Weather-API – interface description

Version: v=1

Contents

Weather-API – interface description..............................................................................................................1
1  Introduction .................................................................................................................................................2
2  Registration...............................................................................................................................................2
3  Basics .......................................................................................................................................................2
   3.1  Version number ...................................................................................................................................2
   3.2  API-Requests .....................................................................................................................................2
4  Authentication ..........................................................................................................................................2
5  Search ......................................................................................................................................................3
   6  Weather ................................................................................................................................................5
   6.1  Current Weather / package=current .................................................................................................6
   6.2  Daily forecast / package=daily8, package=daily14 ..........................................................................7
   6.3  Interval-based forecast / package=hourly72, package=day6parts .........................................................9
   6.4  Weather situation of the previous day / package=yesterday .............................................................14
   6.5  Weather forecast as text ...................................................................................................................17
7  Other formats ..........................................................................................................................................18
   7.1  XML-format .......................................................................................................................................18
8  Weather condition encoding ...................................................................................................................18
9  Sample code .............................................................................................................................................19
   9.1  PHP sample code for search request ..............................................................................................19
   9.2  PHP sample code for weather data request ...................................................................................20
   9.3  PHP sample code for text request ...................................................................................................22
10  Example of generating a checksum .....................................................................................................23
   10.1  How to generate a checksum for a search request ......................................................................23
   10.2  How to generate a checksum for a weather-API package request .............................................24
1 Introduction

The Weather-API is a tool to access current weather information as well as forecasts for any location worldwide. The information is linked to distinct location-specific identifiers (geo-IDs). These geo-IDs can be retrieved through a search function via geo coordinates, location names and zip codes (for Germany only), which can also be accessed through the account. The API is protected via Authentication. Different weather packages are available, to ensure a variety of data sets, parameters and applications.

2 Registration

When your account is activated, WetterOnline will provide you with access details consisting of an user-ID and a secret key. The account details must not be shared with third parties.

3 Basics

3.1 Version number

The current version is v=1. This parameter is part of each API-request.

3.2 API-Requests

API-requests are made via http-GET-method. The API will return a data file in JSON format

Procedure to access weather data:

1. Search
   Search for desired location / request for geo-ID
   Syntax:
   https://api.wetteronline.de/geo?name=<locationname/zipcode(Germany)>&v=<versionnumber>
   https://api.wetteronline.de/geo?lat=<latitude>&lon=<longitude>&v=<versionnumber>

2. requesting weather data
   Accessing the weather package of choice for one or more geo-IDs.
   Syntax:
   https://api.wetteronline.de/weather?package=<packagename>&gid=<geo-ID>&v=<versionnumber>

Every request has to include additional parameters for authentication purposes

4 Authentication

The API is protected via an authentication method.

To generate a valid request it is essential to have a secret key, all parameters of the request, the user-ID and the current date and time. The user-ID has to be encoded in Base64, the parameter name is uid. The parameter name for the time stamp is date and has to be in the format YYYY-MM-DD-hh as well as in UTC time.
Using the parameter value and the secret key, a checksum is created (parameter name: `checksum`) and added to the request. All parameters then have to be URL-encoded.

A checksum is generated as follows:

1. Sort all parameters alphabetically by parameter name.
   
   Syntax:  
   
   https://api.wetteronline.de/weather?package=<packagename>& gid=<geo-ID>&v=<versionnumber>&uid=<Base64(user-ID)>&date=<YYYY-MM-DD-hh>  
   
   alphabetic order: date, gid, package, uid, v

2. Put the values of these parameters together separated by “|” (Pipe).
   
   <YYYY-MM-DD-hh>|<geo-ID>|< packagename>|< Base64(user-ID)>|< versionnumber >

3. Add the secret key, again separated by “|”.
   
   <YYYY-MM-DD-hh>|<geo-ID>|<packagename>|< Base64(user-ID)|< versionnumber >|<secret key>

4. Generate a Base64-encoded MD5-hash for the byte-sequence of this string (use raw binary format, not hexadezimal form). Then convert the result into a Base64-encoded string.

5. The resulting string is URL encoded and inserted as the value for the parameter checksum.

   https://api.wetteronline.de/weather? package=<packagename>& gid=<geo-ID>&v=<versionnumber>&uid=<Base64(user-ID)&date=<YYYY-MM-DD-hh>&checksum=<checksum>

An example of how to generate a checksum can be found in the attachment.

**Note:**
The date has to be in UTC time. You can use https://api.wetteronline.de/weather?package=date to obtain the current date (no authentication is necessary for this request).

### 5 Search

To access weather information for a specific location, you must know the geo-ID first. This information can be obtained using the search function. It enables the user to find any place worldwide via geo-coordinates. The German location name and the zip code (for locations in Germany) can also be used.

The search request is made via:

https://api.wetteronline.de/geo?name=< locationname/zipcode(Germany)>&v=<versionnumber>

https://api.wetteronline.de/geo?lat=<latitude>&lon=<longitude>&v=< versionnumber >

<table>
<thead>
<tr>
<th>parameter name</th>
<th>meaning</th>
<th>valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>location name (in German) or zip code</td>
<td>alpha-numerical, UTF-8</td>
</tr>
<tr>
<td>lat</td>
<td>latitude</td>
<td>float, up to 4 significant figures 90.0, ... -90.0</td>
</tr>
<tr>
<td>lon</td>
<td>longitude</td>
<td>float, up to 4 significant figures 180.0, ... -180.0</td>
</tr>
</tbody>
</table>
As described in chapter 4 (Authentication) a valid request also has to contain version number, user-ID, current date in UTC and MD5 checksum.

https://api.wetteronline.de/geo?name=<locationname>/zipcode(Germany)&v=<versionnumber>&uid=<user-ID>&date=<YYYY-MM-DD-hh>&checksum=<checksum>
https://api.wetteronline.de/geo?lat=latitude&lon=longitude&v=<versionnumber>&uid=<user-ID>&date=<YYYY-MM-DD-hh>&checksum=<checksum>

The output of the search request is a string. By means of a JSON parser it can be converted into a JSON object.

The output file contains the following parameters:

<table>
<thead>
<tr>
<th>parameter name</th>
<th>meaning</th>
<th>valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>locationName</td>
<td>location name (in German)</td>
<td>alpha-numerical</td>
</tr>
<tr>
<td>subLocationName</td>
<td>name of city district</td>
<td>alpha-numerical</td>
</tr>
<tr>
<td>gid</td>
<td>WetterOnline geo-ID</td>
<td>5 digits, alpha-numerical</td>
</tr>
<tr>
<td>subStateID</td>
<td>abbreviation of the federal state (no ISO-code)</td>
<td>alpha-numerical</td>
</tr>
<tr>
<td>subStateName</td>
<td>name of the federal state</td>
<td>alpha-numerical</td>
</tr>
<tr>
<td>stateID</td>
<td>abbreviation of the country (no ISO-code)</td>
<td>2 digits, alpha-numerical</td>
</tr>
<tr>
<td>stateName</td>
<td>country name</td>
<td>alpha-numerical</td>
</tr>
<tr>
<td>latitude</td>
<td>latitude of location</td>
<td>float, up to 4 significant figures</td>
</tr>
<tr>
<td>longitude</td>
<td>longitude of location</td>
<td>float, up to 4 significant figures</td>
</tr>
<tr>
<td>altitude</td>
<td>height above sea level in meters</td>
<td>integer</td>
</tr>
<tr>
<td>zip</td>
<td>zip code</td>
<td>5 digits, integer</td>
</tr>
</tbody>
</table>

Example: Search for Neukölln
The output is structured as follows:
```json
[
    {
        "locationName":"Berlin",
        "subLocationName":"Neukölln",
        "gid":"10382",
        "subStateID":"BER",
        "subStateName":"Berlin",
        "stateID":"DL",
        "stateName":"Deutschland",
        "latitude":52.5166,
        "longitude":13.3999,
        "altitude":34,
        "zip":"10965",
    }
]
```
In case of an unknown value the default "null" delivered.

If the search function is used with the additional parameter `function=url`, the result also contains the URL to the WetterOnline page of the specific location.

The location names can also be given in the different languages. To use this feature, the parameter `lang=<Language as ISO 639-1 Code>` has to be added to the search link.

If no valid search result can be found, the array only contains the entry "match": "no".

If one or more valid result can be obtained, the output consists of one or more JSON objects with the last JSON object containing the entry "match": "yes".

**NOTE: The search function only accepts requests in UTF-8 format**

Geo-IDs are fix and do not change. Hence, it could be useful to save the geo-IDs of recurring locations within the algorithm. This way there is no need to re-use the search function for each weather data request for these specific locations.

An example of using the search mechanism correctly, can be found at the end of this document.

### 6 Weather

The standard request for weather information is:

https://api.wetteronline.de/weather?gid=<geo-ID>&package=<packagename>&v=<versionnumber>

There are several packages to request different weather data sets. To access the correct information, the parameter `package` has to be part of the API request. The packages are *current*, *daily8*, *daily14*, *day6parts*, *hourly72* and *yesterday*.

Additionally all requests have to contain a geo-ID (parameter name: `gid`). The correct geo-ID can be identified with the help of a search request.

All packages can be accessed for only one or up to 20 geo-IDs. Requests for multiple geo-IDs should be separated by comma. If a request contains more than 20 geo-IDs, only the first 20 geo-IDs are processed.

Geo-IDs are fix and do not change. Hence, it could be useful to save the geo-IDs of recurring locations within the algorithm. This way there is no need to re-use the search function for each weather data request for these specific locations.

As described in chapter 4 (Authentication) a valid request also has to contain version number, user-ID, current date in UTC and MD5 checksum.

The output is a JSON format, which is structured as follows:

```json
{
   "match": "yes"
}
```

// one object or an array of objects containing weather information for this geo-ID
A data object can in itself also contain an array of different data objects. The following chapters describe structure, content and parameters of each available weather data package. An example for accessing the weather information can be found at the end of this document.

### 6.1 Current Weather / package=current

The API-request for accessing the current weather is:

https://api.wetteronline.de/weather?package=current&gid=<geo-ID>&v=<versionnumber>

As described in chapter 4 ([Authentication](#)) a valid request also has to contain version number, user-ID, current date in UTC and MD5 checksum.

The output contains the following parameters for the current weather situation:

<table>
<thead>
<tr>
<th>parameter name</th>
<th>meaning</th>
<th>valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wm</td>
<td>weather condition key</td>
<td>see chapter 7</td>
</tr>
<tr>
<td>tt_C</td>
<td>current temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tta_C</td>
<td>wind chill temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>dd_dir</td>
<td>wind direction in sectors</td>
<td>n, nw, w, sw, s, so, o, no, 0 (baffling wind)</td>
</tr>
<tr>
<td>dd_deg</td>
<td>wind direction in degree</td>
<td>integer between 0 (calm) and 360 (north) in steps of 5</td>
</tr>
<tr>
<td>ff_ms</td>
<td>wind speed in m/s</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>ff_bft</td>
<td>wind speed in Beaufort</td>
<td>Integer</td>
</tr>
</tbody>
</table>
fx_kmh  
maximum wind gust in km/h  
integer

fx_bft  
maximum wind gust in Beaufort  
integer

pp_hpa  
surface pressure in hPa  
float, one significant figure

rh  
relative humidity in percent  
integer between 0 and 100 in steps of 5

pop  
probability of precipitation in percent  
integer between 0 and 100 in steps of 5

rr_mm  
amount of precipitation in mm  
float, one significant figure

rad_wm2  
global radiation in W/m²  
integer in steps of 5

The weather data can be requested for one or up to 20 locations (geo-IDs). The data format is JSON.

The following is the example of an output file for the package current:

```json
{
  "10382" : {
    "data" : {
      "dd_deg" : 140,
      "dd_dir" : "so",
      "ff_bft" : 2,
      "ff_ms" : 2.90,
      "fx_bft" : 4,
      "fx_kmh" : 20,
      "pop" : 5,
      "pp_hpa" : 1016.0,
      "rad_wm2" : 945,
      "rh" : 90,
      "rr_mm" : 0.0,
      "tt_C" : 25.20,
      "tta_C" : 25.20,
      "wm" : "so___"
    },
    "meta" : {
      "local_date" : "2014-05-21 11:39:57"
    },
    "info" : {
      "locations" : [ "10382" ],
      "type" : "city"
    }
  }
}
```

6.2 Daily forecast / package=daily8, package=daily14

The request for accessing weather information in daily resolution is:

https://api.wetteronline.de/weather?package=daily8&gid=<geo-ID>&v=<versionnumber>

or

https://api.wetteronline.de/weather?package=daily14&gid=<geo-ID>&v=<versionnumber>

The weather information can be obtained for the next 8 or 14 days.

As with the other cases, version number, user-ID, current date in UTC and MD5 checksum have to be added to create a valid request (Authentication).
The output file contains the following weather parameters for each local day:

<table>
<thead>
<tr>
<th>parameter name</th>
<th>meaning</th>
<th>valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>wm</td>
<td>weather condition key</td>
<td>see chapter 7</td>
</tr>
<tr>
<td>tn_C</td>
<td>minimum temperature (of the night previous to the specified day) in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tg_C</td>
<td>minimum ground temperature (5 cm above ground, of the night previous to the specified day) in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tx_C</td>
<td>maximum temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>dd_dir</td>
<td>wind direction in sectors (daily average)</td>
<td>n, nw, w, sw, s, so, o, no, 0 (baffling wind)</td>
</tr>
<tr>
<td>dd_deg</td>
<td>wind direction in degree (daily average)</td>
<td>integer between 0 (calm) and 360 (north) in steps of 5</td>
</tr>
<tr>
<td>ff_ms</td>
<td>wind speed in m/s (daily average)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>ff_bft</td>
<td>wind speed in Beaufort (daily average)</td>
<td>integer</td>
</tr>
<tr>
<td>fx_kmh</td>
<td>maximum wind gust in km/h</td>
<td>integer</td>
</tr>
<tr>
<td>fx_bft</td>
<td>maximum wind gust in Beaufort</td>
<td>integer</td>
</tr>
<tr>
<td>pp_hpa</td>
<td>surface pressure in hPa (daily average)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>rh</td>
<td>relative humidity in percent (daily average)</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>pop</td>
<td>probability of precipitation in percent</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>rr_mm</td>
<td>amount of precipitation in mm</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>pd_h</td>
<td>duration of precipitation in hours</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>rad_wm2</td>
<td>global radiation in W/m²</td>
<td>integer in steps of 5</td>
</tr>
<tr>
<td>sd_h</td>
<td>sunshine duration in hours</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>sr</td>
<td>sunrise time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>ss</td>
<td>sunset time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>mr</td>
<td>moonrise time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>ms</td>
<td>moonset time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>date</td>
<td>date (local time)</td>
<td>YYYY-MM-DD</td>
</tr>
<tr>
<td>weekday</td>
<td>day of the week (local time)</td>
<td>Montag, ..., Sonntag</td>
</tr>
</tbody>
</table>
An example of an output file obtained from a daily package request (abridged):

```json
{
  "10382": {
    "data": [
      {
        "date": "2014-03-25",
        "dd_deg": 310,
        "dd_dir": "nw",
        "ff_bft": 2,
        "ff_ms": 2.9,
        "ff_kmh": 34,
        "ms": "02:47",
        "ms": "11:59",
        "pd_h": 1,
        "pop": 60,
        "pp_hpa": 1009.4,
        "rad_wm2": 125,
        "rr_mm": 0.8,
        "sd_h": 1.5,
        "se": "05:58",
        "ss": "18:28",
        "tg_C": 1.8,
        "tn_C": 1.9,
        "tx_C": 8.2,
        "weekday": "Dienstag",
        "wm": "bdr1__"
      },
      {SECOND DAY
        ...
      },
      {LAST DAY
        }
    ],
    "meta": {
      "local_date": "2014-03-25 08:32:45"
    }
  },
  "info": {
    "type": "city"
  }
}
```

The JSON object may contain data for up to 20 geo-IDs:

```json
{
  Geo-ID1: {...}, Geo-ID2: {...}, info: {...}
}
```

Every geo-ID object holds the array `data`. This array in turn contains a JSON object for each day, starting with the current day. The parameter `local_day` (date and time in local time) is the time stamp of the most recent data update.

### 6.3 Interval-based forecast / package=hourly72, package=day6parts

The request for interval based forecasts is:

[https://api.wetteronline.de/weather?package={hourly72|day6parts}&gid=<geo-ID>](https://api.wetteronline.de/weather?package={hourly72|day6parts}&gid=<geo-ID>)
These packages consist of daily weather information (see package=daily8) and additional weather data for certain periods of time within the local day.

- **hourly72** contains hourly weather information for the next 3 days (72 hours), starting with the current day at 0.00 a.m. local time.
- **day6parts** contains information for each 6-hour period within the next 6 days. The first time interval also starts at the current day at 0.00 a.m. local time.

The data structure is identical to the forecast on daily basis. However the array data is extended by an array **periods**. This array contains JSON objects with weather information for either each hour (package=hourly72) or each 6-hour period (package=day6parts).

**period_duration** describes the length of the time interval in hours. The starting point of the interval is defined through the parameter **hour**. For example, if hour=3 the interval starts at 3 a.m. local time. The name of the period is given by the parameter **periodname**.

The output contains the following weather information:

<table>
<thead>
<tr>
<th>parameter name</th>
<th>meaning</th>
<th>valid value</th>
</tr>
</thead>
<tbody>
<tr>
<td>wm</td>
<td>weather condition key</td>
<td>see chapter 7</td>
</tr>
<tr>
<td>tn_C</td>
<td>minimum temperature (of the night previous to the specified day) in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tg_C</td>
<td>minimum ground temperature (5 cm above ground, of the night previous to the specified day) in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tx_C</td>
<td>maximum temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>dd_dir</td>
<td>wind direction in sectors (daily average)</td>
<td>n, nw, w, sw, s, so, o, no, 0 (baffling wind)</td>
</tr>
<tr>
<td>dd_deg</td>
<td>wind direction in degree (daily average)</td>
<td>integer between 0 (calm) and 360 (north) in steps of 5</td>
</tr>
<tr>
<td>ff_ms</td>
<td>wind speed in m/s (daily average)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>ff_bft</td>
<td>wind speed in Beaufort (daily average)</td>
<td>integer</td>
</tr>
<tr>
<td>fx_kmh</td>
<td>maximum wind gust in km/h</td>
<td>integer</td>
</tr>
<tr>
<td>fx_bft</td>
<td>maximum wind gust in Beaufort</td>
<td>integer</td>
</tr>
<tr>
<td>pp_hpa</td>
<td>surface pressure in hPa (daily average)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>rh</td>
<td>relative humidity in percent (daily average)</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>pop</td>
<td>probability of precipitation in percent</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>rr_mm</td>
<td>amount of precipitation in mm</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>pd_h</td>
<td>duration of precipitation in hours</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>rad_WM2</td>
<td>global radiation in W/m² (daily average)</td>
<td>integer in steps of 5</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Format</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>sd_h</td>
<td>sunshine duration in hours</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>sr</td>
<td>sunrise time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>ss</td>
<td>sunset time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>mr</td>
<td>moonrise time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>ms</td>
<td>moonset time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>date</td>
<td>date (local time)</td>
<td>YYYY-MM-DD</td>
</tr>
<tr>
<td>weekday</td>
<td>day of the week von date (local time)</td>
<td>Montag, ..., Sonntag</td>
</tr>
<tr>
<td>periods</td>
<td>contains an array of additional weather data, valid for the time interval defined by the parameters hour and period_duration.</td>
<td></td>
</tr>
</tbody>
</table>

**Contents of "periods":**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>wm</td>
<td>weather condition key</td>
<td>see chapter 7</td>
</tr>
<tr>
<td>tt_C</td>
<td>temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tta_C</td>
<td>wind chill temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>dd_dir</td>
<td>wind direction in sectors (average over period_duration)</td>
<td>n, nw, w, sw, s, so, o, no, 0 (baffling wind)</td>
</tr>
<tr>
<td>dd_deg</td>
<td>wind direction in degree (average over period_duration)</td>
<td>integer between 0 (calm) and 360 (north) in steps of 5</td>
</tr>
<tr>
<td>ff_ms</td>
<td>wind speed in m/s (average over period_duration)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>ff_bft</td>
<td>wind speed in Beaufort (average over period_duration)</td>
<td>integer</td>
</tr>
<tr>
<td>fx_kmh</td>
<td>maximum wind gust in km/h</td>
<td>integer</td>
</tr>
<tr>
<td>fx_bft</td>
<td>maximum wind gust in Beaufort</td>
<td>integer</td>
</tr>
<tr>
<td>pp_hpa</td>
<td>surface pressure in hPa (average over period_duration)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>rh</td>
<td>relative humidity in percent (average over period_duration)</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>pop</td>
<td>probability of precipitation in percent</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>rr_mm</td>
<td>amount of precipitation in mm</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>pd_min* / pd_h**</td>
<td>duration of precipitation in minutes* / in hours**</td>
<td>integer/float, one significant figure</td>
</tr>
<tr>
<td>rad_wm2</td>
<td>global radiation in W/m² (average over period_duration)</td>
<td>integer in steps of 5</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>sd_min* / sd_h**</td>
<td>Sunshine duration in minutes* / in hours**</td>
<td>Integer/float, one significant figure</td>
</tr>
<tr>
<td>hour</td>
<td>Starting point of the interval (hour in local time)</td>
<td>Integer</td>
</tr>
<tr>
<td>period_duration</td>
<td>Length of the interval (in hours)</td>
<td>Integer</td>
</tr>
<tr>
<td>periodname</td>
<td>Name of the interval</td>
<td></td>
</tr>
</tbody>
</table>

* applies for package=hourly72  
** applies for package=day6parts

As above, **authentication** is needed for a valid request.

An example of data output of the package hourly72 (abridged):

```json
{
  "10382": {
    "data": [
    {
      "date": "2014-03-25",
      "dd_deg": 310,
      "dd_dir": "nw",
      "ff_bft": 2,
      "ff_ms": 2.9,
      "fx_bft": 5,
      "fx_kmh": 34,
      "ms": "02:47",
      "ms": "11:59",
      "pd_min": 0,
      "period_duration": 1,
      "periodname": "00:00",
      "pop": 35,
      "pp_hpa": 1011.5,
      "rad_wm2": 0,
      "rh": 85,
      "sd_min": 0,
      "sd_h": 1.5,
      "tt_C": 4.5,
      "tta_C": 1.9,
      "wm": "md____
    },
    { "HOUR": 2 }
    ...
    { "HOUR": 24 }
    
    "pop": 60,
    "pp_hpa": 1009.4,
    "rad_wm2": 125,
    "rr_mm": 1.1,
    "sd_h": 1.5,
    "ss": "05:58",
    "ss": "18:28",
    "tg_C": 1.8,
    "tn_C": 1.9,
    "tx_C": 8.2,
    }
  }
}
```
An example of data output of the package day6parts (abridged):

```json
"10382": {
  "data": [
    {
      "date": "2014-03-25",
      "dd_deg": 310,
      "dd_dir": "nw",
      "ff_bft": 2,
      "ff_ms": 2.9,
      "fx_bft": 5,
      "fx_kmh": 34,
      "ms": "02:47",
      "mt": "11:59",
      "pd_h": 1,
      "periods": [null]
    },
    {
      "dd_deg": 335,
      "dd_dir": "nw",
      "ff_bft": 2,
      "ff_ms": 2.2,
      "fx_bft": 3,
      "fx_kmh": 14,
      "hour": 0,
      "pd_h": 0,
      "period_duration": 6,
      "periodname": "Nacht",
      "pop": 35,
      "pp_hpa": 1010.4,
      "rad_wm2": 0,
      "rh": 95,
      "rr_mm": 0,
      "sd_h": 0,
      "sr": "05:58",
      "ss": "18:28",
      "tt_C": 1.8,
      "tn_C": 1.9,
      "tx_C": 8.2,
      "weekday": "Dienstag",
      "wm": "bws1__"
    },
    {... INTERVAL 2}, ...
  ]
},
"pop": 60,
"pp_hpa": 1009.4,
"rad_wm2": 125,
"tt_C": 1.8,
"tn_C": 1.9,
"tx_C": 8.2,
"weekday": "Dienstag",
"wm": "bws1__"
},
{"DAY 2}, ...\{DAY 6\}
},
"meta": null
},
"info": {
"type": "city"
}
```

6.4 Weather situation of the previous day / package=yesterday

The request for weather data of the previous day is:
https://api.wetteronline.de/weather?package=yesterday&gid=<geo-ID>

This package contains analysed weather information for the previous day as daily values and in hourly resolution.

The data structure is identical to the one used in the interval-based forecast packages. The array data is extended by an array periods. This array contains JSON objects with weather information for each hour of the previous day.

period_duration describes the length of the time interval in hours. The starting point of the interval is defined through the parameter hour. For example, if hour=3 the interval starts at 3 a.m. local time. The name of the period is given by the parameter periodname.

The output contains the following data:

Daily weather information for the previous day:

<table>
<thead>
<tr>
<th>parameter name</th>
<th>meaning</th>
<th>valid value</th>
</tr>
</thead>
<tbody>
<tr>
<td>wm</td>
<td>weather condition key</td>
<td>see chapter 7</td>
</tr>
<tr>
<td>tn_C</td>
<td>minimum temperature (of the night previous to the specified day) in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tg_C</td>
<td>minimum ground temperature (5 cm above ground, of the night previous to the specified day) in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tx_C</td>
<td>maximum temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>dd_dir</td>
<td>wind direction in sectors (daily average)</td>
<td>n, nw, w, sw, s, so, o, no, 0 (baffling wind)</td>
</tr>
<tr>
<td>dd_deg</td>
<td>wind direction in degree (daily average)</td>
<td>integer between 0 (calm) and 360 (north) in steps of 5</td>
</tr>
<tr>
<td>ff_ms</td>
<td>wind speed in m/s (daily average)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>ff_bft</td>
<td>wind speed in Beaufort (daily average)</td>
<td>integer</td>
</tr>
<tr>
<td>fx_kmh</td>
<td>maximum wind gust in km/h</td>
<td>integer</td>
</tr>
<tr>
<td>fx_bft</td>
<td>maximum wind gust in Beaufort</td>
<td>integer</td>
</tr>
<tr>
<td>pp_hpa</td>
<td>surface pressure in hPa (daily average)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>rh</td>
<td>relative humidity in percent (daily average)</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>pop</td>
<td>probability of precipitation in percent</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>rr_mm</td>
<td>amount of precipitation in mm</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>pd_h</td>
<td>duration of precipitation in hours</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>rad_wm2</td>
<td>global radiation in W/m² (daily average)</td>
<td>integer in steps of 5</td>
</tr>
<tr>
<td>sd_h</td>
<td>sunshine duration in hours</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>sr</td>
<td>sunrise time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>ss</td>
<td>sunset time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>mr</td>
<td>moonrise time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>ms</td>
<td>moonset time (local time)</td>
<td>hh:mm</td>
</tr>
<tr>
<td>date</td>
<td>date (local time)</td>
<td>YYYY-MM-DD</td>
</tr>
<tr>
<td>weekday</td>
<td>day of the week von date (local time)</td>
<td>Montag, ..., Sonntag</td>
</tr>
<tr>
<td>periods</td>
<td>contains an array of additional weather data, valid for the time interval defined by the parameters hour and period_duration.</td>
<td></td>
</tr>
</tbody>
</table>

Contents of "periods":

<table>
<thead>
<tr>
<th>wm</th>
<th>weather condition key</th>
<th>see chapter 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>tt_C</td>
<td>temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>tta_C</td>
<td>wind chill temperature in °C</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>dd_dir</td>
<td>wind direction in sectors (average over period_duration)</td>
<td>n, nw, w, sw, s, so, o, no, 0 (baffling wind)</td>
</tr>
<tr>
<td>dd_deg</td>
<td>wind direction in degree (average over period_duration)</td>
<td>integer between 0 (calm) and 360 (north) in steps of 5</td>
</tr>
<tr>
<td>ff_ms</td>
<td>wind speed in m/s (average over period_duration)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>ff_bft</td>
<td>wind speed in Beaufort (average over period_duration)</td>
<td>integer</td>
</tr>
<tr>
<td>fx_kmh</td>
<td>maximum wind gust in km/h</td>
<td>integer</td>
</tr>
<tr>
<td>fx_bft</td>
<td>maximum wind gust in Beaufort</td>
<td>integer</td>
</tr>
<tr>
<td>pp_hpa</td>
<td>surface pressure in hPa (average over period_duration)</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>rh</td>
<td>relative humidity in percent (average over period_duration)</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>pop</td>
<td>probability of precipitation in percent</td>
<td>integer between 0 and 100 in steps of 5</td>
</tr>
<tr>
<td>rr_mm</td>
<td>amount of precipitation in mm</td>
<td>float, one significant figure</td>
</tr>
<tr>
<td>pd_min</td>
<td>duration of precipitation in minutes</td>
<td>integer</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>rad_wm2</td>
<td>global radiation in W/m² (average over period_duration)</td>
<td>integer in steps of 5</td>
</tr>
<tr>
<td>sd_min</td>
<td>sunshine duration in minutes</td>
<td>integer</td>
</tr>
<tr>
<td>hour</td>
<td>starting point of the interval (hour in local time)</td>
<td>integer</td>
</tr>
<tr>
<td>period_duration</td>
<td>length of the interval (in hours)</td>
<td>integer</td>
</tr>
<tr>
<td>periodname</td>
<td>name of the interval</td>
<td>00:00, ..., 23:00</td>
</tr>
</tbody>
</table>

Again, the request is only valid with correct authentication.

An example of data output of the package yesterday (abridged):

```json
{
    "10382": {
      "data": {
        "date": "2014-03-24",
        "dd_deg": 310,
        "dd_dir": "nw",
        "ff_bft": 2,
        "ff_ms": 2.9,
        "fx_bft": 5,
        "fx_kmh": 34,
        "ms": "02:47",
        "ms": "11:59",
        "pd_h": 1,
        "periods": [
          {
            "dd_deg": 340,
            "dd_dir": "n",
            "ff_bft": 2,
            "ff_ms": 2,
            "fx_bft": 3,
            "fx_kmh": 14,
            "hour": 0,
            "pd_min": 0,
            "period_duration": 1,
            "periodname": "00:00",
            "pop": 35,
            "pp_hpa": 1011.5,
            "rad_wm2": 0,
            "rh": 85,
            "rr_mm": 0,
            "sd_min": 0,
            "tt_C": 4.5,
            "tta_C": 1.9,
            "wm": "md_____"
          },
          { HOUR: 2 },
          ...
        ]
      },
      "pop": 60,
      "pp_hpa": 1009.4,
      "rad_wm2": 125,
      "rr_mm": 1.1,
      "sd_h": 1.5,
      "ss": "05:58",
      "ss": "18:28"
    }
  }
```
6.5 Weather forecast texts

Please note that the weather texts are available in German language only.

To obtain a weather forecast text the package text can be used. The standard request is:
https://api.wetteronline.de/weather?package=text&gid=<geo-ID>&locationname=<locationname>&v=<versionnumber>

The request has to contain a geo-ID (parameter name: gid). If the location name mentioned in the text is to be the name of a city district, this name can be specified in the parameter locationname.

Geo-ID and location name can be obtained by the search function. The location name can be found in subLocationName. If it is null, the parameter location name can be left out or replaced by the name under locationName. It is essential to only set valid names for the parameter locationname. In case of doubt, the parameter locationname should not be filled.

A valid request always has to contain user-ID, current date in UTC and MD5 checksum (see chapter 4 Authentication in the main document).

If the parameter locationname is used, it also has to be recognised for the generation of the checksum.

The data format is JSON.
The file structure is as follows:

```json
{
  "text12hours": "<text12hours>" // forecast text for the next 12 hours
  "text3days": "<text3days>" // forecast text for the next 3 days
}
```

The forecast text for the next 12 hours (text12hours) is not yet available. Until it is, the value is null.

**NOTE: Only requests in UTF-8 format are valid.**

Example:
The forecast text for Neukölln is requested. The geo-ID 10382 and the city district name Neukölln are known from the search result.
Hence the request is:
https://api.wetteronline.de/weather?package=text&gid=10382&locationname=Neukölln&v=1

1 The parameters necessary for authentication (uid, date and checksum) are omitted in favour of clarity.
The data output is:

```json
{
    "text12hours": null,
    "text3days": "Mal Wolken, mal Sonne – mit beidem ist in den nächsten Tagen beim Wetter in der Region Neukölln zu rechnen. Vor allem am Donnerstag zeigt sich das Wetter im Raum Neukölln vielfach sonnig. Am Donnerstag muss vereinzelt mit Schauern gerechnet werden. Es wird kühler in der Region Neukölln: Bis Donnerstag gehen die Tagestemperaturen auf 17 Grad zurück. Vor allem am Donnerstag weht ein zum Teil starker Wind aus westlicher Richtung."
}
```

If the request doesn’t contain the parameter `locationname`, the location mentioned in the text would be Berlin instead of Neukölln. Of course both are valid options.

7 Other formats

The data format can be changed using the parameter `format`. Possible formats are JSON and XML. All packages (with the exception of date) are then delivered in the requested format. If the parameter format is used, it also has to be included is the checksum.

7.1 XML-format

With `format=xml` the data has XML-Format. An separate interface description for the WetterOnline Weather-API in XML format is also available.

8 Weather condition encoding

The weather condition (parameter name: `wm`) is described by an icon and a short text. A classification of the weather conditions with their corresponding texts can be found here:

In German:
https://api.wetteronline.de/weather?package=symbol&format=text

In English:
https://api.wetteronline.de/weather?package=symbol&format=text&lang=en

The weather condition icons can be downloaded as ZIP-Files in different sizes (40x28, 50x35 or 60x42 pixel):

https://api.wetteronline.de/weather?package=symbol&format=png&size=40x28
https://api.wetteronline.de/weather?package=symbol&format=png&size=50x35
https://api.wetteronline.de/weather?package=symbol&format=png&size=60x42

The weather condition key always consists of six alphanumeric figures. The first two describe cloudiness and day- or night time. The others (figures 3-6) give information about the nature and intensity of the precipitation.

Weather conditions without precipitation have an underline character “_” as the 3rd to 6th figure.

The first two figures with information about cloudiness and day- or night time:

<table>
<thead>
<tr>
<th>Cloudiness</th>
<th>Day symbol</th>
<th>Night symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overcast sky</td>
<td>bd</td>
<td>md</td>
</tr>
<tr>
<td>Cloudy</td>
<td>bw</td>
<td>mw</td>
</tr>
</tbody>
</table>
Various clouds | wb | mb
Mostly sunny | ms | mm
Clear sky | so | mo
Fog | nb
Partly fog | ns | nm

Fog, mostly sunny and clear sky do not exist with any precipitation indicators. All other conditions can be enhanced with rain, snow or thunderstorm indicators:

<table>
<thead>
<tr>
<th>Weather</th>
<th>light / isolated</th>
<th>medium</th>
<th>heavy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rain</td>
<td>r1</td>
<td>r2</td>
<td>r3</td>
</tr>
<tr>
<td>Showers</td>
<td>s1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sleet</td>
<td>sr1</td>
<td>sr2</td>
<td>sr3</td>
</tr>
<tr>
<td>Sleetshowers</td>
<td>srs1</td>
<td>srs2</td>
<td></td>
</tr>
<tr>
<td>Snow</td>
<td>sn1</td>
<td>sn2</td>
<td>sn3</td>
</tr>
<tr>
<td>Snowshowers</td>
<td>sns1</td>
<td>sns2</td>
<td></td>
</tr>
<tr>
<td>Thunder with snow*</td>
<td></td>
<td>sg</td>
<td></td>
</tr>
<tr>
<td>Thunderstorms</td>
<td>g1</td>
<td>g2</td>
<td>g3</td>
</tr>
<tr>
<td>Freezing rain</td>
<td>gr1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Thunderstorms with snow have one intensity only.

These keys are added to the cloudiness keys. The weather condition key always consist of 6 figures, hence underline characters ("_") are used to fill up the keys with fewer indicators.

Examples:
The key for isolated thunderstorms on an overcast sky bdg1__.
The key for isolated snow showers on a cloudy sky is bwsns1.

# Sample code

## 9.1 PHP sample code for search request

```php
<?php
header('Content-Type: text/html; charset=utf-8');

// WetterOnline Geo-API exemplary request

// basic path for geo-API request
$basepath = "https://api.wetteronline.de/geo?";

// individual UserID
$uid = "bWFpbEBleGFtcGxlLmNvbQ==";

// individual secret key for the account of "UserID"
$secret = "TmF0/HJsaWNoIGtlaW4gZWNodGVzIFNlY3JldCE=";

// for this example Bonn is the requested location
$name = "Bonn";
// if more than one location is searched, a second name can be defined
// $name = "Aal";

// if one of the required parameters is missing: end
if (isset($basepath) == false || isset($uid) == false || isset($secret) == false ||
```
isset($name) == false )
{
    echo "at least one of the required parameters is missing";
    exit();
}

// date - required for checksum
$date = new DateTime("now", new DateTimeZone('UTC'));

// compile query-parameter
$data = array(
    // current date, format: 2014-04-11-09
    'date' => $date->format('Y-m-d-H'),
    // UserID
    'uid' => $uid,
    // name of search location
    'name' => $name
);

// array sort keys alphabetically by parameter name:
ksort($data);

/*************************
calculate checksum ***********/
data[$checksum] = base64_encode(md5_implode("|", $data) . "$" . $secret, true));

// build query string
$qs = http_build_query($data);

// construct path for API-request
$apipath = $basepath . $qs;

// do API-request
$result = @file_get_contents($apipath);

// if the request was not successful, the HTTP-Response is returned
if ($result == false) exit("Error: ". $http_response_header[0]);

// convert result in JSON object
$json = json_decode($result);

/*************************/
/********************** Debugging Output *********************************/
echo "<a href="$apipath" target="_blank">$apipath</a><br>
<pre>
print_r($json);
</pre>

// end of program
exit();
?>

9.2 PHP sample code for weather data request

<?php
header('Content-Type: text/html; charset=utf-8');

/**************************
* WetterOnline Weather-API exemplary request
***************************/
// basic path for API request
$basepath = "https://api.wetteronline.de/weather?";

// required package (available current, hourly72, day6parts, daily8)
$package = "daily8";

// individual UserID
$uid = "bWFpbEBleGFtcGxlLmNvbQ==";

// individual secret key for the account of "UserID"
$secret = "TmF0/HJsaWNoIGtlaW4gZWNodGVzIFNlY3JldCE=";

// for this example the weather for Bonn is requested and the obtained geo-ID is used
$gid = "10518";

// if more than one location is required, the geo-ID should be comma separated
// $gid = "10518,10513,10400";

// if one of the required parameters is missing: end
if (isset($basepath) == false || isset($package) == false || isset($uid) == false || isset($secret) == false || isset($gid) == false)
{
    echo "at least one of the required parameters is missing";
    exit();
}

// date – required for checksum
$date = new DateTime("now", new DateTimeZone('UTC'));

// compile query-parameter
$data = array(  // current date, format: 2014-04-11-09
'date' => $date->format('Y-m-d-H'),
// required weather package
'package' => $package,
// UserID
'uid' => $uid,
// Geo-ID of the required location
'gid' => $gid,
);

// array sort keys alphabetically by parameter name:
ksort($data);

/******************** calculate checksum ********************
- put sorted parameter values together, separated by |
- attach secret key, separated by |
- hash string in md5_binary
- convert md5-hashed string in base64 encoding
************************************************************************/
$data["checksum"] = base64_encode(md5_implode("", $data) . "" . $secret, true));

// build query string
$qs = http_build_query($data);

// construct path for API-request
$apipath = $basepath . $qs;

// do API-request
$result = @file_get_contents($apipath);

// if the request was not successful, the HTTP-Response is returned
if ($result == false) exit("Error: " . $http_response_header[0]);

// convert result in JSON object
$json = json_decode($result);

/******************** Debugging Output ********************

// test-link to see the generated API-request
echo "<a href=""$apipath"" target=""_blank"">$apipath</a><br><br>
";
// simple test output of the API response
echo "<pre>";
print_r($json);
echo "</pre>";

// end of program
exit();

?>

9.3 PHP sample code for text request

```php
<?php
header('Content-Type: text/html; charset=utf-8');

/******************************************************************************
* WetterOnline Weather-API examplary request of forecast text
******************************************************************************

// basic path for API request
$basepath = "https://api.wetteronline.de/weather?";

// required package
$package = "text";

// individual UserID
$uid = "bWFpbEBleGFtcGxlLmNvbQ==";

// individual secret key für the account UserID
$secret = "TmF0/HJsaWNoIGtlaW4gZWNodGVzIFNlY3JldCE=";

// for this example the weather text for Berlin, district Neukölln, is requested
$gid = "10518";
$locationname = "Neukölln";

// $locationname = utf8_encode("Neukölln"); //Encoding in UTF8, if file is ISO-8859-1 encoded

// if one of the required parameters is missing: end
if (isset($basepath) === false || isset($package) === false || isset($uid) === false || isset($secret) === false || isset($gid) === false) {
    echo "at least one of the required parameters is missing";
    exit();
}

// date - required for checksum
$date = new DateTime("now", new DateTimeZone('UTC'));

// compile query-parameter
$data = array;
// current date, format: 2014-04-11-09
'date' => $date->format('Y-m-d-H'),
// required weather package
'package' => $package,
// UserID
'uid' => $uid,
// Geo-ID of the required location
'gid' => $gid,
// name of city district
'locationname' => $locationname,
);

// sort array keys alphabetically by parameter name:
ksort($data);

/**************************** calculate checksum ****************************
- put sorted parameter values together, separated by |
- attach secret, again separated by|
- hash string in md5_binary
- convert md5-hashed string base64 encoding
```
10 Example of generating a checksum

10.1 How to generate a checksum for a search request

.parameter:
User-ID: mail@example.com
Encode in Base64: mail@example.com -> bWFpbEBleGFtcGxlLmNvbQ==
.secret key: TmF0/HJsaWNoIGtlaW4gZWNodGVzIFNlY3JldCE=

uid=bWFpbEBleGFtcGxlLmNvbQ==
date=2014-05-23-08
v=1
name=Münster

1. Sort parameter names alphabetically & put parameter values together (separated by “|”)

date, name, uid, v
2014-05-23-08|Münster|bWFpbEBleGFtcGxlLmNvbQ==|1

2. Add secret key

2014-05-23-08|Münster|bWFpbEBleGFtcGxlLmNvbQ==|1|TmF0/HJsaWNoIGtlaW4gZWNodGVzIFNlY3JldCE=

3. Generate md5-hash, then encode into Base64 and URL
checksum= U8b/JLuojb82dWv8459YMW==
10.2 How to generate a checksum for a weather-API package request

parameter:
   User-ID: mail@example.com
   Base64-kodieren: mail@example.com -> bWFpbEBleGItcGxlLmNvbQ==
secret key: TmF0/HJsaWNoIGtlaW4gZWNodGVzIFNlY3JldCE=

uid=bWFpbEBleGItcGxlLmNvbQ==
date=2014-05-23-08
package=daily8
v=1
gid=10518

4. Sort parameter names alphabetically & put parameter values together (separated by "|")

   date, gid, package, uid, v
   2014-05-23-08|10518|daily8|bWFpbEBleGItcGxlLmNvbQ==|1

5. Add secret key

   2014-05-23-08|10518|daily8|bWFpbEBleGItcGxlLmNvbQ==|1|TmF0/HJsaWNoIGtlaW4gZWNodGVzIFNlY3JldCE=

6. Generate md5-hash, then encode into Base64 and URL

   checksum=lb9fpSERMkmdm0m46AR0aw==

https://api.wetteronline.de/weather?date=2014-05-23-08&gid=10518&package=daily8&uid=bWFpbEBleGItcGxlLmNvbQ%3D%3D&v=1&checksum=lb9fpSERMkmdm0m46AR0aw%3D%3D