
meteofrance-api

HACF

Jan 17, 2022

CONTENTS

1	Pour les francophones	3
2	For English speaking users	5
3	Credits	7
	Python Module Index	19
	Index	21

Client Python pour l'API Météo-France. | Python client for Météo-France API.

You will find English README content in the section *For English speaking users*.

Vous trouverez le contenu francophone du README dans la section *Pour les francophones*.

POUR LES FRANCOPHONES

1.1 Description

Ce package Python permet de gérer la communication avec l'API non publique de Météo-France utilisée par les applications mobiles officielles.

Le client permet:

- Rechercher des lieux de prévisions.
- Accéder aux prévisions météorologiques horaires ou quotidiennes.
- Accéder aux prévisions de pluie dans l'heure quand disponibles.
- Accéder aux alertes météo pour chaque département français et d'Andorre. Deux bulletins sont disponibles : un synthétique et un second avec l'évolution des alertes pour les prochaines 24 heures (exemple [ici](#)).

Ce package a été développé avec l'intention d'être utilisé par [Home-Assistant](#) mais il peut être utilisé dans d'autres contextes.

1.2 Installation

Pour utiliser le module Python `meteofrance` vous devez en premier installer le package en utilisant `pip` depuis PyPI:

```
$ pip install meteofrance-api
```

Vous pouvez trouver un exemple d'usage dans un module Python en regardant le [test d'intégration](#).

1.3 Contribuer

Les contributions sont les bienvenues. Veuillez consulter les bonnes pratiques détaillées dans [CONTRIBUTING.rst](#).

FOR ENGLISH SPEAKING USERS

2.1 Description

This Python package manages the communication with the private Météo-France API used by the official mobile applications.

The client allows:

- Search a forecast location.
- Fetch daily or hourly weather forecast.
- Fetch rain forecast within the next hour if available.
- Fetch the weather alerts or phenomenoms for each French department or Andorre. Two bulletin are available: one basic and an other advanced with the timelaps evolution for the next 24 hours (example [here](#)).

This package have been developed to be used with [Home-Assistant](#) but it can be used in other contexts.

2.2 Installation

To use the `meteofrance` Python module, you have to install this package first via `pip` from `PyPI`:

```
$ pip install meteofrance-api
```

You will find an example ot usage in a Python program in the [integration test](#).

2.3 Contributing

Contributions are welcomed. Please check the guidelines in [CONTRIBUTING.rst](#).

This project was generated from [@cjolowicz's Hypermodern Python Cookiecutter](#) template.

3.1 Reference

- *meteofrance*
- *meteofrance.model*

3.1.1 meteofrance

Météo-France API.

class `meteofrance_api.MeteoFranceClient` (*access_token=None*)

Proxy to the Météo-France REST API.

You will find methods and helpers to request weather forecast, rain forecast and weather alert bulletin.

get_forecast (*latitude, longitude, language='fr'*)

Retrieve the weather forecast for a given GPS location.

Results can be fetched in french or english according to the language parameter.

Parameters

- **latitude** (*float*) – Latitude in degree of the GPS point corresponding to the weather forecast.
- **longitude** (*float*) – Longitude in degree of the GPS point corresponding to the weather forecast.
- **language** (*str*) – Optional; If language is equal “fr” (default value) results will be in French. All other value will give results in English.

Returns A Forecast instance representing the hourly and daily weather forecast.

Return type `meteofrance_api.model.forecast.Forecast`

get_forecast_for_place (*place, language='fr'*)

Retrieve the weather forecast for a given Place instance.

Results can be fetched in french or english according to the language parameter.

Parameters

- **place** (*meteofrance_api.model.place.Place*) – Place class instance corresponding to a location.
- **language** (*str*) – Optional; If language is equal “fr” (default value) results will be in French. All other value will give results in English.

Returns A Forecast instance representing the hourly and daily weather forecast.

Return type *meteofrance_api.model.forecast.Forecast*

get_picture_of_the_day ()

Retrieve the picture of the day image URL & description.

Returns PictureOfTheDay instance with the URL and the description of the picture of the day.

Return type *meteofrance_api.model.picture_of_the_day.PictureOfTheDay*

get_rain (*latitude, longitude, language='fr'*)

Retrieve the next 1 hour rain forecast for a given GPS the location.

Results can be fetched in french or english according to the language parameter.

Parameters

- **latitude** (*float*) – Latitude in degree of the GPS point corresponding to the rain forecast.
- **longitude** (*float*) – Longitude in degree of the GPS point corresponding to the rain forecast.
- **language** (*str*) – Optional; If language is equal “fr” (default value) results will be in French. All other value will give results in English.

Returns A Rain instance representing the next hour rain forecast.

Return type *meteofrance_api.model.rain.Rain*

get_warning_current_phenomenoms (*domain, depth=0, with_costal_bulletin=False*)

Return the current weather phenomenoms (or alerts) for a given domain.

Parameters

- **domain** (*str*) – could be *france* or any metropolitan France department numbers on two digits. For some departments you can access an additional bulletin for coastal phenomenoms. To access it add *10* after the domain id (example: *1310*).
- **depth** (*int*) – Optional; To be used with *domain = 'france'*. With *depth = 0* the results will show only national sum up of the weather alerts. If *depth = 1*, you will have in addition, the bulletin for all metropolitan France department and Andorre
- **with_costal_bulletin** (*bool*) – Optional; If set to *True* (default is *False*), you can get the basic bulletin and coastal bulletin merged.

Returns A *warning.CurrentPhenomenons* instance representing the weather alert bulletin.

Return type *meteofrance_api.model.warning.CurrentPhenomenons*

get_warning_full (*domain, with_costal_bulletin=False*)

Retrieve a complete bulletin of the weather phenomenoms for a given domain.

For a given domain we can access the maximum alert, a timelaps of the alert evolution for the next 24 hours, a list of alerts and other metadatas.

Parameters

- **domain** (*str*) – could be *france* or any metropolitan France department numbers on two digits. For some departments you can access an additional bulletin for coastal phenomena. To access it add *10* after the domain id (example: *1310*).
- **with_costal_bulletin** (*bool*) – Optional; If set to True (default is False), you can get the basic bulletin and coastal bulletin merged.

Returns A warning.Full instance representing the complete weather alert bulletin.

Return type meteofrance_api.model.warning.Full

get_warning_thumbnail (*domain='france'*)

Retrieve the thumbnail URL of the weather phenomena or alerts map.

Parameters domain (*str*) – could be *france* or any metropolitan France department numbers on two digits.

Returns The URL of the thumbnail representing the weather alert status.

Return type str

search_places (*search_query, latitude=None, longitude=None*)

Search the places (cities) linked to a query by name.

You can add GPS coordinates in parameter to search places around a given location.

Parameters

- **search_query** (*str*) – A complete name, only a part of a name or a postal code (for France only) corresponding to a city in the world.
- **latitude** (*Optional[str]*) – Optional; Latitude in degree of a reference point to order results. The nearest places first.
- **longitude** (*Optional[str]*) – Optional; Longitude in degree of a reference point to order results. The nearest places first.

Returns A list of places (Place instance) corresponding to the query.

Return type List[meteofrance_api.model.place.Place]

3.1.2 meteofrance.model

Météo-France models for the REST API.

class meteofrance_api.model.**CurrentPhenomenons** (*raw_data*)

Class to access the results of a *warning/currentPhenomenons* REST API request.

For coastal department two bulletins are available corresponding to two different domains.

update_time

A timestamp (as integer) corresponding to the latest update of the phenomena.

end_validity_time

A timestamp (as integer) corresponding to expiration date of the phenomena.

domain_id

A string corresponding to the domain ID of the bulletin. Value is 'France' or a department number.

phenomenons_max_colors

A list of dictionaries with type of phenomena and the current alert level.

property domain_id

Return the domain ID of the phenomena.

property end_validity_time

Return the end of validity time of the phenomenoms.

get_domain_max_color()

Get the maximum level of alert of a given domain (class helper).

Returns An integer corresponding to the status code representing the maximum alert.

Return type int

merge_with_coastal_phenomenoms (*coastal_phenomenoms*)

Merge the classical phenomenoms bulleting with the coastal one.

Extend the phenomenomes_max_colors property with the content of the coastal weather alert bulletin.

Parameters **coastal_phenomenoms** (*meteofrance_api.model.warning.CurrentPhenomenoms*) – CurrentPhenomenoms instance corresponding to the coastal weather alert bulletin.

Return type None

property phenomenoms_max_colors

Return the list and colors of the phenomenoms.

property update_time

Return the update time of the phenomenoms.

class meteofrance_api.model.**Forecast** (*raw_data*)

Class to access the results of a *forecast* API request.

position

A dictionary with metadata about the position of the forecast place.

updated_on

A timestamp as int corresponding to the latest update date.

daily_forecast

A list of dictionaries to describe the daily forecast for the next 15 days.

forecast

A list of dictionaries to describe the hourly forecast for the next days.

probability_forecast

A list of dictionaries to describe the event probability forecast (rain, snow, freezing) for next 10 days.

today_forecast

A dictionary corresponding to the daily forecast for the current

day.

nearest_forecast

A dictionary corresponding to the nearest hourly forecast.

current_forecast

A dictionary corresponding to the hourly forecast for the current hour.

property current_forecast

Return the forecast of the current hour.

property daily_forecast

Return the daily forecast for the following days.

property forecast

Return the hourly forecast.

property nearest_forecast

Return the nearest hourly forecast.

property position

Return the position information of the forecast.

property probability_forecast

Return the wheather event forecast.

timestamp_to_locale_time (*timestamp*)

Convert timestamp in datetime in the forecast location timezone (Helper).

Parameters **timestamp** (*int*) – An integer to describe the UNIX timestamp.

Returns

Datetime instance corresponding to the timestamp with the timezone of the forecast location.

Return type datetime.datetime

property today_forecast

Return the forecast for today.

property updated_on

Return the update timestamp of the forecast.

class meteofrance_api.model.Full (*raw_data*)

This class allows to access the results of a *warning/full* API command.

For a given domain we can access the maximum alert, a timelaps of the alert evolution for the next 24 hours, and a list of alerts.

For coastal department two bulletins are available corresponding to two different domains.

update_time

A timestamp (as integer) corresponding to the latest update of the pheomenoms.

end_validity_time

A timestamp (as integer) corresponding to expiration date of the phenomenoms.

domain_id

A string corresponding do the domain ID of the bulletin. Value is 'France' or a department number.

color_max

An integer representing the maximum alert level in the domain.

timelaps

A list of dictionnaires corresponding to the schedule of each phenomenoms in the next 24 hours.

phenomenons_items

list of dictionnaires corresponding the alert level for each phenomenoms type.

property color_max

Return the color max of the domain.

property domain_id

Return the domain ID of the the full bulletin.

property end_validity_time

Return the end of validty time of the full bulletin.

merge_with_coastal_phenomenons (*coastal_phenomenoms*)

Merge the classical phenomenon bulletin with the coastal one.

Extend the `color_max`, `timelaps` and `phenomenons_items` properties with the content of the coastal weather alert bulletin.

Parameters `coastal_phenomenons` (*meteofrance_api.model.warning.Full*)
– Full instance corresponding to the coastal weather alert bulletin.

Return type None

property `phenomenons_items`

Return the phenomenom list of the domain.

property `timelaps`

Return the timelaps of each phenomenom for the domain.

property `update_time`

Return the update time of the full bulletin.

class `meteofrance_api.model.PictureOfTheDay` (*raw_data*)

Class to access the results of a *ImageJour/last* REST API request.

image_url

A string corresponding to the picture of the day URL.

image_hd_url

A string corresponding to the URL for the HD version of the picture of the day.

descriptpion

A string with the description of the picture of the day.

property `description`

Return the description of the picture of the day.

property `image_hd_url`

Return the image HD URL of the picture of the day.

property `image_url`

Return the image URL of the picture of the day.

class `meteofrance_api.model.Place` (*raw_data*)

Class to access the results of 'places' REST API request.

insee

A string corresponding to the INSEE ID of the place.

name

Name of the place.

lat

A float with the latitude in degree of the place.

lon

A float with the longitude in degree of the place

country

A string corresponding to the country code of the place.

admin

A string with the name of the administrative area ('Département' for France and Region for other countries).

admin2

A string corresponding to an administrative code ('Département' number for France)

postCode
A string corresponding to the ZIP code of location.

property admin
Return the admin of the place.

property admin2
Return the admin2 of the place.

property country
Return the country code of the place.

property insee
Return the INSEE ID of the place.

property latitude
Return the latitude of the place.

property longitude
Return the longitude of the place.

property name
Return the name of the place.

property postal_code
Return the postal code of the place.

class `meteofrance_api.model.Rain` (*raw_data*)
Class to access the results of 'rain' REST API request.

position
A dictionary with metadata about the position of the forecast place.

updated_on
A timestamp as int corresponding to the latest update date.

forecast
A list of dictionaries to describe the following next hour rain forecast.

quality
An integer. Don't know yet the usage.

property forecast
Return the rain forecast.

next_rain_date_locale ()
Estimate the date of the next rain in the Place timezone (Helper).

Returns

A datetime instance representing the date estimation of the next rain within the next hour. If no rain is expected in the following hour 'None' is returned.

The datetime use the location timezone.

Return type Optional[datetime.datetime]

property position
Return the position information of the rain forecast.

property quality
Return the quality of the rain forecast.

timestamp_to_locale_time (*timestamp*)
Convert timestamp in datetime with rain forecast location timezone (Helper).

Parameters `timestamp` (*int*) – An integer representing the UNIX timestamp.

Returns

A **datetime instance corresponding to the timestamp with the timezone of the** `rain` forecast location.

Return type `datetime.datetime`

property `updated_on`

Return the update timestamp of the rain forecast.

3.2 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the [MIT license](#) and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- [Source Code](#)
- [Documentation](#)
- [Issue Tracker](#)
- [Code of Conduct](#)

3.2.1 How to report a bug

Report bugs on the [Issue Tracker](#).

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

3.2.2 How to request a feature

Request features on the [Issue Tracker](#).

3.2.3 How to set up your development environment

You need Python 3.6+ and the following tools:

- Poetry
- Nox
- nox-poetry

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session:

```
$ poetry run python
```

3.2.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the `tests` directory, and are written using the `pytest` testing framework.

3.2.5 How to submit changes

Open a [pull request](#) to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

3.3 Contributor Covenant Code of Conduct

3.3.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

3.3.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

3.3.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

3.3.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

3.3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at contact@hacf.fr. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

3.3.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

3.3.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at <https://www.contributor-covenant.org/faq>. Translations are available at <https://www.contributor-covenant.org/translations>.

3.4 MIT License

Copyright (c) 2020 HACF Home Assistant Communauté Francophone

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

The software is provided “as is”, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

PYTHON MODULE INDEX

m

`meteofrance_api`, 7

`meteofrance_api.model`, 9

A

admin (*meteofrance_api.model.Place attribute*), 12
 admin() (*meteofrance_api.model.Place property*), 13
 admin2 (*meteofrance_api.model.Place attribute*), 12
 admin2() (*meteofrance_api.model.Place property*), 13

C

color_max (*meteofrance_api.model.Full attribute*), 11
 color_max() (*meteofrance_api.model.Full property*), 11
 country (*meteofrance_api.model.Place attribute*), 12
 country() (*meteofrance_api.model.Place property*), 13
 current_forecast (*meteofrance_api.model.Forecast attribute*), 10
 current_forecast() (*meteofrance_api.model.Forecast property*), 10
 CurrentPhenomenons (*class in meteofrance_api.model*), 9

D

daily_forecast (*meteofrance_api.model.Forecast attribute*), 10
 daily_forecast() (*meteofrance_api.model.Forecast property*), 10
 description() (*meteofrance_api.model.PictureOfTheDay property*), 12
 descritpion (*meteofrance_api.model.PictureOfTheDay attribute*), 12
 domain_id (*meteofrance_api.model.CurrentPhenomenons attribute*), 9
 domain_id (*meteofrance_api.model.Full attribute*), 11
 domain_id() (*meteofrance_api.model.CurrentPhenomenons property*), 9
 domain_id() (*meteofrance_api.model.Full property*), 11

E

end_validity_time (*meteofrance_api.model.CurrentPhenomenons attribute*), 9

end_validity_time (*meteofrance_api.model.Full attribute*), 11
 end_validity_time() (*meteofrance_api.model.CurrentPhenomenons property*), 9
 end_validity_time() (*meteofrance_api.model.Full property*), 11

F

Forecast (*class in meteofrance_api.model*), 10
 forecast (*meteofrance_api.model.Forecast attribute*), 10
 forecast (*meteofrance_api.model.Rain attribute*), 13
 forecast() (*meteofrance_api.model.Forecast property*), 10
 forecast() (*meteofrance_api.model.Rain property*), 13
 Full (*class in meteofrance_api.model*), 11

G

get_domain_max_color() (*meteofrance_api.model.CurrentPhenomenons method*), 10
 get_forecast() (*meteofrance_api.MeteoFranceClient method*), 7
 get_forecast_for_place() (*meteofrance_api.MeteoFranceClient method*), 7
 get_picture_of_the_day() (*meteofrance_api.MeteoFranceClient method*), 8
 get_rain() (*meteofrance_api.MeteoFranceClient method*), 8
 get_warning_current_phenomenoms() (*meteofrance_api.MeteoFranceClient method*), 8
 get_warning_full() (*meteofrance_api.MeteoFranceClient method*), 8

`get_warning_thumbnail()` (*meteofrance_api.MeteoFranceClient* method), 9

I

`image_hd_url` (*meteofrance_api.model.PictureOfTheDay* attribute), 12

`image_hd_url()` (*meteofrance_api.model.PictureOfTheDay* property), 12

`image_url` (*meteofrance_api.model.PictureOfTheDay* attribute), 12

`image_url()` (*meteofrance_api.model.PictureOfTheDay* property), 12

`insee` (*meteofrance_api.model.Place* attribute), 12

`insee()` (*meteofrance_api.model.Place* property), 13

L

`lat` (*meteofrance_api.model.Place* attribute), 12

`latitude()` (*meteofrance_api.model.Place* property), 13

`lon` (*meteofrance_api.model.Place* attribute), 12

`longitude()` (*meteofrance_api.model.Place* property), 13

M

`merge_with_coastal_phenomenons()` (*meteofrance_api.model.CurrentPhenomenons* method), 10

`merge_with_coastal_phenomenons()` (*meteofrance_api.model.Full* method), 11

`meteofrance_api` module, 7

`meteofrance_api.model` module, 9

`MeteoFranceClient` (*class in meteofrance_api*), 7

module `meteofrance_api`, 7
`meteofrance_api.model`, 9

N

`name` (*meteofrance_api.model.Place* attribute), 12

`name()` (*meteofrance_api.model.Place* property), 13

`nearest_forecast` (*meteofrance_api.model.Forecast* attribute), 10

`nearest_forecast()` (*meteofrance_api.model.Forecast* property), 10

`next_rain_date_locale()` (*meteofrance_api.model.Rain* method), 13

P

`phenomenons_items` (*meteofrance_api.model.Full* attribute), 11

`phenomenons_items()` (*meteofrance_api.model.Full* property), 12

`phenomenons_max_colors` (*meteofrance_api.model.CurrentPhenomenons* attribute), 9

`phenomenons_max_colors()` (*meteofrance_api.model.CurrentPhenomenons* property), 10

`PictureOfTheDay` (*class in meteofrance_api.model*), 12

`Place` (*class in meteofrance_api.model*), 12

`position` (*meteofrance_api.model.Forecast* attribute), 10

`position` (*meteofrance_api.model.Rain* attribute), 13

`position()` (*meteofrance_api.model.Forecast* property), 11

`position()` (*meteofrance_api.model.Rain* property), 13

`postal_code()` (*meteofrance_api.model.Place* property), 13

`postCode` (*meteofrance_api.model.Place* attribute), 12

`probability_forecast` (*meteofrance_api.model.Forecast* attribute), 10

`probability_forecast()` (*meteofrance_api.model.Forecast* property), 11

Q

`quality` (*meteofrance_api.model.Rain* attribute), 13

`quality()` (*meteofrance_api.model.Rain* property), 13

R

`Rain` (*class in meteofrance_api.model*), 13

S

`search_places()` (*meteofrance_api.MeteoFranceClient* method), 9

T

`timelaps` (*meteofrance_api.model.Full* attribute), 11

`timelaps()` (*meteofrance_api.model.Full* property), 12

`timestamp_to_locale_time()` (*meteofrance_api.model.Forecast* method), 11

`timestamp_to_locale_time()` (*meteofrance_api.model.Rain* method), 13

`today_forecast` (*meteofrance_api.model.Forecast* attribute), 10

`today_forecast()` (*meteofrance_api.model.Forecast* property), 11

U

`update_time` (*meteofrance_api.model.CurrentPhenomenons attribute*), 9

`update_time` (*meteofrance_api.model.Full attribute*), 11

`update_time()` (*meteofrance_api.model.CurrentPhenomenons property*), 10

`update_time()` (*meteofrance_api.model.Full property*), 12

`updated_on` (*meteofrance_api.model.Forecast attribute*), 10

`updated_on` (*meteofrance_api.model.Rain attribute*), 13

`updated_on()` (*meteofrance_api.model.Forecast property*), 11

`updated_on()` (*meteofrance_api.model.Rain property*), 14