

# Package ‘darksky’

September 22, 2016

**Type** Package

**Title** Tools to Work with the Dark Sky API

**Version** 1.0.0

**Date** 2016-09-21

**Maintainer** Bob Rudis <bob@rud.is>

**Description** Provides programmatic access to the Dark Sky API  
<<https://darksky.net/dev/docs>>, which provides current or historical global  
weather conditions.

**URL** <https://github.com/hrbrmstr/darksky>

**Suggests** testthat

**Imports** stats, httr, dplyr, grid, gridExtra, gtable, ggplot2

**Depends** R (>= 3.1.0)

**License** MIT + file LICENSE

**BugReports** <https://github.com/hrbrmstr/darksky/issues>

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Author** Bob Rudis [aut, cre],  
Lasse Vilhelmsen [ctb]

**Repository** CRAN

**Date/Publication** 2016-09-22 09:22:49

## R topics documented:

darksky . . . . .	2
darksky_api_key . . . . .	2
get_current_forecast . . . . .	3
get_forecast_for . . . . .	4
plot.darksky . . . . .	5
print.darksky . . . . .	6

<b>Index</b>	<b>7</b>
--------------	----------

---

darksky	<i>Tools to Work with the Dark Sky API</i>
---------	--

---

**Description**

Programmatic access to the Dark Sky API <<https://darksky.net/dev/docs>>, which provides current or historical global weather conditions.

**Author(s)**

Bob Rudis (bob@rud.is)

---

darksky_api_key	<i>Get or set DARKSKY_API_KEY value</i>
-----------------	---

---

**Description**

The API wrapper functions in this package all rely on a Dark Sky API key residing in the environment variable DARKSKY\_API\_KEY. The easiest way to accomplish this is to set it in the ‘.Renviron’ file in your home directory.

**Usage**

```
darksky_api_key(force = FALSE)
```

**Arguments**

force	force setting a new Dark Sky API key for the current environment?
-------	---

**Value**

atomic character vector containing the Dark Sky API key

---

get\_current\_forecast    *Retrieve the current forecast (for the next week)*

---

## Description

The Dark Sky API lets you query for most locations on the globe, and returns:

1. current conditions
2. minute-by-minute forecasts out to 1 hour (where available)
3. hour-by-hour forecasts out to 48 hours
4. day-by-day forecasts out to 7 days

## Usage

```
get_current_forecast(latitude, longitude, units = "us", language = "en",  
    exclude = NULL, extend = NULL, add_json = FALSE, add_headers = FALSE,  
    ...)
```

## Arguments

latitude	forecast latitude (character, decimal format)
longitude	forecast longitude (character, decimal format)
units	return the API response in units other than the default Imperial unit
language	return text summaries in the desired language
exclude	exclude some number of data blocks from the API response. This is useful for reducing latency and saving cache space. This should be a comma-separated string (without spaces) including one or more of the following: (currently, minutely, hourly, daily, alerts, flags). Crafting a request with all of the above blocks excluded is exceedingly silly and not recommended. Setting this parameter to NULL (the default) does not exclude any parameters from the results.
extend	setting this parameter to hourly the API will return hourly data for the next seven days, rather than the next two.
add_json	add the raw JSON response to the object?
add_headers	add the return headers to the object?
...	pass through parameters to <code>httr::GET</code> (e.g. to configure ssl options or proxies)

## Details

If you wish to have results in something besides Imperial units, set `units` to one of (`si`, `ca`, `uk`). Setting `units` to `auto` will have the API select the relevant units automatically, based on geographic location. This value is set to `us` (Imperial) units by default.

If you wish to have text summaries presented in a different language, set `language` to one of (`ar`, `bs`, `de`, `es`, `fr`, `it`, `nl`, `pl`, `pt`, `ru`, `sv`, `tet`, `tr`, `x-pig-latin`, `zh`). This value is set to `en` (English) by default.

See the Options section of the official [Dark Sky API documentation](#) for more information.

Value

an darksky object that contains the original JSON response object (optionally), a list of named 'tbl\_df' 'data.frame' objects corresponding to what was returned by the API and (optionally) relevant response headers (cache-control, expires, x-forecast-api-calls, x-response-time).

Examples

```
## Not run:
tmp <- get_current_forecast(37.8267, -122.423)

## End(Not run)
```

---

get_forecast_for	Retrieve weather data for a specific place/time
------------------	---

---

Description

Query for a specific time, past or future (for many places, 60 years in the past to 10 years in the future).

Usage

```
get_forecast_for(latitude, longitude, timestamp, units = "us",
  language = "en", exclude = NULL, add_json = FALSE,
  add_headers = FALSE, ...)
```

Arguments

latitude	forecast latitude (character, decimal format)
longitude	forecast longitude (character, decimal format)
timestamp	should either be a UNIX time (that is, seconds since midnight GMT on 1 Jan 1970) or a string formatted as follows: [YYYY]-[MM]-[DD]T[HH]:[MM]:[SS] (with an optional time zone formatted as Z for GMT time or [+ -][HH][MM] for an offset in hours or minutes). For the latter format, if no timezone is present, local time (at the provided latitude and longitude) is assumed. (This string format is a subset of ISO 8601 time. An as example, 2013-05-06T12:00:00-0400.)
units	return the API response in units other than the default Imperial unit
language	return text summaries in the desired language
exclude	exclude some number of data blocks from the API response. This is useful for reducing latency and saving cache space. This should be a comma-separated string (without spaces) including one or more of the following: (currently, minutely, hourly, daily, alerts, flags). Crafting a request with all of the above blocks excluded is exceedingly silly and not recommended. Setting this parameter to NULL (the default) does not exclude any parameters from the results.
add_json	add the raw JSON response to the object?
add_headers	add the return headers to the object?
...	pass through parameters to http::GET (e.g. to configure ssl options or proxies)

**Details**

If you wish to have results in something besides Imperial units, set `units` to one of (si, ca, uk). Setting `units` to `auto` will have the API select the relevant units automatically, based on geographic location. This value is set to `us` (Imperial) units by default.

If you wish to have text summaries presented in a different language, set `language` to one of (ar, bs, de, es, fr, it, nl, pl, pt, ru, sv, tet, tr, x-pig-latin, zh). This value is set to `en` (English) by default.

See the Options section of the official [Dark Sky API documentation](#) for more information.

**Value**

an `darksky` object that contains the original JSON response object (optionally), a list of named `'tbl_df'` `'data.frame'` objects corresponding to what was returned by the API and (optionally) relevant response headers (`cache-control`, `expires`, `x-forecast-api-calls`, `x-response-time`).

**Examples**

```
## Not run:
tmp <- get_forecast_for(37.8267,-122.423, "2013-05-06T12:00:00-0400")

## End(Not run)
```

---

plot.darksky

*Plot method for darksky objects*


---

**Description**

Uses `ggplot2` & `grid.arrange` to produce graphs for `darksky` objects

**Usage**

```
## S3 method for class 'darksky'
plot(x, ..., readings = c("hourly", "minutely", "daily"))
```

**Arguments**

<code>x</code>	a <code>darksky</code> <code>x</code>
<code>readings</code>	specify which readings to plot. will plot all available by default
<code>...</code>	ignored

**Value**

frame grob

**Note**

only forecast/readings components of `x` that have more than one observation will be plotted

---

print.darksky	<i>Slightly more human-readable output for darksky objects</i>
---------------	--

---

**Description**

Slightly more human-readable output for darksky objects

**Usage**

```
## S3 method for class 'darksky'  
print(x, ...)
```

**Arguments**

x	a darksky object
...	ignored

# Index

`darksky`, [2](#)  
`darksky-package (darksky)`, [2](#)  
`darksky_api_key`, [2](#)  
  
`get_current_forecast`, [3](#)  
`get_forecast_for`, [4](#)  
  
`plot.darksky`, [5](#)  
`print.darksky`, [6](#)