

# Package ‘feltr’

December 11, 2025

**Title** Access the Felt API

**Version** 0.1.0

**Description** Upload, download, and edit internet maps with the Felt API (<https://developers.felt.com/rest-api/getting-started>). Allows users to create new maps, edit existing maps, and extract data. Provides tools for working with layers, which represent geographic data, and elements, which are interactive annotations. Spatial data accessed from the API is transformed to work with 'sf'.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** cli, curl, dplyr, fs, geojsonsf, httr2, jsonlite, purrr, rlang, sf, stringr, tibble, tidyr

**URL** <https://github.com/christopherkenny/feltr>,  
<https://christophertkenny.com/feltr/>

**Suggests** httptest2, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Depends** R (>= 4.1.0)

**NeedsCompilation** no

**Author** Christopher T. Kenny [aut, cre] (ORCID:  
<https://orcid.org/0000-0002-9386-6860>)

**Maintainer** Christopher T. Kenny <ctkenny@proton.me>

**Repository** CRAN

**Date/Publication** 2025-12-11 16:20:02 UTC

## Contents

felt_add_library_layer . . . . .	2
felt_add_library_layer_group . . . . .	3
felt_add_map_elements . . . . .	4

felt_add_map_layers . . . . .	4
felt_add_map_layers_url . . . . .	5
felt_create_map . . . . .	6
felt_delete_map . . . . .	7
felt_delete_map_elements . . . . .	8
felt_delete_map_layer . . . . .	8
felt_get_comments . . . . .	9
felt_get_library . . . . .	10
felt_get_map . . . . .	10
felt_get_map_elements . . . . .	11
felt_get_map_layer_group . . . . .	12
felt_get_user . . . . .	13
felt_patch_style . . . . .	13
felt_refresh_layer . . . . .	14
felt_update_layer_details . . . . .	15
felt_update_layer_group_details . . . . .	16
felt_update_map_details . . . . .	17
key . . . . .	18
set_felt_key . . . . .	18

## Index 20

---

felt\_add\_library\_layer

*Add layer to library*

---

### Description

Add layer to library

### Usage

```
felt_add_library_layer(map_id, layer_id, name = NULL, clean = TRUE)
```

### Arguments

map_id	character, map identifier from url, from <code>https://felt.com/map/Readable-Name-map_id</code>
layer_id	character, layer identifier from url, from <code>felt_get_map()</code>
name	character, name to save the layer under. Optional.
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

### Value

a [tibble::tibble](#) if `clean = TRUE`, otherwise a list

## Examples

```
felt_add_library_layer(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA',  
                      layer_id = '4Lc7RaEyRP2LfARGmR6e4C',  
                      name = paste0('Test layer ', Sys.time()))
```

---

```
felt_add_library_layer_group  
      Add layer group to library
```

---

## Description

Add layer group to library

## Usage

```
felt_add_library_layer_group(map_id, layer_group_id, name = NULL, clean = TRUE)
```

## Arguments

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
layer_group_id	character, layer group identifier from url, from <code>felt_get_map()</code> .
name	character, name to save the layer group under. Optional.
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

## Value

a [tibble::tibble](#) if `clean = TRUE`, otherwise a list

## Examples

```
felt_add_library_layer_group(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA',  
                            layer_group_id = 'rHxyTef7S9C08W7n1PvBVwC',  
                            name = paste0('Test layer group ', Sys.time()))
```

---

felt\_add\_map\_elements *Add Elements to Existing Map*

---

**Description**

Add Elements to Existing Map

**Usage**

```
felt_add_map_elements(map_id, elements, clean = TRUE)
```

**Arguments**

map_id	character, map identifier from url, from <code>https://felt.com/map/Readable-Name-map_id</code>
elements	a <code>sf::sf</code> object or a path to a geojson file
clean	logical, whether to turn the API response into a <code>tibble::tibble</code>

**Value**

a `tibble::tibble` with the elements added

**Examples**

```
elem <- felt_add_map_elements(map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA',
                             elements = fs::path_package('feltr', 'bbox.geojson'))
elem
# and delete layer
felt_delete_map_elements(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA', element_id = elem$felt_id)
```

---

felt\_add\_map\_layers *Add Layers to Existing Map*

---

**Description**

Add Layers to Existing Map

**Usage**

```
felt_add_map_layers(
  map_id,
  name = NULL,
  file_names = NULL,
  lat = NULL,
  lng = NULL,
  zoom = NULL,
  clean = TRUE
)
```

**Arguments**

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
name	Name of the layer. Required.
file_names	Files to include. Required.
lat	For images, the latitude of the center of the image. Optional.
lng	For images, the longitude of the center of the image. Optional.
zoom	For images, the zoom level of the image. Optional.
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

**Value**

status of the upload

**Examples**

```
layer <- felt_add_map_layers(map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA',
  file_names = fs::path_package('feltr', 'towns.geojson'),
  name = 'Towns test')
layer
```

---

felt\_add\_map\_layers\_url

*Add Layers to Existing Map from URL*

---

**Description**

See [Felt "Upload Anything" documentation](#) for detailed examples of potential URLs.

**Usage**

```
felt_add_map_layers_url(map_id, url, name = NULL, clean = TRUE)
```

**Arguments**

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
url	Link to layer to include. Required
name	Name of the layer. Required.
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

**Value**

a [tibble::tibble](#) for the created layer

**Examples**

```
# split the URL for length reasons
url <- paste0(
  'https://www.rocklandgis.com/portal/sharing/rest/',
  'content/items/73fc78cb0fb04580b4788937fe5ee697/data'
)
layer <- felt_add_map_layers_url(
  map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA',
  url = url,
  name = 'URL Parks test')
layer
# and delete the new layer
felt_delete_map_layer(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA', layer_id = layer$layer_id)
```

---

felt_create_map	<i>Create a new map</i>
-----------------	-------------------------

---

**Description**

Create a new map

**Usage**

```
felt_create_map(
  title = NULL,
  basemap = NULL,
  layer_urls = NULL,
  lat = NULL,
  lon = NULL,
  zoom = NULL,
  description = NULL,
  public_access = NULL,
  clean = TRUE
)
```

**Arguments**

title	Title to use for the map. Defaults to NULL.
basemap	Basemap for the new map. Defaults to NULL. Can be a URL or color hex code. Text options include 'default' (same as NULL), 'light', 'dark', or 'satellite'.
layer_urls	vector of URLs to generate layers in map. Defaults to NULL.
lat	latitude to center the map. Defaults to NULL.
lon	longitude to center the map. Defaults to NULL.
zoom	zoom level to initialize the map with. Defaults to NULL.

description	Description for the map legend. Defaults to NULL.
public_access	Degree of public access. Defaults to NULL, which is view_only. Text options also include 'private', 'view_and_comment', and 'view_comment_and_edit'.
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

### Value

a [tibble::tibble](#) for the new map

### Examples

```
map <- felt_create_map(title = 'feltr example')
map
# and delete it again
felt_delete_map(map_id = map$id)
```

---

felt_delete_map	<i>Delete an existing map</i>
-----------------	-------------------------------

---

### Description

Delete an existing map

### Usage

```
felt_delete_map(map_id)
```

### Arguments

map\_id            character, map identifier from url, from [https://felt.com/map/Readable-Name-map\\_id](https://felt.com/map/Readable-Name-map_id)

### Value

response code

### Examples

```
map <- felt_create_map(title = 'feltr example')
felt_delete_map(map_id = map$id)
```

---

```
felt_delete_map_elements
      Delete an existing element
```

---

**Description**

Delete an existing element

**Usage**

```
felt_delete_map_elements(map_id, element_id)
```

**Arguments**

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
element_id	element identifier, as returned by <code>felt_get_map_elements()</code> or <code>felt_add_map_elements()</code>

**Value**

response code

**Examples**

```
elem <- felt_add_map_elements(map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA',
                             elements = fs::path_package('feltr', 'bbox.geojson'))
elem
# and delete layer
felt_delete_map_elements(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA', element_id = elem$felt_id)
```

---

```
felt_delete_map_layer Delete Layer from an Existing Map
```

---

**Description**

Delete Layer from an Existing Map

**Usage**

```
felt_delete_map_layer(map_id, layer_id, clean = TRUE)
```

**Arguments**

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
layer_id	character, layer identifier from url, from <code>felt_get_map()</code>
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>



**Value**

response code

**Examples**

```
# split the URL for length reasons
url <- paste0(
  'https://www.rocklandgis.com/portal/sharing/rest/',
  'content/items/73fc78cb0fb04580b4788937fe5ee697/data'
)
layer <- felt_add_map_layers_url(
  map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA',
  url = url,
  name = 'URL Parks test')
layer
# and delete the new layer
felt_delete_map_layer(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA', layer_id = layer$layer_id)
```

---

felt\_get\_comments      *Export Comments on an Existing Map*

---

**Description**

Export Comments on an Existing Map

**Usage**

```
felt_get_comments(map_id, clean = TRUE)
```

**Arguments**

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

**Value**

a [tibble::tibble](#) for the map

**Examples**

```
felt_get_comments(map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA')
```

---

felt\_get\_library      *List all layers in your workspace library*

---

**Description**

List all layers in your workspace library

**Usage**

```
felt_get_library(source = "workspace", clean = TRUE)
```

**Arguments**

source            character, source of the layers, one of workspace, felt, or all  
clean             logical, whether to turn the API response into a [tibble::tibble](#)

**Value**

a [tibble::tibble](#) if clean = TRUE, otherwise a list

**Examples**

```
felt_get_library()
```

---

felt\_get\_map            *Get Map Information from Map ID*

---

**Description**

- felt\_get\_map() returns identifying information for the map
- felt\_get\_map\_layers() returns information about each layer in the map
- felt\_get\_map\_elements() returns the shapes for each layer in the map

**Usage**

```
felt_get_map(map_id, clean = TRUE)
```

```
felt_get_map_layers(map_id, clean = TRUE)
```

```
felt_get_map_layer(map_id, layer_id, clean = TRUE)
```

**Arguments**

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>
layer_id	character, layer identifier from url, from <code>felt_get_map()</code>

**Value**

a [tibble::tibble](#) for the map, if `clean = TRUE`, otherwise a list

**Examples**

```
felt_get_map('Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA')
felt_get_map_layers('Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA')

# slower, as it has to build the shapes from the API result
felt_get_map_elements('Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA')
```

---

`felt_get_map_elements` *Get Map Elements from Map ID*

---

**Description**

Get Map Elements from Map ID

**Usage**

```
felt_get_map_elements(map_id, clean = TRUE)

felt_get_map_element_groups(map_id, clean = TRUE)

felt_get_map_elements_in_group(map_id, group_id, clean = TRUE)
```

**Arguments**

map_id	character, map identifier from url, from <a href="https://felt.com/map/Readable-Name-map_id">https://felt.com/map/Readable-Name-map_id</a>
clean	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>
group_id	group identifier, as returned by <code>felt_get_map_element_groups()</code>

**Value**

a [tibble::tibble](#) for the map

## Examples

```
felt_get_map_elements(map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA')
felt_get_map_element_groups('TBI8sDkmQjuK2GX9CSiHiUA')
felt_get_map_elements_in_group('TBI8sDkmQjuK2GX9CSiHiUA', '3W15s2AqRmiYg09CrBFx03D')
```

---

felt\_get\_map\_layer\_group

*Get information for a layer group*

---

## Description

Get information for a layer group

## Usage

```
felt_get_map_layer_group(map_id, layer_group_id, clean = TRUE)
```

## Arguments

`map_id` character, map identifier from url, from [https://felt.com/map/Readable-Name-map\\_id](https://felt.com/map/Readable-Name-map_id)  
`layer_group_id` character, layer group identifier from url, from `felt_get_map()`.  
`clean` logical, whether to turn the API response into a [tibble::tibble](#)

## Value

a [tibble::tibble](#) for the layer group, if `clean = TRUE`, otherwise a list

## Examples

```
felt_get_map_layer_group(  
  map_id = 'Rockland-2024-Districts-TBI8sDkmQjuK2GX9CSiHiUA',  
  layer_group_id = 'rHxyTef7S9C08W7n1PvBVwC'  
)
```

---

felt_get_user	<i>Obtain information about the user</i>
---------------	--

---

**Description**

Obtain information about the user

**Usage**

```
felt_get_user(clean = TRUE)
```

**Arguments**

clean                    logical, whether to turn the API response into a [tibble::tibble](#)

**Value**

a [tibble::tibble](#) of information about the user

**Examples**

```
felt_get_user()
```

---

felt_patch_style	<i>Update Felt Style Information for a Layer</i>
------------------	--

---

**Description**

For details on the Felt Style Language, see <https://felt.com/blog/felt-style-language>.

**Usage**

```
felt_patch_style(map_id, layer_id, fsl, clean = TRUE)
```

**Arguments**

map\_id                    character, map identifier from url, from [https://felt.com/map/Readable-Name-map\\_id](https://felt.com/map/Readable-Name-map_id)  
layer\_id                    character, layer identifier from url, from `felt_get_map()`  
fsl                        A list indicating the Felt style language to update the layer to. It must be valid FSL.  
clean                      logical, whether to turn the API response into a [tibble::tibble](#)

**Value**

response data



---

`felt_update_layer_details`*Update a Layer's Details*

---

**Description**

Allows for updates to the name, ordering key, and subtitle.

**Usage**

```
felt_update_layer_details(  
  map_id,  
  layer_id,  
  layer_group_id = NULL,  
  name = NULL,  
  ordering_key = NULL,  
  subtitle = NULL,  
  clean = TRUE  
)
```

**Arguments**

<code>map_id</code>	character, map identifier from url, from <code>https://felt.com/map/Readable-Name-map_id</code>
<code>layer_id</code>	character, layer identifier from url, from <code>felt_get_map()</code>
<code>layer_group_id</code>	character, layer group identifier from url, from <code>felt_get_map()</code> .
<code>name</code>	Name of the layer. Defaults to NULL.
<code>ordering_key</code>	Integer to order layers. Defaults to NULL.
<code>subtitle</code>	Subtitle for the layer. Defaults to NULL.
<code>clean</code>	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

**Value**

response data

**Examples**

```
felt_update_layer_details(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA',  
                          layer_id = 'eufG5hWKRRSURHE8YcGGXA',  
                          subtitle = paste0('tested ', Sys.Date()))
```

---

`felt_update_layer_group_details`*Update a Layer Group's Details*

---

## Description

Allows for updates to the name, ordering key, and subtitle.

## Usage

```
felt_update_layer_group_details(  
  map_id,  
  layer_group_id,  
  name = NULL,  
  ordering_key = NULL,  
  subtitle = NULL,  
  clean = TRUE  
)
```

## Arguments

<code>map_id</code>	character, map identifier from url, from <code>https://felt.com/map/Readable-Name-map_id</code>
<code>layer_group_id</code>	character, layer group identifier from url, from <code>felt_get_map()</code> .
<code>name</code>	Name of the layer group. Required.
<code>ordering_key</code>	Integer to order by. Defaults to NULL.
<code>subtitle</code>	Subtitle for the layer group. Defaults to NULL.
<code>clean</code>	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

## Value

response data

## Examples

```
felt_update_layer_group_details(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA',  
                               layer_group_id = 'rHxyTef7S9C08W7n1PvBVwC',  
                               name = 'Polling Sites 2020',  
                               subtitle = paste0('tested ', Sys.Date()))
```



---

`felt_update_map_details`*Update a Map's Details*

---

## Description

Allows for updates to the title, description, and level of public access.

## Usage

```
felt_update_map_details(  
  map_id,  
  title = NULL,  
  description = NULL,  
  public_access = NULL,  
  clean = TRUE  
)
```

## Arguments

<code>map_id</code>	character, map identifier from url, from <code>https://felt.com/map/Readable-Name-map_id</code>
<code>title</code>	Title to use for the map. Defaults to NULL.
<code>description</code>	Description for the map legend. Defaults to NULL.
<code>public_access</code>	Degree of public access. Defaults to NULL, which is <code>view_only</code> . Text options also include <code>'private'</code> , <code>'view_and_comment'</code> , and <code>'view_comment_and_edit'</code> .
<code>clean</code>	logical, whether to turn the API response into a <a href="#">tibble::tibble</a>

## Value

response data

## Examples

```
felt_update_map_details(map_id = 'TBI8sDkmQjuK2GX9CSiHiUA',  
  title = paste0('Rockland 2024 Districts, tested ', Sys.Date()))
```

---

key	<i>Check or Get Felt API Key</i>
-----	----------------------------------

---

**Description**

Check or Get Felt API Key

**Usage**

```
has_felt_key()
```

```
get_felt_key()
```

```
felt_get_key()
```

```
felt_has_key()
```

**Value**

logical if has, key if get

**Examples**

```
has_felt_key()
```

---

set_felt_key	<i>Add Entry to Renviron</i>
--------------	------------------------------

---

**Description**

Adds Felt API key to .Renviron.

**Usage**

```
set_felt_key(key, overwrite = FALSE, install = FALSE, r_env = NULL)
```

```
felt_set_key(key, overwrite = FALSE, install = FALSE, r_env = NULL)
```

**Arguments**

key	Character. API key to add to add.
overwrite	Defaults to FALSE. Boolean. Should existing FELT_KEY in Renviron be overwritten?
install	Defaults to FALSE. Boolean. Should this be added to an environment file, r_env?
r_env	Path to install to if install is TRUE.

**Value**

key, invisibly

**Examples**

```
example_env <- tempfile(fileext = '.Renviron')
set_felt_key('1234', r_env = example_env)
# r_env should likely be: file.path(Sys.getenv('HOME'), '.Renviron')
```

# Index

- \* **comments**
  - felt\_get\_comments, 9
- \* **edits**
  - felt\_add\_map\_elements, 4
  - felt\_add\_map\_layers, 4
  - felt\_add\_map\_layers\_url, 5
  - felt\_delete\_map\_layer, 8
- \* **get**
  - felt\_get\_map, 10
  - felt\_get\_map\_elements, 11
  - felt\_update\_layer\_details, 15
  - felt\_update\_layer\_group\_details, 16
  - felt\_update\_map\_details, 17
- \* **key**
  - key, 18
  - set\_felt\_key, 18
- \* **map**
  - felt\_create\_map, 6
  - felt\_delete\_map, 7
- \* **style**
  - felt\_patch\_style, 13
- \* **user**
  - felt\_get\_user, 13

felt\_add\_library\_layer, 2  
felt\_add\_library\_layer\_group, 3  
felt\_add\_map\_elements, 4  
felt\_add\_map\_layers, 4  
felt\_add\_map\_layers\_url, 5  
felt\_create\_map, 6  
felt\_delete\_map, 7  
felt\_delete\_map\_elements, 8  
felt\_delete\_map\_layer, 8  
felt\_get\_comments, 9  
felt\_get\_key (key), 18  
felt\_get\_library, 10  
felt\_get\_map, 10  
felt\_get\_map\_element\_groups  
    (felt\_get\_map\_elements), 11  
felt\_get\_map\_elements, 11  
felt\_get\_map\_elements\_in\_group  
    (felt\_get\_map\_elements), 11  
felt\_get\_map\_layer (felt\_get\_map), 10  
felt\_get\_map\_layer\_group, 12  
felt\_get\_map\_layers (felt\_get\_map), 10  
felt\_get\_user, 13  
felt\_has\_key (key), 18  
felt\_patch\_style, 13  
felt\_refresh\_layer, 14  
felt\_set\_key (set\_felt\_key), 18  
felt\_update\_layer\_details, 15  
felt\_update\_layer\_group\_details, 16  
felt\_update\_map\_details, 17  
  
get\_felt\_key (key), 18  
  
has\_felt\_key (key), 18  
  
key, 18  
  
set\_felt\_key, 18  
sf::sf, 4  
  
tibble::tibble, 2–5, 7–17