Package: readwfs (via r-universe)

May 15, 2026

Title Read Vector Features from Web Feature Services and Friends

Version 0.1.0

Description Read vector features from OGC Web Feature Services (WFS), OGC API Features (OAPIF), and ArcGIS REST FeatureServer/MapServer endpoints. Uses GDAL's OGR vector drivers via 'gdalraster' for connection handling and returns tibbles with 'wk' geometry columns. Provides service discovery, layer listing, capability inspection, and bbox-filtered feature retrieval. Aims to make navigating public vector web services less painful.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports gdalraster (>= 1.11.0), wk, tibble, methods

Suggests dplyr, geos, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/mdsumner/readwfs

BugReports https://github.com/mdsumner/readwfs/issues

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libxml2-dev libzstd-dev

Repository https://hypertidy.r-universe.dev

Date/Publication 2026-02-14 10:48:46 UTC

RemoteUrl https://github.com/hypertidy/readwfs

RemoteRef HEAD

RemoteSha 3bc88d6fd70dd8fdb726b2fad03a911e9ef27d4e

Contents

`wfs_example_bbox` *Example bounding boxes*

Description

Pre-configured bounding boxes for example areas.

Usage

```
wfs_example_bbox(area = c("sandy_bay", "hobart"))
```

Arguments

`area`	Character. One of:

- `"sandy_bay"`: Sandy Bay, Hobart – EPSG:28355 (MGA Zone 55), ~600m x 600m
- `"hobart"`: Greater Hobart – EPSG:28355, ~10km x 10km

Value

A named numeric vector (`xmin`, `ymin`, `xmax`, `ymax`).

Examples

```
wfs_example_bbox("sandy_bay")
wfs_example_bbox("hobart")
```

wfs_example_url *Example service endpoint URLs*

Description

Pre-configured URLs for example services used in documentation, vignettes, and testing.

Usage

```
wfs_example_url(service = c("list_tasmania", "esri_sample"))
```

Arguments

service	Character. One of: • "list_tasmania": Tasmania LIST open data WFS (parcels, vegetation, local government areas, hydro, transport, and more) • "esri_sample": Esri SampleWorldCities WFS (continents, cities – small dataset, always available)

Value

A character string with the service URL.

Examples

```
wfs_example_url("list_tasmania")
wfs_example_url("esri_sample")
```

wfs_fields *Get field (column) information for a layer*

Description

Returns a tibble describing each non-geometry field in a layer: name, type, width, and whether it's nullable.

Usage

```
wfs_fields(base_url, layer, driver = "auto", version = NULL, srs = NULL)
```

Arguments

base_url	Character. The service endpoint URL.
layer	Character. Layer name.
driver, version, srs	Passed to wfs_connection().

Value

A tibble::tibble with columns: `name`, `type`, `width`, `precision`, `is_nullable`.

Examples

```
## Not run:
url <- wfs_example_url("list_tasmania")
wfs_fields(url, "Public_OpenDataWFS:LIST_Cadastral_Parcels",
           version = "2.0.0", srs = "EPSG:28355")

## End(Not run)
```

`wfs_find_layers` *Search layer names by pattern*

Description

A convenience wrapper around `wfs_layers()` that filters results with `grep()`. Useful when a service has dozens or hundreds of layers.

Usage

```
wfs_find_layers(
  base_url,
  pattern,
  driver = "auto",
  version = NULL,
  srs = NULL,
  ignore.case = TRUE
)
```

Arguments

`base_url`	Character. The service endpoint URL. Can be a raw URL or a GDAL-prefixed connection string (e.g. `"WFS:https://..."`).
`pattern`	Character. A regular expression to match against layer names.
`driver`	Character. One of `"auto"`, `"WFS"`, `"OAPIF"`, `"ESRIJSON"`.
`version`	Character or `NULL`. WFS version.
`srs`	Character or `NULL`. Target SRS.
`ignore.case`	Logical. Passed to `grep()`.

Value

A character vector of matching layer names.

Examples

```
## Not run:
url <- wfs_example_url("list_tasmania")
wfs_find_layers(url, "CADASTRAL|TASVEG")

## End(Not run)
```

`wfs_layer_info` *Get layer metadata from a vector web service*

Description

Returns a tibble describing each layer: geometry type, feature count (if available), spatial extent, and spatial reference.

Usage

```
wfs_layer_info(
  base_url,
  layers = NULL,
  driver = "auto",
  version = NULL,
  srs = NULL
)
```

Arguments

`base_url`	Character. The service endpoint URL. Can be a raw URL or a GDAL-prefixed connection string (e.g. `"WFS:https://..."`).
`layers`	Character vector of layer names to inspect, or `NULL` to inspect all layers. Inspecting all layers can be slow for large services – consider using `wfs_find_layers()` first.
`driver`	Character. One of `"auto"`, `"WFS"`, `"OAPIF"`, `"ESRIJSON"`.
`version`	Character or `NULL`. WFS version.
`srs`	Character or `NULL`. Target SRS.

Value

A tibble::tibble with columns: `name`, `geom_column`, `geom_type`, `feature_count`, `xmin`, `ymin`, `xmax`, `ymax`, `srs_wkt`.

Examples

```
## Not run:
url <- wfs_example_url("list_tasmania")
wfs_layer_info(url, layers = "Public_OpenDataWFS:LIST_Cadastral_Parcels",
               version = "2.0.0", srs = "EPSG:28355")

## End(Not run)
```

`wfs_layers` *List available layers from a vector web service*

Description

Queries the service endpoint and returns the names of all available feature types / layers.

Usage

```
wfs_layers(base_url, driver = "auto", version = NULL, srs = NULL)
```

Arguments

`base_url`	Character. The service endpoint URL. Can be a raw URL or a GDAL-prefixed connection string (e.g. `"WFS:https://..."`).
`driver`	Character. One of `"auto"`, `"WFS"`, `"OAPIF"`, `"ESRIJSON"`.
`version`	Character or `NULL`. WFS version.
`srs`	Character or `NULL`. Target SRS.

Value

A character vector of layer names.

Examples

```
## Not run:
# Tasmania LIST WFS
wfs_layers(wfs_example_url("list_tasmania"))

# Esri sample WFS
wfs_layers(wfs_example_url("esri_sample"))

## End(Not run)
```

`wfs_read` *Read features from a vector web service*

Description

Fetches vector features from a WFS, OGC API Features, or ArcGIS REST endpoint and returns a tibble with a `wk::wkb` geometry column.

Usage

```
wfs_read(
  base_url,
  layer,
  bbox = NULL,
  max_features = 100,
  where = NULL,
  driver = "auto",
  version = NULL,
  srs = NULL,
  convert_linear = TRUE,
  promote_to_multi = FALSE
)
```

Arguments

`base_url`	Character. The service endpoint URL.
`layer`	Character. Layer name to read. Use `wfs_layers()` to discover available layers.
`bbox`	Numeric vector of length 4 (`xmin`, `ymin`, `xmax`, `ymax`) or `NULL` for no spatial filter. Coordinates should be in the SRS specified by `srs` (or the layer's native SRS if `srs` is `NULL`).
`max_features`	Integer or `NULL`. Maximum number of features to request. For WFS, passed as `count` in the URL. For other drivers, features are truncated after fetch.
`where`	Character or `NULL`. An OGR SQL `WHERE` clause to filter features by attribute.
`driver`	Character. Driver hint; see `wfs_connection()`.
`version`	Character or `NULL`. WFS version (e.g. `"2.0.0"`).
`srs`	Character or `NULL`. Target SRS as `"EPSG:XXXX"`.
`convert_linear`	Logical. If `TRUE` (default), convert circular arc geometries to linear approximations.
`promote_to_multi`	Logical. If `TRUE`, promote single-part geometries to multi-part. Default `FALSE`.

Value

A tibble::tibble with attribute columns and a `geometry` column of class `wk::wkb`.

Examples

```
## Not run:
# Tasmania LIST: cadastral parcels in Sandy Bay
parcels <- wfs_read(
  wfs_example_url("list_tasmania"),
  layer = "Public_OpenDataWFS:LIST_Cadastral_Parcels",
  bbox = wfs_example_bbox("sandy_bay"),
  srs = "EPSG:28355",
  max_features = 100
)
parcels
wk::wk_plot(parcels$geometry)

# Esri sample world cities
cities <- wfs_read(
  wfs_example_url("esri_sample"),
  layer = "esri:cities"
)

## End(Not run)
```

`wfs_services` *Catalogue of known public vector web services*

Description

Returns a tibble of known, tested public services that work with readwfs. This isn't exhaustive – it's a starting point. If you find a good one, open an issue.

Usage

```
wfs_services()
```

Value

A tibble::tibble with columns: `name`, `url`, `driver`, `region`, `description`, `srs`, `notes`.

Examples

```
wfs_services()

# Try one
## Not run:
svc <- wfs_services()
wfs_layers(svc$url[1])

## End(Not run)
```

Index