)

client <- auth_shiny()

server <- function(input, output, session) {
  # Set auto_redirect = FALSE for manual login
  auth <- shinyOAuth::oauth_module_server(
    "AGOL",
    client,
    auto_redirect = FALSE
  )

  output$login_information <- renderUI({
    if (auth$authenticated) {
      user_info <- auth$token@userinfo
      tagList(
        tags$p("You are logged in! Your details:"),
        tags$pre(paste(capture.output(str(user_info)), collapse = "\n"))
      )
    } else {
      tagList(
        tags$p("You are not logged in."),
        actionButton("login_btn", "Login with ArcGIS")
      )
    }
  })
}
```

```

    })

    # Trigger login when button is clicked
    observeEvent(input$login_btn, {
      auth$request_login()
    })
  }

  runApp(
    shinyApp(ui, server),
    port = 8100,
    launch.browser = FALSE
  )

  # Open the app in your regular browser at http://localhost:8100

```

compact

*General utility functions***Description**

General utility functions

**Usage**

```

compact(.x)

a %||% b

check_dots_named(dots, call = rlang::caller_env())

data_frame(x, call = rlang::caller_call())

```

**Arguments**

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>.x</code>   | a list                                                        |
| <code>a</code>    | an R object                                                   |
| <code>b</code>    | an R object                                                   |
| <code>dots</code> | a list collected from dots via <code>rlang::list2(...)</code> |
| <code>call</code> | default <code>rlang::caller_call()</code> .                   |
| <code>x</code>    | a data.frame                                                  |

**Details**

- `compact()` removes any NULL list elements
- `%||%` is a special pipe operator that returns `b` if `a` is NULL

**Value**

- `compact()` a list
- `%||%` the first non-null item or NULL if both are NULL

**Examples**

```
# remove null elements
compact(list(a = NULL, b = 1))

# if NULL return rhs
NULL %||% 123

# if not NULL return lhs
123 %||% NULL
```

---

|         |                             |
|---------|-----------------------------|
| content | <i>Portal Content Items</i> |
|---------|-----------------------------|

---

**Description**

For a given user or group, returns a data.frame of all content items owned by them.

**Usage**

```
arc_group_content(group, host = arc_host(), token = arc_token())

arc_user_content(
  user = arc_user_self(token = token),
  host = arc_host(),
  token = arc_token()
)
```

**Arguments**

|       |                                                                                                      |
|-------|------------------------------------------------------------------------------------------------------|
| group | a scalar character of the group ID or a PortalGroup object created using <a href="#">arc_group()</a> |
| host  | default "https://www.arcgis.com". The host of your ArcGIS Portal.                                    |
| token | an httr2_token as created by <code>auth_code()</code> or similar                                     |
| user  | a scalar character of the username or a PortalUser object created using <a href="#">arc_user()</a>   |

**Value**

a data.frame of content item metadata

**References**

- [Group Content API Reference](#)
- [User Content API Reference](#)

## Examples

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

# authenticate
set_arc_token(auth_user())

# get your own content items
self <- arc_user_self()
arc_user_content(self$username)

# get a specific group's items
arc_group_content("2f0ec8cb03574128bd673cefab106f39")

## End(Not run)
```

---

detect\_errors

*Detect errors in parsed json response*


---

## Description

The requests responses from ArcGIS don't return the status code in the response itself but rather from the body in the json. This function checks for the existence of an error. If an error is found, the contents of the error message are bubbled up.

## Usage

```
detect_errors(response, error_call = rlang::caller_env())

catch_error(response, error_call = rlang::caller_env())
```

## Arguments

|            |                                                                                                                                                                                                                           |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| response   | for <code>detect_errors()</code> , a list typically from <code>RcppSimdJson::fparse(httr2::resp_body_string(resp))</code> . For <code>catch_error()</code> , the string from <code>httr2::resp_body_string(resp)</code> . |
| error_call | default <code>rlang::caller_env()</code> . The environment from which to throw the error from.                                                                                                                            |

## Value

Nothing. Used for it's side effect. If an error code is encountered in the response an error is thrown with the error code and the error message.

**Examples**

```
## Not run:
response <- list(
  error = list(
    code = 400L,
    message = "Unable to generate token.",
    details = "Invalid username or password."
  )
)

detect_errors(response)

## End(Not run)
```

determine\_dims

*Determine the dimensions of a geometry object***Description**

Given an sfc or sfg object determine what dimensions are represented.

**Usage**

```
determine_dims(x)
```

```
has_m(x)
```

```
has_z(x)
```

**Arguments**

x                      an object of class sfc or sfg

**Value**

determine\_dims() returns a scalar character of the value "xy", "xyz", or "xyzm" depending on what dimensions are represented.

has\_m() and has\_z() returns a logical scalar of TRUE or FALSE if the geometry has a Z or M dimension.

**Examples**

```
geo <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)[["geometry"]]

determine_dims(geo)
has_z(geo)
has_m(geo)
```

---

`determine_esri_geo_type`*Determine Esri Geometry type*

---

## Description

Takes an `sf` or `sfc` object and returns the appropriate Esri geometry type.

## Usage

```
determine_esri_geo_type(x, call = rlang::caller_env())
```

## Arguments

- |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>    | an object of class <code>data.frame</code> , <code>sf</code> , <code>sfc</code> , or <code>sfg</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>call</code> | <p>The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be <code>NULL</code> or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |

## Details

### Geometry type mapping:

- POINT: `esriGeometryPoint`
- MULTIPOINT: `esriGeometryMultipoint`
- LINESTRING: `esriGeometryPolyline`
- MULTILINESTRING: `esriGeometryPolyline`
- POLYGON: `esriGeometryPolygon`
- MULTIPOLYGON: `esriGeometryPolygon`

## Value

returns a character scalar of the corresponding Esri geometry type

## Examples

```
determine_esri_geo_type(sf::st_point(c(0, 0)))
```

---

|                      |                          |
|----------------------|--------------------------|
| fetch_layer_metadata | <i>Retrieve metadata</i> |
|----------------------|--------------------------|

---

## Description

Utility functions for feature service metadata.

## Usage

```
fetch_layer_metadata(url, token = NULL, call = rlang::caller_env())
```

## Arguments

|       |                                                                                                             |
|-------|-------------------------------------------------------------------------------------------------------------|
| url   | the url of the item.                                                                                        |
| token | an <code>httr2_token</code> from one of the provided <code>auth_</code> functions                           |
| call  | default <code>rlang::caller_env()</code> . The calling environment passed to <code>detect_errors()</code> . |

## Details

- `fetch_layer_metadata()` given a request, fetches the metadata by setting the query parameter `f=json`

## Value

returns a list object

## Examples

```
# url is broken into parts to fit within 100 characters to avoid CRAN notes
url_parts <- c(
  "https://services.arcgis.com/P3ePLMYs2RVChkJx/ArcGIS/rest/services",
  "/USA_Counties_Generalized_Boundaries/FeatureServer/0"
)

furl <- paste0(url_parts, collapse = "")
meta <- fetch_layer_metadata(furl)
head(names(meta))
```

---

|                 |                                 |
|-----------------|---------------------------------|
| gp_job_from_url | Create GP Job from existing URL |
|-----------------|---------------------------------|

---

### Description

Create GP Job from existing URL

### Usage

```
gp_job_from_url(url, token = arc_token())
```

### Arguments

|       |                                                                          |
|-------|--------------------------------------------------------------------------|
| url   | the url of an existing geoprocessing job                                 |
| token | default <a href="#">arc_token()</a> . The token to be used with the job. |

### See Also

Other geoprocessing: [arc\\_form\\_params\(\)](#), [arc\\_gp\\_job](#), [arc\\_job\\_status\(\)](#), [gp\\_params](#)

### Examples

```
if (interactive()) {
  job_url <- paste0(
    "https://hydro.arcgis.com/arcgis/rest/services/Tools/Hydrology/",
    "GPSTraceDownstream/jobs/jfde67910074649e4a567f0adbb8af870"
  )

  gp_job_from_url(
    job_url,
    token = auth_user()
  )
}
```

---

|           |                               |
|-----------|-------------------------------|
| gp_params | Geoprocessing Parameter Types |
|-----------|-------------------------------|

---

### Description

Functions for converting R objects to and from ArcGIS geoprocessing parameter types. These functions handle the serialization and parsing of various data types used in ArcGIS geoprocessing services.

**Usage**

```

parse_gp_feature_record_set(json)

as_gp_feature_record_set(x)

parse_gp_record_set(json)

as_record_set(x)

as_gp_raster_layer(x)

gp_linear_unit(distance = integer(0), units = character(0))

as_gp_linear_unit(x)

parse_gp_linear_unit(json)

gp_areal_unit(area = integer(0), units = character(0))

as_gp_areal_unit(x)

parse_gp_areal_unit(json)

as_gp_date(x)

parse_gp_date(json)

as_spatial_reference(x)

from_spatial_reference(sr, error_call = rlang::caller_call())

parse_spatial_reference(json)

from_envelope(x, error_call = rlang::caller_call())

```

**Arguments**

|          |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| json     | raw json to parse                                                                                                                                                                                                                                                                                                                                                                                                 |
| x        | the object to convert into json                                                                                                                                                                                                                                                                                                                                                                                   |
| distance | a scalar number of the distance.                                                                                                                                                                                                                                                                                                                                                                                  |
| units    | the unit of the measurement. Must be one of "esriUnknownAreaUnits", "esriSquareInches", "esriSquareFeet", "esriSquareYards", "esriAcres", "esriSquareMiles", "esriSquareMillimeters", "esriSquareCentimeters", "esriSquareDecimeters", "esriSquareMeters", "esriAres", "esriHectares", "esriSquareKilometers", "esriSquareInchesUS", "esriSquareFeetUS", "esriSquareYardsUS", "esriAcresUS", "esriSquareMilesUS". |
| area     | a scalar number of the measurement.                                                                                                                                                                                                                                                                                                                                                                               |

sr                    a list with fields latestWkid, wkid, or wkt representing a spatial reference  
 error\_call          the caller environment to be used when propagating errors.

## Details

### [Experimental]

This package provides support for the following geoprocessing parameter types:

#### Implemented Types:

- **GPFeatureRecordSetLayer**: Feature collections with geometry and attributes
- **GPRecordSet**: Tabular data without geometry
- **GPRasterDataLayer**: Raster datasets from Portal items, Image Servers, or URLs
- **GPLinearUnit**: Linear distance measurements with units
- **GPArealUnit**: Area measurements with units
- **GPDate**: Date/time values in milliseconds since epoch
- **GPSpatialReference**: Coordinate reference systems

#### Not Yet Implemented:

The following types are planned for future implementation:

- **GPField**: Field definitions with name, type, and properties
- **GPMultiValue**: Arrays of values for a single data type
- **GPValueTable**: Flexible table-like objects with rows and columns
- **GPComposite**: Parameters that accept multiple data types
- **GPEnvelope**: Bounding box extents (use `as_extent()` for `GPEntent`)

## Usage Patterns

Most functions follow a consistent pattern:

- `as_gp_*`(): Convert R objects to geoprocessing parameter JSON
- `parse_gp_*`(): Parse geoprocessing response JSON to R objects
- Constructor functions (e.g., `gp_linear_unit()`, `gp_areal_unit()`) create typed S7 objects

## Examples

```
# Create a linear unit
distance <- gp_linear_unit(distance = 100, units = "esriMeters")

# Convert spatial data to feature record set
as_gp_feature_record_set(my_sf_data)

# Parse a geoprocessing response
parse_gp_feature_record_set(response_json)
```

## References

[API Documentation](#)

**See Also**

Other geoprocessing: [arc\\_form\\_params\(\)](#), [arc\\_gp\\_job](#), [arc\\_job\\_status\(\)](#), [gp\\_job\\_from\\_url\(\)](#)

**Examples**

```
# create a feature record set
fset <- as_gp_feature_record_set(iris[1,])
fset

# create fake gp feature record set to parse
fset_list <- list(
  list(
    dataType = "GPFeatureRecordSetLayer",
    paramName = "example",
    value = as_featureset(iris[1,])
  )
)

# create the json
json <- yyjsonr::write_json_str(fset_list, auto_unbox = TRUE)

# parse the record set json
parse_gp_feature_record_set(json)

# linear units
lu <- gp_linear_unit(10, "esriMeters")
lu
as_gp_linear_unit(lu)

# areal units
au <- gp_areal_unit(10, "esriSquareMeters")
au

as_gp_areal_unit(au)

# dates
json <- r"({
  \"paramName\": \"Output_Date\",
  \"dataType\": \"GPDate\",
  \"value\": 1199145600000
})"

parse_gp_date(json)
sr <- list(wkid = 4326L)
from_spatial_reference(sr)
x <- list(
  xmin = -122.4195,
  ymin = 37.3302190000000056,
  xmax = -122.030757,
  ymax = 37.776503600000007,
  spatialReference = list(wkid = 4326L, latestWkid = 4326L)
)
```

```
from_envelope(x)
```

---

```
is_date
```

```
Date handling
```

---

## Description

Esri date fields are represented as milliseconds from the Unix Epoch.

## Usage

```
is_date(x, tz)
```

```
date_to_ms(x, tz = "UTC")
```

```
from_esri_date(x)
```

## Arguments

|    |                                                                                                                                                                                                                                                                                                                                   |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x  | an object of class Date or POSIXt. In the case of is_date(), any R object.                                                                                                                                                                                                                                                        |
| tz | a character string. The time zone specification to be used for the conversion, <i>if one is required</i> . System-specific (see <a href="#">time zones</a> ), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning. |

## Details

- is\_date(): checks if an object is a Date or POSIXt class object.
- date\_to\_ms() converts a date object to milliseconds from the Unix Epoch in the specified time zone.

## Value

- is\_date() returns a logical scalar
- date\_to\_ms() returns a numeric vector of times in milliseconds from the Unix Epoch in the specified time zone.

## Examples

```
today <- Sys.Date()
```

```
is_date(today)
```

```
date_to_ms(today)
```

opts

*Configuration Options***Description**

These options are used to customize behavior of the R packages within the R-ArcGIS Bridge project.

**Custom Headers:** `arcgis.req_headers`:

A named list that is injected into all requests in `arc_base_req()` via [http2::req\\_headers](#).

Use this to inject, for example, a Referer header.

**Debug Requests:** `arcgis.req_debug`:

When configured by a package, will print the cURL request created by the used package via [http2::http2\\_translate](#).

**Examples**

```
options(
  "arcgis.req_headers" = list(
    referer = "https://myportal.com",
    `X-Custom-Header` = "sneaky-value"
  )
)
# see the headers
arc_base_req("https://arcgis.com")
```

parse\_esri\_json

*Parse Esri JSON***Description**

Parses an Esri FeatureSet JSON object into an R object. If there is no geometry present, a `data.frame` is returned. If there is geometry, an `sf` object is returned.

**Usage**

```
parse_esri_json(string, ..., call = rlang::caller_env())
```

**Arguments**

`string` the raw Esri JSON string.

`...` additional arguments passed to [RcppSimdJson::fparse](#)

**call** The execution environment of a currently running function, e.g. `call = caller_env()`. The corresponding function call is retrieved and mentioned in error messages as the source of the error.

You only need to supply `call` when throwing a condition from a helper function which wouldn't be relevant to mention in the message.

Can also be `NULL` or a [defused function call](#) to respectively not display any call or hard-code a code to display.

For more information about error calls, see [Including function calls in error messages](#).

### Value

A data.frame. If geometry is found, returns an sf object.

### Examples

```
esri_json <- '{
  "geometryType": "esriGeometryPolygon",
  "spatialReference": {
    "wkid": 4326
  },
  "hasZ": false,
  "hasM": false,
  "features": [
    {
      "attributes": {
        "id": 1
      },
      "geometry": {
        "rings": [
          [
            [0.0, 0.0],
            [1.0, 0.0],
            [1.0, 1.0],
            [0.0, 1.0],
            [0.0, 0.0]
          ]
        ]
      }
    }
  ]
}'

parse_esri_json(esri_json)
```

## Description

Every portal item has an associated item type. Each of those item types have keywords which can be used to help narrow down search further.

## Usage

```
item_type(item_type = character(0))

item_keyword(keyword = character(0))

portal_item_keywords()

portal_item_types()
```

## Arguments

`item_type` a scalar character of the item type. See [portal\\_item\\_types\(\)](#) for valid item types.

`keyword` a scalar character of the item type keyword. See [portal\\_item\\_keywords\(\)](#).

## References

[REST API Documentation](#)

---

|               |                                     |
|---------------|-------------------------------------|
| rbind_results | <i>Combine multiple data.frames</i> |
|---------------|-------------------------------------|

---

## Description

A general function that takes a list of `data.frames` and returns a single and combines them into a single object. It will use the fastest method available. In order this is [collapse::rowbind\(\)](#), [data.table::rbindlist\(\)](#), [vctrs::list\\_unchop\(\)](#), then `do.call(rbind.data.frame, x)`.

## Usage

```
rbind_results(x, call = rlang::current_env(), .ptype = data.frame())
```

## Arguments

`x` a list where each element is a `data.frame` or `NULL`.

`call` The execution environment of a currently running function, e.g. `call = caller_env()`. The corresponding function call is retrieved and mentioned in error messages as the source of the error.

You only need to supply `call` when throwing a condition from a helper function which wouldn't be relevant to mention in the message.

Can also be `NULL` or a [defused function call](#) to respectively not display any call or hard-code a code to display.

For more information about error calls, see [Including function calls in error messages](#).

.ptype currently unused. Reserved for a future release.

**Details**

If all items in the list are data.frames, then the result will be a data.frame. If all elements are an sf object, then the result will be an sf object. If the items are mixed, the result will be a data.frame.

If any items are NULL, then an attribute null\_elements will be attached to the result. The attribute is an integer vector of the indices that were NULL.

**Value**

see details.

**Examples**

```
x <- head(iris)
res <- rbind_results(list(x, NULL, x))
attr(res, "null_elements")
```

---

|              |                                |
|--------------|--------------------------------|
| search_items | <i>Search for Portal Items</i> |
|--------------|--------------------------------|

---

**Description**

Perform full text search or use parameters to programamatically query your portal for content items.

**Usage**

```
search_items(
  query = NULL,
  filter = NULL,
  title = NULL,
  description = NULL,
  snippet = NULL,
  tags = NULL,
  owner = NULL,
  orgid = NULL,
  item_type = NULL,
  type_keywords = NULL,
  created = NULL,
  modified = NULL,
  categories = NULL,
  category_filters = NULL,
  sort_field = NULL,
```

```

    sort_order = NULL,
    count_fields = NULL,
    count_size = NULL,
    display_sublayers = FALSE,
    filter_logic = "and",
    bbox = NULL,
    page_size = 50,
    max_pages = Inf,
    .progress = TRUE,
    host = arc_host(),
    token = arc_token()
)

```

## Arguments

|                      |                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| query                | a scalar character for free text search or a valid query string as defined by the REST API.                                                                             |
| filter               | a scalar character. If provided all other arguments except query are ignored.                                                                                           |
| title                | optional character vector of content item titles.                                                                                                                       |
| description, snippet | optional scalar character of text to check for.                                                                                                                         |
| tags                 | optional character vector of tags to search for.                                                                                                                        |
| owner                | optional character vector of owner usernames to search for.                                                                                                             |
| orgid                | optional character vector of organization IDs to search for.                                                                                                            |
| item_type            | optional character vector of content item types. Validated with <a href="#">item_type()</a> .                                                                           |
| type_keywords        | optional character vector of content type keywords. Validated with <a href="#">item_keyword()</a> .                                                                     |
| created, modified    | optional length two vector which must be coercible to a date time vector. Converted using <a href="#">as.POSIXct()</a> . Returns only items within this range.          |
| categories           | optional character vector of up to 8 organization content categories.                                                                                                   |
| category_filters     | optional character vector of up to 3 category terms. Items that have matching categories are returned. Exclusive with categories.                                       |
| sort_field           | optional character vector of fields to sort by. Can sort by title, created, type, owner, modified, avgrating, numratings, numcomments, numviews, and scorecompleteness. |
| sort_order           | optional string. One of either asc or desc for ascending or descending order respectively.                                                                              |
| count_fields         | optional character vector of up to 3 fields to count. Must be one of c("type", "access", "contentstatus", "categories").                                                |
| count_size           | optional integer determines the maximum number of field values to count for each counted field in count_fields. Maximum of 200.                                         |
| display_sublayers    | default FALSE. Returns feature layers inside of feature services.                                                                                                       |

|              |                                                                                             |
|--------------|---------------------------------------------------------------------------------------------|
| filter_logic | default "and" must be one of c("and", "or", "not"). Determines if parameters                |
| bbox         | unimplemented.                                                                              |
| page_size    | a scalar integer between 1 and 100 indicating the number of responses per page.             |
| max_pages    | the maximum number of pages to fetch. By default fetches all pages.                         |
| .progress    | default TRUE. Whether to display a progress bar for requests.                               |
| host         | default "https://www.arcgis.com". The host of your ArcGIS Portal.                           |
| token        | an object of class httr2_token as generated by <code>auth_code()</code> or related function |

## Details

### [Experimental]

Search is quite nuanced and should be handled with care as you may get unexpected results.

- Most arguments are passed as filter parameters to the API endpoint.
- If multiple values are passed to an argument such as tags, the search will use an "OR" statement.
- When multiple arguments are provided, for example tags, owner, and item\_type, the search will use "AND" logic—i.e. results shown match the tags **and** owner **and** item\_type.
  - Note: you can change this to "OR" behavior by setting filter\_logic = "or"
- If the filter argument is provided, all other arguments except query are ignored.

## Value

a data.frame.

## References

[API Documentation](#)

## Examples

```
crime_items <- search_items(
  query = "crime",
  item_type = "Feature Service",
  max_pages = 1
)
crime_items
```

---

**self***Access the Portal Self Resource*

---

## Description

The function returns the `/self` resource from the ArcGIS REST API. The `/self` endpoint returns the view of the portal as seen by the current user, whether anonymous or signed in.

## Usage

```
arc_self_meta(token = arc_token(), error_call = rlang::current_call())
```

```
arc_portal_self(token = arc_token(), error_call = rlang::current_call())
```

## Arguments

|                         |                                                                                                          |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>token</code>      | an object of class <code>httr2_token</code> as generated by <code>auth_code()</code> or related function |
| <code>error_call</code> | the caller environment to be used when propagating errors.                                               |

## Details

See the [endpoint documentation](#) for more details.

The Portal Self response can vary based on whether it's called by a user, an app, or both.

The response includes user and appinfo properties, and the variations in responses are primarily related to these two properties. As the names indicate, the user property includes information about the user making the call, and the appinfo property includes information pertaining to the app that made the call.

## Value

A named list.

## See Also

Other portal: `arc_portal_resources()`, `arc_portal_urls()`, `arc_portal_users()`

## Examples

```
## Not run:
set_arc_token(auth_code())
self <- arc_self_meta()
names(self)

## End(Not run)
```

---

|     |                                                                   |
|-----|-------------------------------------------------------------------|
| url | <i>Parse an ArcGIS service or content URL into its components</i> |
|-----|-------------------------------------------------------------------|

---

### Description

`arc_url_parse()` uses `httr2::url_parse()` to parse URL components and combine the components with a service or content URL type and a layer number if applicable. A layer component is only included if the type is "MapServer" or "FeatureServer" and the URL includes a trailing digit. A full url value is also included in the returned list. The url, type, and layer components are not part of the `httr2_url` class object returned by `httr2::url_parse()`.

### Usage

```
arc_url_parse(url, base_url = NULL, error_call = rlang::caller_call())

arc_url_type(url, error_call = rlang::caller_call())

is_url(url, error_call = rlang::caller_call())
```

### Arguments

|            |                                                            |
|------------|------------------------------------------------------------|
| url        | A string containing the URL to parse.                      |
| base_url   | Use this as a parent, if url is a relative URL.            |
| error_call | the caller environment to be used when propagating errors. |

### Details

**[Experimental]**

### Value

A named list with the following components: scheme, hostname, username, password, port, path, query, fragment, url, type, and layer.

### Examples

```
arc_url_parse(
  "https://services.arcgisonline.com/arcgis/rest/services/USA_Topo_Maps/MapServer/0"
)
arc_url_parse(
  "https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer"
)
arc_url_parse(
  "https://services.arcgisonline.com/arcgis/rest/services/WorldElevation3D/Terrain3D/ImageServer"
)
```

---

validate\_crs

Validate CRS object

---

## Description

Takes a representation of a CRS and ensures that it is a valid one. The CRS is validated using `sf::st_crs()` if it cannot be validated, a null CRS is returned.

## Usage

```
validate_crs(crs, arg = rlang::caller_arg(crs), call = rlang::caller_env())
```

## Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>crs</code>  | a representation of a coordinate reference system.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>arg</code>  | An argument name in the current function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>call</code> | <p>The execution environment of a currently running function, e.g. <code>call = caller_env()</code>. The corresponding function call is retrieved and mentioned in error messages as the source of the error.</p> <p>You only need to supply <code>call</code> when throwing a condition from a helper function which wouldn't be relevant to mention in the message.</p> <p>Can also be <code>NULL</code> or a <a href="#">defused function call</a> to respectively not display any call or hard-code a code to display.</p> <p>For more information about error calls, see <a href="#">Including function calls in error messages</a>.</p> |

## Details

See `sf::st_crs()` for more details on valid representations.

## Value

Returns a list of length 1 with an element named `spatialReference` which is itself a named list. If the provided CRS returns a valid well-known ID (WKID) `spatialReference` contains a named element called `wkid` which is the integer value of the WKID. If the WKID is not known but the CRS returned is a valid well-known text representation the `wkid` field is `NA` and another field `wkt` contains the valid `wkt`.

## Examples

```
# using epsg code integer or string representation
validate_crs(3857)
validate_crs("EPSG:4326")

# using a custom proj4 string
proj4string <- "+proj=longlat +datum=WGS84 +no_defs"
```

```
crs <- validate_crs(proj4string)

# using wkt2 (from above result)
crs <- validate_crs(crs$spatialReference$wkt)
```

# Index

- \* **content**
  - content, [37](#)
- \* **geoprocessing**
  - arc\_form\_params, [4](#)
  - arc\_gp\_job, [5](#)
  - arc\_job\_status, [12](#)
  - gp\_job\_from\_url, [42](#)
  - gp\_params, [42](#)
- \* **portal item**
  - arc\_item, [10](#)
  - arc\_item\_data, [11](#)
- \* **portal organization**
  - arc\_group, [7](#)
  - arc\_user, [19](#)
- \* **portal**
  - arc\_portal\_resources, [13](#)
  - arc\_portal\_urls, [15](#)
  - arc\_portal\_users, [15](#)
  - arc\_user\_self, [19](#)
  - content, [37](#)
  - self, [53](#)
- \* **requests**
  - detect\_errors, [38](#)
- \* **self**
  - arc\_user\_self, [19](#)

arc\_agent, [3](#)  
arc\_base\_req, [3](#)  
arc\_base\_req(), [13](#)  
arc\_form\_params, [4](#)  
arc\_form\_params(), [5–7, 12, 42, 45](#)  
arc\_gp\_job, [5, 5, 12, 42, 45](#)  
arc\_group, [7](#)  
arc\_group(), [9, 19, 37](#)  
arc\_group\_content (content), [37](#)  
arc\_group\_users, [8](#)  
arc\_host, [9](#)  
arc\_item, [10](#)  
arc\_item(), [11](#)  
arc\_item\_data, [11](#)  
arc\_item\_data(), [10](#)  
arc\_job\_status, [12](#)  
arc\_job\_status(), [5–7, 42, 45](#)  
arc\_paginate\_req, [12](#)  
arc\_portal\_resources, [13](#)  
arc\_portal\_resources(), [15, 17, 53](#)  
arc\_portal\_self (self), [53](#)  
arc\_portal\_self(), [13, 14, 16, 35](#)  
arc\_portal\_servers, [14](#)  
arc\_portal\_urls, [15](#)  
arc\_portal\_urls(), [14, 17, 53](#)  
arc\_portal\_users, [15](#)  
arc\_portal\_users(), [14, 15, 53](#)  
arc\_self\_meta (self), [53](#)  
arc\_token, [17](#)  
arc\_token(), [5, 6, 42](#)  
arc\_url\_parse (url), [54](#)  
arc\_url\_parse(), [54](#)  
arc\_url\_type (url), [54](#)  
arc\_user, [19](#)  
arc\_user(), [8, 37](#)  
arc\_user\_content (content), [37](#)  
arc\_user\_self, [19](#)  
as.POSIXct(), [51](#)  
as\_esri\_features (as\_features), [23](#)  
as\_esri\_featureset (as\_featureset), [25](#)  
as\_esri\_featureset(), [31](#)  
as\_esri\_geometry, [20](#)  
as\_extent, [22](#)  
as\_feature\_collection (as\_layer), [29](#)  
as\_features, [23](#)  
as\_features(), [21](#)  
as\_featureset, [25](#)  
as\_featureset(), [21](#)  
as\_fields, [27](#)  
as\_fields(), [27, 31](#)  
as\_form\_params (arc\_form\_params), [4](#)  
as\_gp\_areal\_unit (gp\_params), [42](#)  
as\_gp\_date (gp\_params), [42](#)

as\_gp\_feature\_record\_set(gp\_params), 42  
 as\_gp\_linear\_unit(gp\_params), 42  
 as\_gp\_raster\_layer(gp\_params), 42  
 as\_layer, 29  
 as\_layer\_definition(as\_layer), 29  
 as\_record\_set(gp\_params), 42  
 as\_spatial\_reference(gp\_params), 42  
 auth\_binding(auth\_code), 32  
 auth\_client(auth\_code), 32  
 auth\_code, 32  
 auth\_code(), 4, 9, 13, 14, 16, 52, 53  
 auth\_key(auth\_code), 32  
 auth\_shiny, 34  
 auth\_user(auth\_code), 32  
  
 catch\_error(detect\_errors), 38  
 check\_dots\_named(compact), 36  
 check\_token\_has\_user(arc\_token), 17  
 collapse::rowbind(), 49  
 compact, 36  
 content, 37  
  
 data.table::rbindlist(), 49  
 data\_frame(compact), 36  
 date\_to\_ms(is\_date), 46  
 defused function call, 18, 21, 23, 24, 26, 28, 30, 33, 40, 48, 49, 55  
 detect\_errors, 38  
 determine\_dims, 39  
 determine\_esri\_geo\_type, 40  
  
 fetch\_layer\_metadata, 41  
 fields\_as\_ptype\_df(as\_fields), 27  
 from\_envelope(gp\_params), 42  
 from\_esri\_date(is\_date), 46  
 from\_spatial\_reference(gp\_params), 42  
  
 gp\_areal\_unit(gp\_params), 42  
 gp\_job\_from\_url, 42  
 gp\_job\_from\_url(), 5, 7, 12, 45  
 gp\_linear\_unit(gp\_params), 42  
 gp\_params, 5, 7, 12, 42, 42  
  
 has\_m(determine\_dims), 39  
 has\_z(determine\_dims), 39  
 httr2::httr2\_translate, 47  
 httr2::req\_headers, 47  
 httr2::req\_url\_path\_append(), 4  
 httr2::req\_url\_query(), 4  
  
 httr2::request(), 4  
 httr2::url\_parse(), 54  
  
 Including function calls in error  
     messages, 18, 21, 23, 24, 26, 28, 30, 33, 40, 48, 50, 55  
 infer\_esri\_type(as\_fields), 27  
 is\_date, 46  
 is\_url(url), 54  
 item\_keyword(portal\_types), 48  
 item\_keyword(), 51  
 item\_type(portal\_types), 48  
 item\_type(), 51  
  
 new\_gp\_job(arc\_gp\_job), 5  
  
 oauth\_provider\_arcgis(auth\_shiny), 34  
 obj\_check\_token(arc\_token), 17  
 opts, 47  
  
 parse\_esri\_json, 47  
 parse\_gp\_areal\_unit(gp\_params), 42  
 parse\_gp\_date(gp\_params), 42  
 parse\_gp\_feature\_record\_set(gp\_params), 42  
 parse\_gp\_linear\_unit(gp\_params), 42  
 parse\_gp\_record\_set(gp\_params), 42  
 parse\_spatial\_reference(gp\_params), 42  
 portal\_item\_keywords(portal\_types), 48  
 portal\_item\_keywords(), 49  
 portal\_item\_types(portal\_types), 48  
 portal\_item\_types(), 49  
 portal\_types, 48  
 ptype\_tbl(as\_fields), 27  
  
 rbind\_results, 49  
 RcppSimdJson::fparse, 47  
 refresh\_token(auth\_code), 32  
 rlang::caller\_call(), 36  
 rlang::caller\_env(), 38, 41  
  
 search\_items, 50  
 self, 14, 15, 17, 53  
 set\_arc\_token(arc\_token), 17  
 sf::st\_crs(), 21, 24, 26, 55  
 shinyOAuth::oauth\_client(), 34  
  
 time zones, 46  
  
 unset\_arc\_token(arc\_token), 17

`url`, [54](#)

`validate_crs`, [55](#)

`validate_or_refresh_token(auth_code)`,  
[32](#)

`validate_or_refresh_token()`, [18](#)

`vctr::list_unchop()`, [49](#)