

Package: gfwr (via r-universe)

October 11, 2026

Title Access data from Global Fishing Watch APIs

Version 3.0

Description This package connects to several Global Fishing Watch APIs to get vessel and events information in an R-friendly format.

License Apache License (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports class, dplyr, geojsonsf, httr2, jsonlite, lifecycle, lubridate, magrittr, methods, purrr, readr, rjson, rlang, tibble, tidyr

Suggests devtools, ggplot2, glue, knitr, qpdf, rmarkdown, rnaturalearth, rnaturalearthdata, scales, sf

URL <https://github.com/GlobalFishingWatch/gfwr>,
<https://globalfishingwatch.github.io/gfwr/>

BugReports <https://github.com/GlobalFishingWatch/gfwr/issues>

Depends R (>= 3.5)

LazyData true

Config/Needs/website rmarkdown

Config/pak/sysreqs libicu-dev libssl-dev libx11-dev

Repository <https://globalfishingwatch.r-universe.dev>

Date/Publication 2026-02-13 06:40:36 UTC

RemoteUrl <https://github.com/GlobalFishingWatch/gfwr>

RemoteRef HEAD

RemoteSha b8e7dcd1bda1bac49f345f74372493dcbc82ba2f

Contents

gfw_ais_fishing_hours	2
gfw_ais_presence	4
gfw_auth	6
gfw_event	7
gfw_event_stats	10
gfw_last_report	12
gfw_marine_regions	13
gfw_region_id	14
gfw_regions	15
gfw_sar_vessel_detections	16
gfw_test_shape	19
gfw_vessel_info	19
Index	22

gfw_ais_fishing_hours	<i>Retrieve apparent fishing hours derived from AIS data and convert response to tibble</i>
-----------------------	---

Description

Retrieve apparent fishing hours derived from AIS data and convert response to tibble

Usage

```
gfw_ais_fishing_hours(  
  spatial_resolution = NULL,  
  temporal_resolution = NULL,  
  start_date = NULL,  
  end_date = NULL,  
  region_source = NULL,  
  region = NULL,  
  group_by = NULL,  
  filter_by = NULL,  
  key = gfw_auth(),  
  print_request = FALSE  
)
```

Arguments

- spatial_resolution
Raster spatial resolution. Can be "LOW" (0.1 degree) or "HIGH" (0.01 degree).
- temporal_resolution
Raster temporal resolution. Can be "HOURLY", "DAILY", "MONTHLY", "YEARLY".
- start_date
Required. Start of date range to search events, in YYYY-MM-DD format and including this date.

end_date	Required. End of date range to search events, in YYYY-MM-DD format and excluding this date.
region_source	Required. Source of the region: "EEZ", "MPA", "RFMO" or "USER_SHAPEFILE".
region	Required. If region_source is set to "EEZ", "MPA" or "RFMO", GFW region code (see gfw_region_id()). If region_source = "USER_SHAPEFILE", sf shapefile with the area of interest.
group_by	Optional. Parameter to group by. Can be "VESSEL_ID", "FLAG", "GEARTYPE", "FLAGANDGEARTYPE" or "MMSI".
filter_by	Fields to filter AIS-based apparent fishing effort. Possible options are flag, shipname, geartype and id (to filter for vessel ids). Receives SQL expressions like filter_by = "flag IN ('ESP')".
key	Character, API token. Defaults to gfw_auth() .
print_request	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string.

Examples

```
## Not run:
library(gfwr)
# using region codes
code_eez <- gfw_region_id(region_name = "CIV", region_source = "EEZ")
gfw_ais_fishing_hours(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  group_by = "FLAG",
  start_date = "2021-01-01",
  end_date = "2022-01-01",
  region = code_eez$id,
  region_source = "EEZ",
  print_request = TRUE)

code_mpa <- gfw_region_id(region_name = "Galapagos", region_source = "MPA")
gfw_ais_fishing_hours(spatial_resolution = "LOW",
  temporal_resolution = "MONTHLY",
  group_by = "FLAG",
  start_date = "2022-01-01",
  end_date = "2023-01-01",
  region = code_mpa$id[3],
  region_source = "MPA")

code_rfmo <- gfw_region_id(region_name = "IATTC", region_source = "RFMO")
gfw_ais_fishing_hours(spatial_resolution = "LOW",
  temporal_resolution = "MONTHLY",
  start_date = "2022-01-01",
  end_date = "2023-01-01",
  region = code_rfmo$id[1],
  region_source = "RFMO")

#using a sf from disk /loading a test sf object
data(test_shape)
gfw_ais_fishing_hours(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  start_date = "2021-01-01",
  end_date = "2021-10-01",
```

```

    region = test_shape,
    region_source = "USER_SHAPEFILE",
    print_request = TRUE)

## End(Not run)

```

gfw_ais_presence	<i>Retrieve vessel presence from AIS data and convert response to tibble</i>
------------------	--

Description

This function communicates with the public-global-presence latest dataset to extract global vessel presence derived from AIS data. The presence is determined by taking one position per hour per vessel from the positions transmitted by the vessel's AIS. Unlike the apparent fishing effort dataset, this includes all vessel types and focuses on presence rather than fishing classification.

Usage

```

gfw_ais_presence(
  spatial_resolution = NULL,
  temporal_resolution = NULL,
  start_date = NULL,
  end_date = NULL,
  region_source = NULL,
  region = NULL,
  group_by = NULL,
  filter_by = NULL,
  key = gfw_auth(),
  print_request = FALSE
)

```

Arguments

spatial_resolution	Raster spatial resolution. Can be "LOW" (0.1 degree) or "HIGH" (0.01 degree).
temporal_resolution	Raster temporal resolution. Can be "HOURLY", "DAILY", "MONTHLY", "YEARLY".
start_date	Required. Start of date range to search events, in YYYY-MM-DD format and including this date.
end_date	Required. End of date range to search events, in YYYY-MM-DD format and excluding this date .
region_source	Required. Source of the region: "EEZ", "MPA", "RFMO" or "USER_SHAPEFILE".
region	Required. If region_source is set to "EEZ", "MPA" or "RFMO", GFW region code (see gfw_region_id()). If region_source = "USER_SHAPEFILE", sf shapefile with the area of interest.

group_by	Optional. Parameter to group by. Can be "VESSEL_ID", "FLAG", "GEARTYPE", "FLAGANDGEARTYPE" or "MMSI".
filter_by	Fields to filter AIS-based vessel presence. Possible options are flag, vessel_type, speed. See Details for more information
key	Character, API token. Defaults to <code>gfw_auth()</code> .
print_request	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string.

Details

The filter_by parameter accepts flag, vessel_type, speed options. This parameter accepts SQL expressions like filter_by = "flag IN ('ESP')", flag in ('ESP', 'USA'), vessel_type = 'cargo'. Accepted vessel speed ranges are the following:

- <2 – Less than 2 knots
- 2-4 – 2 to 4 knots
- 4-6 – 4 to 6 knots
- 6-10 – 6 to 10 knots
- 10-15 – 10 to 15 knots
- 15-25 – 15 to 25 knots
- >25 – Greater than 25 knots

References

AIS vessel presence examples in the API documentation <https://globalfishingwatch.org/our-apis/documentation#report-ais-vessel-presence-examples>

Examples

```
## Not run:
library(gfwr)
# using region codes

code_eez <- gfw_region_id(region_name = "CIV", region_source = "EEZ")
gfw_ais_presence(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  group_by = "FLAG",
  start_date = "2021-01-01",
  end_date = "2021-01-10",
  region = code_eez$id,
  region_source = "EEZ",
  key = gfw_auth(),
  print_request = TRUE)

# Grouping by flag

code_mpa <- gfw_region_id(region_name = "Galapagos", region_source = "MPA")
gfw_ais_presence(spatial_resolution = "LOW",
  temporal_resolution = "MONTHLY",
```

```

      group_by = "FLAG",
      start_date = "2022-01-01",
      end_date = "2022-02-01",
      region = code_mpa$id[3],
      region_source = "MPA")

# Filter by flag but grouping by VESSEL_ID

code_mpa <- gfw_region_id(region_name = "Galapagos", region_source = "MPA")
gfw_ais_presence(spatial_resolution = "LOW",
  temporal_resolution = "MONTHLY",
  group_by = "VESSEL_ID",
  filter_by = "flag in ('ECU')",
  start_date = "2022-01-01",
  end_date = "2022-02-01",
  region = code_mpa$id[3],
  region_source = "MPA")

code_rfmo <- gfw_region_id(region_name = "GFCM", region_source = "RFMO")
gfw_ais_presence(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  start_date = "2022-01-01",
  end_date = "2023-01-01",
  region = code_rfmo$id[1],
  region_source = "RFMO")

# Using a sf from disk /loading a test sf object
data(test_shape)
gfw_ais_presence(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  start_date = "2021-01-01",
  end_date = "2021-02-01",
  region = test_shape,
  region_source = "USER_SHAPEFILE",
  key = gfw_auth(),
  print_request = TRUE)

## End(Not run)

```

gfw_auth

Get user API token from .Renviron

Description

Get user API token from .Renviron

Usage

```
gfw_auth()
```

gfw_event

*Get events from API and convert response to tibble***Description**

Get events from API and convert response to tibble

Usage

```
gfw_event(
  event_type,
  start_date = "2012-01-01",
  end_date = "2024-12-31",
  sort = "+start",
  vessels = NULL,
  flags = NULL,
  vessel_types = NULL,
  region_source = NULL,
  region = NULL,
  duration = 1,
  encounter_types = NULL,
  confidences = c(2, 3, 4),
  key = gfw_auth(),
  quiet = FALSE,
  print_request = FALSE,
  gap_intentional_disabling = deprecated(),
  ...
)
```

Arguments

event_type	Type of event to get data of. A vector with any combination of "ENCOUNTER", "FISHING", "GAP", "LOITERING", "PORT_VISIT"
start_date	Start of date range to search events, in YYYY-MM-DD format and including this date
end_date	End of date range to search events, in YYYY-MM-DD format and excluding this date
sort	How to sort the events. By default, +start, which sorts the events in ascending order (+) of the start dates of the events. Other possible values are -start, +end, -end.
vessels	A vector of vesselIds, obtained via gfw_vessel_info() . The maximum number of vesselIds depends on the character length of the whole request, the request will fail with error HTTP 422: Unprocessable entity when too many vesselIds are sent. This value is around 2,800 vessels, depending on the other parameters of the request.

flags	ISO3 code for the flag of the vessels. NULL by default.
vessel_types	A vector of vessel types, any combination of: "FISHING", "CARRIER", "SUPPORT", "PASSENGER", "OTHER_NON_FISHING", "SEISMIC_VESSEL", "BUNKER_OR_TANKER", "CARGO".
region_source	Optional. Source of the region ('EEZ', 'MPA', 'RFMO' or 'USER_SHAPEFILE').
region	Optional but required if a value for region_source is specified. If region_source is set to "EEZ", "MPA" or "RFMO", GFW region code (see gfw_region_id()). If region_source = "USER_SHAPEFILE", sf shapefile with the area of interest.
duration	Minimum duration that the event should have (in minutes). The default value is 1.
encounter_types	Only useful when event_type = "ENCOUNTER". Filters for types of vessels during the encounter. A vector with any combination of: "CARRIER-FISHING", "FISHING-CARRIER", "FISHING-SUPPORT", "SUPPORT-FISHING".
confidences	Only useful when event_type = "PORT_VISIT". Confidence levels of port visits. Possible values: 2, 3, or 4. Low-confidence port visits (confidence 1) are not available for download. See the API documentation for more details
key	Character, API token. Defaults to gfw_auth() .
quiet	Boolean. Whether to print the number of events returned by the request
print_request	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string
gap_intentional_disabling	Deprecated. The APIs will return only high confidence gaps (Defaults = TRUE).
...	Other arguments

Details

Fishing events (event_type = "FISHING") are specific to fishing vessels and loitering events (event_type = "LOITERING") are specific to carrier vessels. Port visits (event_type = "PORT_VISIT") and encounters (event_type = "ENCOUNTER") are available for all vessel types. For more details about the various event types, see the [GFW API documentation](#).

Encounter events involve multiple vessels and one row is returned for each vessel involved in an encounter. For example, an encounter between a carrier and fishing vessel (CARRIER-FISHING) will have one row for the fishing vessel and one for the carrier vessel. The id field for encounter events has two components separated by a .. The first component is the unique id for the encounter event and will be the same for all vessels involved in the encounter. The second component is an integer used to distinguish between different vessels in the encounter.

Examples

```
## Not run:
library(gfwr)
# port visits
gfw_event(event_type = "PORT_VISIT",
          vessels = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
          start_date = "2017-01-26",
```

```

        end_date = "2017-12-31",
        confidence = c(3, 4), # only for port visits
        key = gfw_auth())
#encounters
gfw_event(event_type = "ENCOUNTER",
vessels = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
  start_date = "2012-01-30",
  end_date = "2024-02-04",
  key = gfw_auth())
# fishing
gfw_event(event_type = "FISHING",
vessels = c("9b3e9019d-d67f-005a-9593-b66b997559e5"),
  start_date = "2017-01-26",
  end_date = "2023-02-04",
  key = gfw_auth())
# GAPS
gfw_event(event_type = "GAP",
vessels = c("e0c9823749264a129d6b47a7aabce377",
  "8c7304226-6c71-edbe-0b63-c246734b3c01"),
  start_date = "2017-01-26",
  end_date = "2023-02-04",
  key = gfw_auth())
# loitering
gfw_event(event_type = "LOITERING",
vessels = c("e0c9823749264a129d6b47a7aabce377",
  "8c7304226-6c71-edbe-0b63-c246734b3c01"),
  start_date = "2017-01-26",
  end_date = "2023-02-04",
  key = gfw_auth())
# encounter type
gfw_event(event_type = "ENCOUNTER",
encounter_types = "CARRIER-FISHING",
start_date = "2020-01-01",
end_date = "2020-01-31",
key = gfw_auth())

# vessel types
gfw_event(event_type = "ENCOUNTER",
vessel_types = c("CARRIER", "FISHING"),
start_date = "2020-01-01",
end_date = "2020-01-31",
key = gfw_auth())

# encounter events in Senegal EEZ
gfw_event(event_type = 'ENCOUNTER',
  start_date = "2020-10-01",
  end_date = "2020-12-31",
  region = 8371,
  region_source = 'EEZ',
  key = gfw_auth())

# Encounter events in user shapefile from a bounding box
test_polygon <- sf::st_bbox(c(xmin = -50, xmax = -20, ymin = -10, ymax = -30),

```

```
crs = 4326) |>
sf::st_as_sf() |>
sf::st_as_sf()
gfw_event(event_type = 'ENCOUNTER',
          start_date = "2023-01-01",
          end_date = "2023-06-01",
          region = test_polygon,
          region_source = 'USER_SHAPEFILE',
          key = gfw_auth())

## End(Not run)
```

gfw_event_stats	<i>Get events stats from API and convert response to tibble</i>
-----------------	---

Description

Get events stats from API and convert response to tibble

Usage

```
gfw_event_stats(
  event_type,
  start_date = "2012-01-01",
  end_date = "2024-12-31",
  interval = NULL,
  vessels = NULL,
  flags = NULL,
  vessel_types = NULL,
  region_source = NULL,
  region = NULL,
  duration = 1,
  encounter_types = NULL,
  confidences = c(2, 3, 4),
  key = gfw_auth(),
  quiet = FALSE,
  print_request = FALSE,
  ...
)
```

Arguments

event_type	Type of event to get data of. A vector with any combination of "ENCOUNTER", "FISHING", "GAP", "LOITERING", "PORT_VISIT"
start_date	Start of date range to search events, in YYYY-MM-DD format and including this date
end_date	End of date range to search events, in YYYY-MM-DD format and excluding this date

interval	Time series granularity. Must be a string. Possible values: 'HOUR', 'DAY', 'MONTH', 'YEAR'.
vessels	A vector of vesselIds, obtained via gfw_vessel_info()
flags	ISO3 code for the flag of the vessels. Null by default.
vessel_types	A vector of vessel types, any combination of: "FISHING", "CARRIER", "SUPPORT", "PASSENGER", "OTHER_NON_FISHING", "SEISMIC_VESSEL", "BUNKER_OR_TANKER", "CARGO"
region_source	Optional. Source of the region ('EEZ', 'MPA', 'RFMO' or 'USER_SHAPEFILE').
region	Optional but required if a value for region_source is specified. If region_source is set to "EEZ", "MPA" or "RFMO", GFW region code (see gfw_region_id()). If region_source = "USER_SHAPEFILE", sf shapefile with the area of interest.
duration	minimum duration of the event in minutes. The default value is 1.
encounter_types	Only useful when event_type = "ENCOUNTER". Filters for types of vessels during the encounter. A vector with any combination of: "CARRIER-FISHING", "FISHING-CARRIER", "FISHING-SUPPORT", "SUPPORT-FISHING".
confidences	Only useful when event_type = "PORT_VISIT". Confidence levels (2-4) of events.
key	Character, API token. Defaults to gfw_auth() .
quiet	Boolean. Whether to print the number of events returned by the request
print_request	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string
...	Other arguments

Details

There are currently four available event types and these events are provided for three vessel types - fishing, carrier, and support vessels. Fishing events (event_type = "FISHING") are specific to fishing vessels and loitering events (event_type = "LOITERING") are specific to carrier vessels. Port visits (event_type = "PORT_VISIT") and encounters (event_type = "ENCOUNTER") are available for all vessel types. For more details about the various event types, see the [GFW API documentation](#).

Examples

```
## Not run:
library(gfwr)
# stats for encounters involving Russian carriers in given time range
gfw_event_stats(event_type = 'ENCOUNTER',
  encounter_types = c("CARRIER-FISHING", "FISHING-CARRIER"),
  vessel_types = 'CARRIER',
  start_date = "2018-01-01",
  end_date = "2023-01-31",
  flags = 'RUS',
  duration = 60,
  interval = "YEAR")
```

```
# port visits stats in a region (Senegal)
gfw_event_stats(event_type = 'PORT_VISIT',
start_date = "2018-01-01",
end_date = "2019-01-31",
confidences = c('3', '4'),
region = 8371,
region_source = 'EEZ',
interval = "YEAR")

## End(Not run)
```

gfw_last_report

Base function to get status of last report generated

Description

Function to check the status of the last API request sent with [gfw_ais_fishing_hours\(\)](#).

Usage

```
gfw_last_report(key = gfw_auth())
```

Arguments

key Authorization token. Can be obtained with [gfw_auth\(\)](#) function

Details

The [gfw_last_report\(\)](#) function will tell you if the APIs are still processing your request and will download the results if the request has finished successfully. You will receive an error message if the request finished but resulted in an error or if it's been >30 minutes since the last report was generated using [gfw_ais_fishing_hours\(\)](#).

For more information, see the <https://globalfishingwatch.org/our-apis/documentation#get-last-report-generated>.

Examples

```
## Not run:
gfw_last_report(key = gfw_auth())

## End(Not run)
```

gfw_marine_regions	<i>Simplified Marine Regions v12 dataset (previously marine_regions)</i>
--------------------	--

Description

A tibble with the ISO codes and names derived from Marine Regions v12, to be used by `gfw_regions()` and `gfw_region_id()` to extract numeric EEZ codes (MRGID)

Usage

```
gfw_marine_regions
```

Format

A tibble with 285 rows and 5 columns

iso ISO Code derived from the different ISO code columns in the Maritime Boundaries Geodatabase v12

name Territory name derived from the different Name and Geoname columns in the Maritime Boundaries Geodatabase v12

MRGID Original numeric id for EEZs from the Marine Boundaries Geodatabase v12. This numeric code should be used in the `region` argument when using functions `gfw_ais_fishing_hours()`, `gfw_ais_presence()`, and `gfw_sar_vessel_detections()` and `gfw_event()`, when `region_source = "EEZ"`.

GEONAME Original GEONAME field from the Maritime Boundaries Geodatabase v12

POL_TYPE Original POLYgon TYPE from the Maritime Boundaries Geodatabase v12. Possible value are "200NM", "Overlapping claim" and "Joint regime"

Details

The main source data for EEZ shapefiles and information across Global Fishing Watch's data and platforms is Marine Regions v12.

The main variable of interest of this dataset is MRGID, the unique identifier for each one of the polygons comprising Marine Regions. MRGID is necessary to retrieve apparent fishing effort data and all event types from other gfw functions when `region_source = "EEZ"`

Functions `gfw_region_id()` and `gfw_regions()` facilitate fetching region ids from ISO codes and region names.

However, the original Marine Region dataset includes several columns that can refer to the ISO of the territory itself (ISO_TER1), the entity that has sovereignty over the territory (ISO_SOV1) or the territories that operate Joint regime areas, or claim Overlapping claims areas (ISO_SOV1, ISO_SOV2, ISO_SOV3).

The table `gfw_marine_regions` unifies all these ISO markers into a single column without modifying the original data, in a reproducible way.

- Joint regime and overlapping claims with no ISO receive an NA: `iso = NA`

- Territories within the 200NM with an ISO receive it: iso = ISO_TER1
- Overlapping claims with an ISO receive it: iso = ISO_TER1
- Territories within the 200NM and no ISO receive the ISO of their mainland: iso = ISO_SOVI

Likewise, gfw_marine_regions also derives a name column from the different names and GEONAME columns present in the original dataset.

- Countries and territories within the 200NM EEZs receive their name: name = TERRITORY1
- Overlapping claims with a name receive their name: name = TERRITORY1
- Joint regimes and overlapping claims with no name receive their GEONAME: name = GEONAME

The code implementing this synthesis can be found in the data-raw/marine_regions.R file.

Source

Marine Regions <https://www.marineregions.org/>. <https://doi.org/10.14284/632>

References

Flanders Marine Institute (2023). Maritime Boundaries Geodatabase: Maritime Boundaries and Exclusive Economic Zones (200NM), version 12. Available online at <https://www.marineregions.org/>. <https://doi.org/10.14284/632>

gfw_region_id	<i>Function to pull region code using region name and viceversa</i>
---------------	---

Description

Function to pull region code using region name and viceversa

Usage

```
gfw_region_id(
  region = NULL,
  region_source = "EEZ",
  key = gfw_auth(),
  region_name = deprecated()
)
```

Arguments

region	Character or numeric EEZ MPA or RFMO name or id.
region_source	Character, source of region data, "EEZ", "MPA" or "RFMO".
key	Character, API token. Defaults to gfw_auth().
region_name	Deprecated, replaced by region.

Value

For `gfw_region_id()`, the corresponding code, region names or iso code for the EEZ, MPA or RFMO label

See Also

[gfw_regions\(\)](#)

Examples

```
## Not run:
gfw_region_id(region = "COL", region_source = "EEZ")
gfw_region_id(region = "Colombia", region_source = "EEZ")
gfw_region_id(region = "Nazca", region_source = "MPA")
gfw_region_id(region = "IOTC", region_source = "RFMO")
gfw_region_id(region = 8456, region_source = "EEZ")
# Handling empty strings (high-seas)
gfw_region_id(region = "", region_source = "EEZ")
gfw_region_id(region = NA, region_source = "EEZ")
gfw_region_id(region = NA, region_source = "MPA")

## End(Not run)
```

gfw_regions	<i>List of available regions in Global Fishing Watch platforms, EEZs, MPAs, and RFMOs</i>
-------------	---

Description

List of available regions in Global Fishing Watch platforms, EEZs, MPAs, and RFMOs

Usage

```
gfw_regions(region_source = "EEZ", key = gfw_auth())
```

Arguments

`region_source` string, source of region data ("EEZ", "MPA", "RFMO")
`key` Character, API token. Defaults to [gfw_auth\(\)](#).

Value

A dataframe with all region ids and names for specified region type

See Also

[gfw_region_id\(\)](#)

Examples

```
## Not run:
gfw_regions(region_source = "EEZ")
gfw_regions(region_source = "RFMO")
gfw_regions(region_source = "MPA")

## End(Not run)
```

```
gfw_sar_vessel_detections
```

Retrieve vessel presence detected using SAR and convert response to tibble

Description

Retrieve vessel presence detected using SAR and convert response to tibble

Usage

```
gfw_sar_vessel_detections(
  spatial_resolution = NULL,
  temporal_resolution = NULL,
  start_date = NULL,
  end_date = NULL,
  region_source = NULL,
  region = NULL,
  group_by = NULL,
  filter_by = NULL,
  key = gfw_auth(),
  print_request = FALSE
)
```

Arguments

<code>spatial_resolution</code>	Raster spatial resolution. Can be "LOW" (0.1 degree) or "HIGH" (0.01 degree).
<code>temporal_resolution</code>	Raster temporal resolution. Can be "HOURLY", "DAILY", "MONTHLY", "YEARLY".
<code>start_date</code>	Required. Start of date range to search events, in YYYY-MM-DD format and including this date.
<code>end_date</code>	Required. End of date range to search events, in YYYY-MM-DD format and excluding this date.
<code>region_source</code>	Required. Source of the region: "EEZ", "MPA", "RFMO" or "USER_SHAPEFILE".
<code>region</code>	Required. If <code>region_source</code> is set to "EEZ", "MPA" or "RFMO", GFW region code (see gfw_region_id()). If <code>region_source</code> = "USER_SHAPEFILE", sf shapefile with the area of interest.

group_by	Optional. Parameter to group by. Can be "VESSEL_ID", "FLAG", "GEARTYPE", "FLAGANDGEARTYPE" or "MMSI".
filter_by	Fields to filter SAR vessel detections. See Details for possible options.
key	Character, API token. Defaults to <code>gfw_auth()</code> .
print_request	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string.

Details

Possible filter options are:

- `matched` – Whether detection matched with AIS data Example: `"matched='true'"` or `"matched='false'"`
- `flag` – Vessel flag state (when matched with AIS). Example: `"flag in ('ESP', 'USA')"`
- `vessel_id` – Vessel identifier (when matched with AIS). See the identity vignette for details about Vessel ID.
- `geartype` – Fishing gear type (when matched with AIS) → View [supported gear types](#). Example: `"geartype in ('tuna_purse_seines', 'driftnets')"`
- `neural_vessel_type` – AI classification based on neural network model. Values: `<= 0.1`: "Likely non-fishing", `>= 0.9`: "Likely fishing", `0.1 - 0.9`: "Other/Unknown"
- `shiptype` – Vessel type classification (when matched with AIS) → See [Vessel types](#)

Examples

```
## Not run:
library(gfwr)
# using region codes
code_eez <- gfw_region_id(region = "Chile", region_source = "EEZ")
gfw_sar_vessel_detections(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  group_by = "FLAG",
  start_date = "2021-01-01",
  end_date = "2022-01-01",
  region = code_eez$id,
  region_source = "EEZ",
  key = gfw_auth())
# filter by matched
gfw_sar_vessel_detections(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  group_by = "VESSEL_ID",
  filter_by = "matched = 'true'",
  start_date = "2021-01-01",
  end_date = "2022-01-01",
  region = code_eez$id,
  region_source = "EEZ",
  key = gfw_auth())
## Unmatched vessels will have no id information:
gfw_sar_vessel_detections(spatial_resolution = "LOW",
  temporal_resolution = "YEARLY",
  group_by = "VESSEL_ID",
```

```

        filter_by = "matched = 'false'",
        start_date = "2021-01-01",
        end_date = "2022-01-01",
        region = code_eez$id,
        region_source = "EEZ",
        key = gfw_auth())

# Filter by flag
gfw_sar_vessel_detections(spatial_resolution = "LOW",
                           temporal_resolution = "YEARLY",
                           group_by = "VESSEL_ID",
                           filter_by = "flag IN ('PER', 'ECU')",
                           start_date = "2021-01-01",
                           end_date = "2022-01-01",
                           region = code_eez$id,
                           region_source = "EEZ",
                           key = gfw_auth())

# Filter by vessel ID
gfw_sar_vessel_detections(spatial_resolution = "LOW",
                           temporal_resolution = "YEARLY",
                           filter_by = "vessel_id = '320335fcf-fbe5-54e0-9367-b36ae25b64b5'",
                           start_date = "2021-01-01",
                           end_date = "2022-01-01",
                           region = code_eez$id,
                           region_source = "EEZ",
                           key = gfw_auth())

# Filter by geartype
gfw_sar_vessel_detections(spatial_resolution = "LOW",
                           temporal_resolution = "YEARLY",
                           group_by = "VESSEL_ID",
                           filter_by = "geartype IN ('tuna_purse_seines',
                                                    'driftnets')",
                           start_date = "2021-01-01",
                           end_date = "2022-01-01",
                           region = code_eez$id,
                           region_source = "EEZ",
                           key = gfw_auth())

# Filter by neural_vessel_type
gfw_sar_vessel_detections(spatial_resolution = "LOW",
                           temporal_resolution = "YEARLY",
                           filter_by = "neural_vessel_type = '0.3'",
                           start_date = "2021-01-01",
                           end_date = "2022-01-01",
                           region = code_eez$id,
                           region_source = "EEZ",
                           key = gfw_auth())

# Filter by shiptype (vessel type)
gfw_sar_vessel_detections(spatial_resolution = "LOW",
                           temporal_resolution = "YEARLY",
                           filter_by = "shiptype = 'CARGO'",
                           start_date = "2021-01-01",
                           end_date = "2022-01-01",
                           region = code_eez$id,
                           region_source = "EEZ",

```

```
key = gfw_auth())
```

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

gfw_test_shape	<i>A sample shapefile (previously test_shape)</i>
----------------	---

Description

An sf shapefile to show as an example of user-defined GeoJSON in [gfw_event\(\)](#), [gfw_ais_fishing_hours\(\)](#), [gfw_ais_presence\(\)](#), and [gfw_sar_vessel_detections\(\)](#)

Usage

```
gfw_test_shape
```

Format

A shapefile with a single polygon.

gfw_vessel_info	<i>Base function to get vessel information from API and convert response to tibble</i>
-----------------	--

Description

Base function to get vessel information from API and convert response to tibble

Usage

```
gfw_vessel_info(
  query = NULL,
  where = NULL,
  search_type = "search",
  ids = NULL,
  includes = c("AUTHORIZATIONS", "OWNERSHIP", "MATCH_CRITERIA"),
  match_fields = NULL,
  registries_info_data = c("ALL"),
  key = gfw_auth(),
  quiet = FALSE,
  print_request = FALSE,
  ...
)
```

Arguments

query	When search_type = "search", a length-1 vector with the identity variable of interest, MMSI, IMO, call sign or ship name.
where	When search_type = "search", an SQL expression to find the vessel of interest.
search_type	Type of vessel search to perform. Can be "search" (the default) or "id". (Note:"advanced" and "basic" are no longer in use as of gfwr 2.0.0.).
ids	When search_type = "id", a vector with the vesselId of interest.
includes	Enhances the response with new information, defaults to include all. "OWNERSHIP" returns ownership information "AUTHORIZATIONS" lists public authorizations for that vessel "MATCH_CRITERIA" adds information about the reason why a vessel is returned
match_fields	Optional. Allows to filter by matchFields levels. Possible values: "SEVERAL_FIELDS", "NO_MATCH", "ALL". Incompatible with where.
registries_info_data	when search_type == "id", gets all the registry objects, only the delta or the latest. "NONE" The API will return the most recent object only "DELTA" The API will return only the objects when the vessel changed one or more identity properties "ALL" The registryInfo array will return all objects we have in the vessel database
key	Character, API token. Defaults to <code>gfw_auth()</code> .
quiet	Boolean. Whether to print the number of events returned by the request and progress. Default is FALSE.
print_request	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string.
...	Other parameters, see API documentation.

Details

When search_type = "search" the search takes basic identity features like MMSI, IMO, callsign, shipname as inputs, using parameter "query". For more advanced SQL searches, use parameter "where". You can combine logic operators like AND, OR, =, >=, <, LIKE (for fuzzy matching). The id search allows the user to search using a GFW vesselId.

Examples

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

# Simple searches, using includes

gfw_vessel_info(query = 224224000, search_type = "search")
```

```
# Advanced search with where instead of query:

gfw_vessel_info(where = "ssvid = '441618000' OR imo = '9047271'",
search_type = "search")

# Vessel id search

gfw_vessel_info(search_type = "id",
ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01",
"6583c51e3-3626-5638-866a-f47c3bc7ef7c"))

all <- gfw_vessel_info(search_type = "id",
ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
registries_info_data = c("ALL"))

none <- gfw_vessel_info(search_type = "id",
ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
registries_info_data = c("NONE"))

delta <- gfw_vessel_info(search_type = "id",
ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
registries_info_data = c("DELTA"))

## End(Not run)
```

Index

* datasets

- gfw_marine_regions, [13](#)
- gfw_test_shape, [19](#)

- gfw_ais_fishing_hours, [2](#)
- gfw_ais_fishing_hours(), [12](#), [13](#), [19](#)
- gfw_ais_presence, [4](#)
- gfw_ais_presence(), [13](#), [19](#)
- gfw_auth, [6](#)
- gfw_auth(), [3](#), [5](#), [8](#), [11](#), [15](#), [17](#), [20](#)
- gfw_event, [7](#)
- gfw_event(), [13](#), [19](#)
- gfw_event_stats, [10](#)
- gfw_last_report, [12](#)
- gfw_last_report(), [12](#)
- gfw_marine_regions, [13](#)
- gfw_region_id, [14](#)
- gfw_region_id(), [3](#), [4](#), [8](#), [11](#), [13](#), [15](#), [16](#)
- gfw_regions, [15](#)
- gfw_regions(), [13](#), [15](#)
- gfw_sar_vessel_detections, [16](#)
- gfw_sar_vessel_detections(), [13](#), [19](#)
- gfw_test_shape, [19](#)
- gfw_vessel_info, [19](#)
- gfw_vessel_info(), [7](#), [11](#)