



Norwegian
Meteorological
Institute

Frost

A REST API for point observations

2025-02-28 Jo Asplin (joa@met.no)

frost.met.no



WHAT IS FROST

HOW TO USE FROST

API REFERENCE

CHANGELOG

What is Frost?

The Frost API provides [free access](#) to [MET Norway's](#) archive of historical weather and climate data. This data includes quality controlled daily, monthly, and yearly measurements of temperature, precipitation, and wind data. Other information, like metadata about weather stations, is also available through the API.

The Frost API is primarily for developers who need to develop scripts or applications that access MET Norway's archive of historical weather and climate data. Please read about the [terms of use](#) for the API, as well as [MET's privacy policy statement](#)

Please email observasjon@met.no if you have questions about Frost, and it will be directed to the appropriate people. **Please read the [FAQs](#) before emailing**, we will keep updating them based on questions we receive. We are looking into setting up a discussion forum in the future, as the old email list has been discontinued.

Create a client ID



WHAT IS FROST

HOW TO USE FROST

API REFERENCE

CHANGELOG

EXAMPLES

AUTHENTICATION

API CONCEPTS

BROWSE WEATHER ELEMENTS

DATA CLARIFICATIONS

How to use Frost

Below you will find a basic introduction to help you learn to use Frost. Please use the submenu for more information.

Basic introduction

Frost is a [RESTful API](#) that provides access to MET Norway's archive of historical weather and climate data.

To access the API you need to [create a user](#), all you need is an email address ([MET's privacy policy statement](#)). You will get a client ID and client secret, which you should save in a safe place. Most users will only need the client ID, but if you require access to data that is not open then you need to use the client secret for [OAuth2](#).

CREATE A USER

There are several different parts of the API, out of which you can retrieve different types of data. [Sources/](#) for instance can be used to find available sources for a particular area or find meta information about a particular source. [Observations/AvailableTimeSeries/](#) can be used to find out what types of [weather elements](#) (types of observations) are available for a particular station or time range. [Observations/](#) can be used to retrieve observations. In order to request observations you must specify 3 things:

- Source[s]
- Reference Time
- Element[s]

Get latest observations from Blindern

URL:

https://frost.met.no/observations/v0.jsonld?sources=SN18700&referencetime=latest&elements=air_temperature

Terminal/cURL:

```
curl --user 0aa125e4-e2d1-478d-bc17-248a720261a4:  
"https://frost.met.no/observations/v0.jsonld?sources=SN18700&referencetime=latest&elements  
=air_temperature"
```

Python:

```
import requests # See https://requests.readthedocs.io/en/latest/
import json
import sys

url = 'https://frost.met.no/observations/v0.jsonld'

query_params = {
    'sources': 'SN18700',
    'referencetime': 'latest',
    'elements': 'air_temperature',
}

client_id = '0aa125e4-e2d1-478d-bc17-248a720261a4'

response = requests.get(url, query_params, auth=(client_id, ''))

if response.status_code == 200:
    print(json.dumps(response.json(), indent=4))
else:
    print('error:')
    print(f'\tstatus code: {response.status_code}')
    print(f'\tcontent: {response.text}')
    sys.exit(1)
```

Response body

```
{
  . . .
  "data": [
    {
      "sourceId": "SN18700:0",
      "referenceTime": "2025-02-26T07:17:00.000Z",
      "observations": [
        {
          "elementId": "air_temperature",
          "value": 0.9,
          "unit": "degC",
          "level": {
            "levelType": "height_above_ground",
            "unit": "m",
            "value": 10
          },
          "timeOffset": "PT0H",
          "timeResolution": "PT10M",
          "timeSeriesId": 0,
          "performanceCategory": "C",
          "exposureCategory": "1",
          "qualityCode": 2
        },
        . . .
      ]
    },
    . . .
  ]
}
```

Parsing response body in Python

```
. . .
import dateutil.parser  # <---

query_params = {
    'sources': 'SN18700',
    'referencetime': '2025-02-26T00/2025-02-26T06:00',  # <---
    'elements': 'air_temperature',
}


. . .

if response.status_code == 200:
    for item in response.json()['data']:
        iso8701 = item['referenceTime']  # obs time in ISO 8601 format
        dt = dateutil.parser.isoparse(iso8601)
        tstamp = int(dt.timestamp())  # obs time as secs since 1970-01-01T00:00:00Z
        val = item['observations'][0]['value']  # obs value
        sys.stdout.write('{} {} {} \n'.format(iso8601, tstamp, val))
else:
    . . .
```

result 

2025-02-26T00:00:00.000Z	1740528000	2.6
2025-02-26T00:01:00.000Z	1740528060	2.6
2025-02-26T00:02:00.000Z	1740528120	2.6
2025-02-26T00:03:00.000Z	1740528180	2.6
2025-02-26T00:04:00.000Z	1740528240	2.6
2025-02-26T00:05:00.000Z	1740528300	2.6
2025-02-26T00:06:00.000Z	1740528360	2.5
2025-02-26T00:07:00.000Z	1740528420	2.5
2025-02-26T00:08:00.000Z	1740528480	2.6
2025-02-26T00:09:00.000Z	1740528540	2.5
2025-02-26T00:10:00.000Z	1740528600	2.5
2025-02-26T00:11:00.000Z	1740528660	2.5
2025-02-26T00:12:00.000Z	1740528720	2.5
2025-02-26T00:13:00.000Z	1740528780	2.5
2025-02-26T00:14:00.000Z	1740528840	2.5
2025-02-26T00:15:00.000Z	1740528900	2.5
2025-02-26T00:16:00.000Z	1740528960	2.5
. . .		

Finding relevant sources (typically weather stations)

 Meteorologisk institutt

WHAT IS FROST HOW TO USE FROST **API REFERENCE** CHANGELOG

Frost

Created by The Norwegian Meteorological Institute
[Norwegian Licence for Open Government Data \(NLOD\) 2.0](#)

elements Show/Hide List Operations Expand Operations

climatenormals Show/Hide List Operations Expand Operations

sources Show/Hide List Operations Expand Operations

GET /sources/v0.{format} Get metadata for Frost API sources.

[Implementation Notes](#)

Get metadata for the source entities defined in the Frost API. Use the query parameters to filter the set of sources returned. Leave the query parameters blank to select **all** sources.



Parameters

Parameter	Value	Description	Parameter Type	Data Type
ids	<input type="text"/>	The Frost API source ID(s) that you want metadata for. Enter a comma-separated list to select multiple sources. For sources of type SensorSystem or RegionDataset, the source ID must be of the form <prefix><int> where <prefix> is SN for SensorSystem and TR, NR, GR, or GF for RegionDataset. The integer following the prefix may contain wildcards, e.g. SN18*7* matches both SN18700 and SN18007.	query	string
types	<input type="text" value="v"/>	The type of Frost API source that you want metadata for.	query	string
elements	<input type="text"/>	If specified, only sources for which observations are available for all of these elements may be included in the result. Enter a comma-separated list of search filters .	query	string
geometry	<input type="text"/>	Get Frost API sources defined by a specified geometry. Geometries are specified as either nearest(POINT(...)) or POLYGON(...) using WKT ; see the reference section on the Geometry Specification for documentation and examples. If the nearest() function is specified, the output will include the distance in kilometers from the reference point.	query	string
nearestmaxcount	<input type="text"/>	The maximum number of sources returned when using nearest(POINT(...)) for geometry. The default value is 1.	query	string



validtime	<input type="text"/>	If specified, only sources that have been, or still are, valid/applicable during some part of this interval may be included in the result. Specify <date>/<date>, <date>/now, <date>, or now, where <date> is of the form YYYY-MM-DD, e.g. 2017-03-06. The default is 'now', i.e. only currently valid/applicable sources are included.	query	string
name	<input type="text"/>	If specified, only sources whose 'name' attribute matches this search filter may be included in the result.	query	string
country	<input type="text"/>	If specified, only sources whose 'country' or 'countryCode' attribute matches this search filter may be included in the result.	query	string
county	<input type="text"/>	If specified, only sources whose 'county' or 'countyld' attribute matches this search filter may be included in the result.	query	string
municipality	<input type="text"/>	If specified, only sources whose 'municipality' or 'municipalityld' attribute matches this search filter may be included in the result.	query	string
wmoid	<input type="text"/>	If specified, only sources whose 'wmold' attribute matches this search filter may be included in the result.	query	string
stationholder	<input type="text"/>	If specified, only sources whose 'stationholders' attribute contains at least one name that matches this search filter may be included in the result.	query	string

Finding relevant elements

(a.k.a. weather- and climate *parameters*)



Weather and Climate Elements

(NOTE: The input for the table below is the response from [this request](#).)

English

Show 10 entries

Search: max(air_temp

id	oldElementCodes	category	name	description	unit	sensorLevels	status
best_estimate_max(air_temperature P1D)	TAX	Temperature	Homogenised maximum air temperature (24 h)	Homogenised daily temperature maximum.	degC	type: height_above_ground, unit: m, default: 2, values: 2	CF-name
max(air_temperature P1D)	TAX, TAXD, TAX_24, X1TAX	Temperature	Maximum air temperature (24 h)	Highest recorded air temperature per 24 hours	degC	type: height_above_ground, unit: m, default: 2, values: 2	CF-name
max(air_temperature P1M)	TAX	Temperature	Maximum air temperature (month)	Highest recorded air temperature per month	degC	type: height_above_ground, unit: m, default: 2, values: 2	CF-name
max(air_temperature P1Y)	TAX	Temperature	Maximum air temperature (year)	Highest recorded air temperature per year	degC	type: height_above_ground, unit: m, default: 2, values: 2	CF-name

values to use in the Frost *elements* query parameter