meteofrance-api

HACF

Dec 06, 2023

CONTENTS

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*.

CHAPTER

ONE

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.

CHAPTER

TWO

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 availabe: 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.

CHAPTER

THREE

CREDITS

This project was generated from @cjolowicz's Hypermodern Python Cookiecutter template.

3.1 Reference

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.

Parameters
access_token (*str | None*) –

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 intance representing the hourly and daily weather forecast.

Return type
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** (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 intance representing the hourly and daily weather forecast.

Return type
Forecast

`get_observation`(*latitude*, *longitude*, *language='fr'*)

Retrieve the weather observation 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
An Observation instance.

Return type
Observation

`get_observation_for_place`(*place*, *language='fr'*)

Retrieve the weather observation for a given Place instance.

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

Parameters

- **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
An Observation intance.

Return type
Observation

`get_picture_of_the_day`(*domain='france'*)

Retrieve the picture of the day image URL & description.

Parameters
domain (*str*) – could be *france*

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

Return type
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
Rain

`get_warning_current_phenomenoms`(*domain*, *depth=0*, *with_coastal_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 natinal sum up of the weather alerts. If depth = 1, you will have in addition, the bulletin for all metropolitan France department and Andorre
- **with_coastal_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
CurrentPhenomenons

`get_warning_dictionary`(*language='fr'*)

Retrieves the meteorological dictionary from the Météo-France API.

This dictionary includes information about various meteorological phenomena and color codes used for weather warnings.

Parameters
language (*str*) – The language in which to retrieve the dictionary data. Default is 'fr' for French. Other language codes can be used if supported by the API.

Returns

An object containing structured data about
meteorological phenomena and warning color codes. It has two main attributes: 'phenomenons' (list of PhenomenonDictionaryEntry) and 'colors' (list of ColorDictionaryEntry).

Return type
WarningDictionary

`get_warning_full`(*domain*, *with_coastal_bulletin=False*)

Retrieve a complete bulletin of the weather phenomenons 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 phenomenoms. To access it add *10* after the domain id (example: *1310*).
- **with_coastal_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
Full

`get_warning_thumbnail`(*domain='france'*)

Retrieve the thumbnail URL of the weather phenomenoms 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 arround 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** (*str | None*) – Optional; Latitude in degree of a reference point to order results. The nearest places first.
- **longitude** (*str | None*) – 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[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 avalaible corresponding to two different domains.

Parameters
raw_data (*WarnningCurrentPhenomenonsData*) –

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.

phenomenons_max_colors

A list of dictionnaries with type of phenomenoms and the current alert level.

property domain_id: str

Return the domain ID of the phenomenoms.

property end_validity_time: int

Return the end of validty 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_phenomenons(*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 (CurrentPhenomenons) – CurrentPhenomenons instance corresponding to the coastal weather alert bulletin.

Return type
None

property phenomenons_max_colors: List[PhenomenonMaxColor]

Return the list and colors of the phenomenoms.

property update_time: int

Return the update time of the phenomenoms.

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

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

Parameters
raw_data (*ForecastData*) –

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: Dict[str, Any]

Return the forecast of the current hour.

property daily_forecast: List[Dict[str, Any]]

Return the daily forecast for the following days.

property forecast: List[Dict[str, Any]]

Return the hourly forecast.

property nearest_forecast: Dict[str, Any]

Return the nearest hourly forecast.

property position: Dict[str, Any]

Return the position information of the forecast.

property probability_forecast: List[Dict[str, Any]]

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

property today_forecast: Dict[str, Any]

Return the forecast for today.

property updated_on: int

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 avalaible corresponding to two different domains.

Parameters
raw_data (*WarnningFullData*) –

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 dictionnaries corresponding to the schedule of each phenomenoms in the next 24 hours.

phenomenons_items

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

property color_max: int

Return the color max of the domain.

property domain_id: str

Return the domain ID of the the full bulletin.

property end_validity_time: int

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_phenomenoms (Full) – Full instance corresponding to the coastal weather alert bulletin.

Return type
None

property phenomenons_items: List[PhenomenonMaxColor]

Return the phenomenom list of the domain.

property timelaps: List[Dict[str, Any]]
Return the timelaps of each phenomenom for the domain.

property update_time: int
Return the update time of the full bulletin.

class meteofrance_api.model.Observation(*raw_data*)
Class to access the results of an *observation* API request.

Parameters
raw_data (*ObservationData*) –

timezone
The observation timezone

time
The time at which the observation was made

temperature
The observed temperature (°C)

wind_speed
The observed wind speed (km/h)

wind_direction
The observed wind direction (°)

wind_icon
An icon ID illustrating the observed wind direction

weather_icon
An icon ID illustrating the observed weather condition

weather_description
A description of the observed weather condition

property temperature: float | None
Returns the observed temp (°C).

property time_as_datetime: datetime | None
Returns the time at which the observation was made.

property time_as_string: str | None
Returns the time at which the observation was made.

property timezone: str | None
Returns the observation timezone.

property weather_description: str | None
Returns a description of the observed weather condition.

property weather_icon: str | None
Returns an icon ID illustrating the observed weather condition.

property wind_direction: int | None
Returns the observed wind direction (°).

property wind_icon: str | None
Returns an icon ID illustrating the observed wind direction.

property wind_speed: float | None

Returns the observed wind speed (km/h).

class meteofrance_api.model.PictureOfTheDay(*raw_data*)

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

Parameters
raw_data (*PictureOfTheDayData*) –

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.

descritpion

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

property description: str

Return the description of the picture of the day.

property image_url: str

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.

Parameters
raw_data (*PlaceData*) –

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 correponding to an administrative code ('Département' number for France)

postCode

A string corresponding to the ZIP code of location.

property admin: str | None

Return the admin of the place.

property admin2: str | None

Return the admin2 of the place.

property country: str

Return the country code of the place.

property insee: str | None

Return the INSEE ID of the place.

property latitude: float

Return the latitude of the place.

property longitude: float

Return the longitude of the place.

property name: str

Return the name of the place.

property postal_code: str | None

Return the postal code of the place.

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

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

Parameters
raw_data (*RainData*) –

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: List[Dict[str, Any]]

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
datetime | None

property position: Dict[str, Any]

Return the position information of the rain forecast.

property quality: int

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

property updated_on: int

Return the update timestamp of the rain forecast.

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

A class to represent and manipulate the Meteo France meteorological dictionary data.

Parameters
raw_data (*WarningDictionaryData*) –

get_phenomenon_name_by_id(phenomenon_id

int): Returns the name of the phenomenon for the given ID.

get_color_name_by_id(color_id

int): Returns the name of the color for the given ID.

get_color_by_id(*color_id*)

Retrieves a warning color based on its ID.

Parameters
color_id (*int*) – The ID of the color.

Returns
The the color object if found, otherwise returns None.

Return type
ColorDictionaryEntry | None

get_color_name_by_id(*color_id*)

Retrieves the name of a warning color based on its ID.

Parameters
color_id (*int*) – The ID of the color.

Returns
The name of the color if found, otherwise returns None.

Return type
str | None

get_phenomenon_by_id(*phenomenon_id*)

Retrieves a meteorological phenomenon based on its ID.

Parameters
phenomenon_id (*int*) – The ID of the meteorological phenomenon.

Returns
The phenomenon if found, otherwise returns None.

Return type
PhenomenonDictionaryEntry | None

`get_phenomenon_name_by_id`(*phenomenon_id*)

Retrieves the name of a meteorological phenomenon based on its ID.

Parameters
phenomenon_id (*int*) – The ID of the meteorological phenomenon.

Returns
The name of the phenomenon if found, otherwise returns None.

Return type
str | None

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 3.8+ 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

PYTHON MODULE INDEX

m

INDEX

I

L

M

N

O

P

Q

R

S

T

U

W