기상청 생활기상지수 조회서비스



Open API 활용가이드

목 차

[1. 서비스 명세 3](#_Toc35612257)

[**1.1 생활기상지수 조회서비스** 3](#_Toc35612258)

[가. API 서비스 개요 3](#_Toc35612259)

[나. 상세기능 목록 4](#_Toc35612260)

[다. 상세기능내역 5](#_Toc35612261)

[1) [체감온도조회] 상세기능명세 5](#_Toc35612262)

[2) [불쾌지수조회] 상세기능명세 19](#_Toc35612263)

[3) [동파가능지수조회] 상세기능명세 3](#_Toc35612264)2

[4) [자외선지수조회] 상세기능명세 45](#_Toc35612265)

[5) [대기확산지수조회] 상세기능명세 5](#_Toc35612266)8

[6) [더위체감지수조회] 상세기능명세](#_Toc35612267) 70

[7) [열지수조회] 상세기능명세 8](#_Toc35612268)5

8)[체감온도(여름철)조회] 상세기능명세 94

**1. 서비스 명세**

**1.1 생활기상지수 조회서비스**

가. API 서비스 개요

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **API 서비스 정보** | **API명(영문)** | LivingWthrIdxService | | |
| **API명(국문)** | 생활기상지수 조회서비스 | | |
| **API 설명** | 체감온도, 불쾌지수, 동파가능지수, 자외선지수, 대기확산지수, 더위체감지수, 열지수 정보, 체감온도(여름철)을 조회하는 서비스 | | |
| **API 서비스**  **보안적용**  **기술 수준** | **서비스 인증/권한** | [O] ServiceKey [ ] 인증서 (GPKI/NPKI)  [ ] Basic (ID/PW) [ ] 없음 | | |
| **메시지 레벨**  **암호화** | [ ] 전자서명 [ ] 암호화 [O] 없음 | | |
| **전송 레벨 암호화** | [ ] SSL [O] 없음 | | |
| **인터페이스 표준** | [ ] SOAP 1.2  (RPC-Encoded, Document Literal, Document Literal Wrapped)  [O] REST (GET)  [ ] RSS 1.0 [ ] RSS 2.0 [ ] Atom 1.0 [ ] 기타 | | |
| **교환 데이터 표준**  **(중복선택가능)** | [O] XML [O] JSON [ ] MIME [ ] MTOM | | |
| **API 서비스**  **배포정보** | **서비스 URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService | | |
| **서비스 명세 URL**  **(WSDL 또는 WADL)** | N/A | | |
| **서비스 버전** | 1.0 | | |
| **서비스 시작일** | 2019-12-20 | **서비스 배포일** | 2019-12-20 |
| **서비스 이력** | 2019-12-20 : 서비스 시작  2020-06-26 : 체감온도(여름철) 추가 | | |
| **메시지 교환유형** | [O] Request-Response [ ] Publish-Subscribe  [ ] Fire-and-Forgot [ ] Notification | | |
| **데이터 갱신주기** | 수시 (일 2~8회) | | |

나. 상세기능 목록

|  |  |  |  |
| --- | --- | --- | --- |
| **번호** | **API명(국문)** | **상세기능명(영문)** | **상세기능명(국문)** |
| 1 | 생활기상지수 조회서비스 | getWindChillIdx | 체감온도조회 |
| 2 | getDiscomfortIdx | 불쾌지수조회 |
| 3 | getFreezeIdx | 동파가능지수조회 |
| 4 | getUVIdx | 자외선지수조회 |
| 5 | getAirDiffusionIdx | 대기확산지수조회 |
| 6 | getHeatFeelingIdx | 더위체감지수조회 |
| 7 | getHeatIdx | 열지수조회 |
| 8 | getSenTaIdx | 체감온도(여름철)조회 |

다. 상세기능내역

1) [체감온도조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 1 | **상세기능 유형** | 조회 (목록) |
| **상세기능명(국문)** | 체감온도조회 | | |
| **상세기능 설명** | 지점코드, 발표시간부터 3시간 단위로 66시간 후 까지의 예측값을 조회하는 기능 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx | | |
| **최대 메시지 사이즈** | [4000] byte | | |
| **평균 응답 시간** | [500] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2017030806 | 2017년 3월 8일 6시 발표 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A03 | 체감온도 코드 |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 지점정보 |
| date | 발표시간 | 10 | 1 | 2017030806 | ‘17년 3월 8일 6시발표 |
| h3 | 3시간 후 예측값 | 1 | 1 | -10 | 3시간 후 예측값(℃) |
| h6 | 6시간 후 예측값 | 1 | 1 | -9 | 6시간 후 예측값(℃) |
| h9 | 9시간 후 예측값 | 1 | 1 | -9 | 9시간 후 예측값(℃) |
| h12 | 12시간 후 예측값 | 1 | 1 | -8 | 12시간 후 예측값(℃) |
| h15 | 15시간 후 예측값 | 1 | 1 | -2 | 15시간 후 예측값(℃) |
| h18 | 18시간 후 예측값 | 1 | 1 | -6 | 18시간 후 예측값(℃) |
| h21 | 21시간 후 예측값 | 1 | 1 | -7 | 21시간 후 예측값(℃) |
| h24 | 24시간 후 예측값 | 1 | 1 | 1 | 24시간 후 예측값(℃) |
| h27 | 27시간 후 예측값 | 1 | 1 | 2 | 27시간 후 예측값(℃) |
| h30 | 30시간 후 예측값 | 1 | 1 | -3 | 30시간 후 예측값(℃) |
| h33 | 33시간 후 예측값 | 1 | 1 | -6 | 33시간 후 예측값(℃) |
| h36 | 36시간 후 예측값 | 1 | 1 | -5 | 36시간 후 예측값(℃) |
| h39 | 39시간 후 예측값 | 1 | 1 | -6 | 39시간 후 예측값(℃) |
| h42 | 42시간 후 예측값 | 1 | 1 | -7 | 42시간 후 예측값(℃) |
| h45 | 45시간 후 예측값 | 1 | 1 | -1 | 45시간 후 예측값(℃) |
| h48 | 48시간 후 예측값 | 1 | 1 | -3 | 48시간 후 예측값(℃) |
| h51 | 51시간 후 예측값 | 1 | 1 | -5 | 51시간 후 예측값(℃) |
| h54 | 54시간 후 예측값 | 1 | 1 | -7 | 54시간 후 예측값(℃) |
| h57 | 57시간 후 예측값 | 1 | 1 | -9 | 57시간 후 예측값(℃) |
| h60 | 60시간 후 예측값 | 1 | 1 | -7 | 60시간 후 예측값(℃) |
| h63 | 63시간 후 예측값 | 1 | 1 | -5 | 63시간 후 예측값(℃) |
| h66 | 66시간 후 예측값 | 1 | 1 | -3 | 66시간 후 예측값(℃) |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 11월 ~ 3월 (최근 발표자료로 제한)

# 체감온도 단계 및 범위

|  |  |
| --- | --- |
| **단계** | **기상조건** |
| 위험 | -15.4℃ 미만 |
| 경고 | -15.4 이상 -10.5℃ 미만 |
| 주의 | -10.5이상 -3.2℃ 미만 |
| 관심 | -3.2℃ 이상 |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx ?serviceKey=인증키&areaNo=1100000000&time=2017030806 |

|  |
| --- |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <code>A03</code>                  <areaNo>1100000000</areaNo>                  <date>2017030806</date>                  <h3>-10</h3>                  <h6>-9</h6>                  <h9>-9</h9>                  <h12>-8</h12>                  <h15>-2</h15>                  <h18>-6</h18>                  <h21>-7</h21>                  <h24>1</h24>                  <h27>2</h27>                  <h30>-3</h30>                  <h33>-6</h33>                  <h36>-5</h36>                  <h39>-6</h39>                  <h42>-7</h42>                  <h45>-1</h45>                  <h48>-3</h48>                  <h51>-5</h51>                  <h54>-7</h54>                  <h57>-9</h57>                  <h60>-7</h60>                  <h63>-5</h63>                  <h66>-3</h66>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer {  public static void main(String[] args) throws IOException  {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*지점코드\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2017030806", "UTF-8")); /\*2017년6월8일6시\*/  urlBuilder.append("&" + URLEncoder.encode("\_type","UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")); /\*xml, json 선택(미입력시 xml)\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300)  {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else  {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null)  {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx'; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*지점코드\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017030806'); /\*2017년6월8일6시\*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\*xml, json 선택(미입력시 xml)\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function () {  if (this.readyState == 4)  {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **C++ Code** |
| /\* C++ 샘플 코드 \*/  using System; using System.Net; using System.Net.Http; using System.IO;  namespace ConsoleApp1 {     class Program     {         static HttpClient client = new HttpClient();         static void Main(string[] args)         {             string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx"; // URL             url += "?ServiceKey=" + "서비스키"; // Service Key             url += "&ServiceKey=-";             url += "&pageNo=1";             url += "&numOfRows=10";             url += "&dataType=XML";             url += "&areaNo=1100000000";             url += "&time=2017030806";                          var request = (HttpWebRequest)WebRequest.Create(url);             request.Method = "GET";             string results = string.Empty;             HttpWebResponse response;             using (response = request.GetResponse() as HttpWebResponse)             {                 StreamReader reader = new StreamReader(response.GetResponseStream());                 results = reader.ReadToEnd();             }              Console.WriteLine(results);         }     } } |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx'; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('ServiceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*지점코드\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2017030806'); /\*2017년6월8일6시\*/  $queryParams .= '&' . urlencode('\_type') . '=' . urlencode('xml'); /\*xml, json 선택(미입력시 xml)\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx?serviceKey=인증키(URL Encode)&numOfRows=10&pageNo=1&areaNo=1100000000&time=2017030806&\_type=xml' |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx'  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('ServiceKey') => '-',CGI::escape('pageNo') => '1',CGI::escape('numOfRows') => '10',CGI::escape('dataType') => 'XML',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2017030806' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@" http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2017030806" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"\_type" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"xml" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  } |
| **Python Code** |
| /\* Python 샘플 코드 \*/  from urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx'  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('ServiceKey') : '-', quote\_plus('pageNo') : '1', quote\_plus('numOfRows') : '10', quote\_plus('dataType') : 'XML', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2017030806' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx';  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 지점코드 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017030806'); /\* 2017년6월8일6시 \*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\* xml, json 선택(미입력시 xml) \*/  request( {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>)  {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getWindChillIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")) /\* 2017년6월8일6시 \*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

2) [불쾌지수조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 2 | **상세기능 유형** | 조회 (목록) |
| **상세기능명(국문)** | 불쾌지수조회 | | |
| **상세기능 설명** | 지점코드, 발표시간부터 3시간 단위로 66시간 후 까지의 예측값을 조회하는 기능 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx | | |
| **최대 메시지 사이즈** | [4000] byte | | |
| **평균 응답 시간** | [500] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2017060806 | 2017년 6월 8일 6시 발표 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A06 | 불쾌지수 코드 |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 지점정보 |
| date | 발표시간 | 10 | 1 | 2017060806 | ‘17년 6월 8일 6시발표 |
| h3 | 3시간 후 예측값 | 1 | 1 | 78 | 3시간 후 예측값 |
| h6 | 6시간 후 예측값 | 1 | 1 | 75 | 6시간 후 예측값 |
| h9 | 9시간 후 예측값 | 1 | 1 | 71 | 9시간 후 예측값 |
| h12 | 12시간 후 예측값 | 1 | 1 | 72 | 12시간 후 예측값 |
| h15 | 15시간 후 예측값 | 1 | 1 | 73 | 15시간 후 예측값 |
| h18 | 18시간 후 예측값 | 1 | 1 | 74 | 18시간 후 예측값 |
| h21 | 21시간 후 예측값 | 1 | 1 | 75 | 21시간 후 예측값 |
| h24 | 24시간 후 예측값 | 1 | 1 | 76 | 24시간 후 예측값 |
| h27 | 27시간 후 예측값 | 1 | 1 | 77 | 27시간 후 예측값 |
| h30 | 30시간 후 예측값 | 1 | 1 | 80 | 30시간 후 예측값 |
| h33 | 33시간 후 예측값 | 1 | 1 | 82 | 33시간 후 예측값 |
| h36 | 36시간 후 예측값 | 1 | 1 | 73 | 36시간 후 예측값 |
| h39 | 39시간 후 예측값 | 1 | 1 | 74 | 39시간 후 예측값 |
| h42 | 42시간 후 예측값 | 1 | 1 | 85 | 42시간 후 예측값 |
| h45 | 45시간 후 예측값 | 1 | 1 | 86 | 45시간 후 예측값 |
| h48 | 48시간 후 예측값 | 1 | 1 | 77 | 48시간 후 예측값 |
| h51 | 51시간 후 예측값 | 1 | 1 | 75 | 51시간 후 예측값 |
| h54 | 54시간 후 예측값 | 1 | 1 | 73 | 54시간 후 예측값 |
| h57 | 57시간 후 예측값 | 1 | 1 | 71 | 57시간 후 예측값 |
| h60 | 60시간 후 예측값 | 1 | 1 | 69 | 60시간 후 예측값 |
| h63 | 63시간 후 예측값 | 1 | 1 | 68 | 63시간 후 예측값 |
| h66 | 66시간 후 예측값 | 1 | 1 | 67 | 66시간 후 예측값 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 6월 ~ 9월 (최근 발표자료로 제한)

# 불쾌지수 단계 및 범위

|  |  |  |
| --- | --- | --- |
| **단계** | **지수범위** | **설명 및 주의사항** |
| 매우높음 | 80 이상 | 전원 불쾌감을 느낌. |
| 높음 | 75이상 80 미만 | 50% 정도 불쾌감을 느낌. |
| 보통 | 68이상 75 미만 | 불쾌감을 나타내기 시작함. |
| 낮음 | 68 미만 | 전원 쾌적함을 느낌. |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx ?serviceKey=인증키&areaNo=1100000000&time=2017060806 |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <code>A06</code>                  <areaNo>1100000000</areaNo>                  <date>20170600806</date>                  <h3>78</h3>                  <h6>75</h6>                  <h9>71</h9>                  <h12>72</h12>                  <h15>73</h15>                  <h18>74</h18>                  <h21>75</h21>                  <h24>76</h24>                  <h27>77</h27>                  <h30>80</h30>                  <h33>82</h33>                  <h36>73</h36>                  <h39>74</h39>                  <h42>85</h42>                  <h45>86</h45>                  <h48>77</h48>                  <h51>75</h51>                  <h54>73</h54>                  <h57>71</h57>                  <h60>69</h60>                  <h63>68</h63>                  <h66>67</h66>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer  {  public static void main(String[] args) throws IOException  {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*서울지점\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")); /\*2017년6월8일6시\*/  urlBuilder.append("&" + URLEncoder.encode("\_type","UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")); /\*xml, json 선택(미 입력시 xml)\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300)  {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else  {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null) {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx'; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*서울지점\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\*2017년6월8일6시\*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\*xml, json 선택(미 입력시 xml)\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function ()  {  if (this.readyState == 4)  {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx'; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('ServiceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*서울지점\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2017060806'); /\*2017년6월8일6시\*/  $queryParams .= '&' . urlencode('\_type') . '=' . urlencode('xml'); /\*xml, json 선택(미 입력시 xml)\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **C++ Code** |
| / C++ 샘플 코드 \*/  using System;  using System.Net;  using System.Net.Http;  using System.IO;  namespace ConsoleApp1  {  class Program  {  static HttpClient client = new HttpClient();  static void Main(string[] args)  {  string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx"; // URL  url += "?ServiceKey=" + "서비스키"; // Service Key  url += "&ServiceKey=-";  url += "&areaNo=1100000000";  url += "&time=2017060806";  url += "&\_type=xml";  var request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "GET";  string results = string.Empty;  HttpWebResponse response;  using (response = request.GetResponse() as HttpWebResponse)  {  StreamReader reader = new StreamReader(response.GetResponseStream());  results = reader.ReadToEnd();  }  Console.WriteLine(results);  }  }  } |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx?serviceKey=인증키(URL Encode)&numOfRows=10&pageNo=1&areaNo=1100000000&time=2017060806&\_type=xml' |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@"http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2017060806" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"\_type" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"xml" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  } |
| **Python Code** |
| /\* Python 샘플 코드 \*/  rom urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx'  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('ServiceKey') : '-', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2017060806', quote\_plus('\_type') : 'xml' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx';  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 서울지점 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\* 2017년6월8일6시 \*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\* xml, json 선택(미 입력시 xml) \*/  request( {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx'  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('ServiceKey') => '-',CGI::escape('pageNo') => '1',CGI::escape('numOfRows') => '10',CGI::escape('dataType') => 'XML',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2017060806' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>)  {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getDiscomfortIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("1168053100", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2019111111", "UTF-8")) /\* 2017년6월8일6시 \*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

3) [동파가능지수조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 3 | **상세기능 유형** | 조회 (목록) |
| **상세기능명(국문)** | 동파가능지수조회 | | |
| **상세기능 설명** | 지점코드, 발표시간부터 3시간 단위로 66시간 후 까지의 예측값을 조회하는 기능 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx | | |
| **최대 메시지 사이즈** | [4000] byte | | |
| **평균 응답 시간** | [500] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2017030806 | 2017년 3월 8일 6시 발표 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A08 | 동파가능지수 코드 |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 지점정보 |
| date | 발표시간 | 10 | 1 | 2017030806 | ‘17년 3월 8일 6시발표 |
| h3 | 3시간 후 예측값 | 1 | 1 | 25 | 3시간 후 예측값 |
| h6 | 6시간 후 예측값 | 1 | 1 | 25 | 6시간 후 예측값 |
| h9 | 9시간 후 예측값 | 1 | 1 | 25 | 9시간 후 예측값 |
| h12 | 12시간 후 예측값 | 1 | 1 | 25 | 12시간 후 예측값 |
| h15 | 15시간 후 예측값 | 1 | 1 | 50 | 15시간 후 예측값 |
| h18 | 18시간 후 예측값 | 1 | 1 | 50 | 18시간 후 예측값 |
| h21 | 21시간 후 예측값 | 1 | 1 | 75 | 21시간 후 예측값 |
| h24 | 24시간 후 예측값 | 1 | 1 | 100 | 24시간 후 예측값 |
| h27 | 27시간 후 예측값 | 1 | 1 | 100 | 27시간 후 예측값 |
| h30 | 30시간 후 예측값 | 1 | 1 | 100 | 30시간 후 예측값 |
| h33 | 33시간 후 예측값 | 1 | 1 | 50 | 33시간 후 예측값 |
| h36 | 36시간 후 예측값 | 1 | 1 | 25 | 36시간 후 예측값 |
| h39 | 39시간 후 예측값 | 1 | 1 | 50 | 39시간 후 예측값 |
| h42 | 42시간 후 예측값 | 1 | 1 | 75 | 42시간 후 예측값 |
| h45 | 45시간 후 예측값 | 1 | 1 | 100 | 45시간 후 예측값 |
| h48 | 48시간 후 예측값 | 1 | 1 | 100 | 48시간 후 예측값 |
| h51 | 51시간 후 예측값 | 1 | 1 | 100 | 51시간 후 예측값 |
| h54 | 54시간 후 예측값 | 1 | 1 | 50 | 54시간 후 예측값 |
| h57 | 57시간 후 예측값 | 1 | 1 | 50 | 57시간 후 예측값 |
| h60 | 60시간 후 예측값 | 1 | 1 | 75 | 60시간 후 예측값 |
| h63 | 63시간 후 예측값 | 1 | 1 | 75 | 63시간 후 예측값 |
| h66 | 66시간 후 예측값 | 1 | 1 | 100 | 66시간 후 예측값 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 11월 ~ 3월 (최근 발표자료로 제한)

# 동파가능지수 단계 및 범위

|  |  |  |
| --- | --- | --- |
| **지수단계** | **단계기준** | **데이터 조건** |
| 매우높음 | 기온 -15℃ 미만 | 100 |
| 높음 | 기온 -15℃ 이상 -10℃ 미만 | 75 |
| 보통 | 기온 -10℃ 이상 -5℃ 미만 | 50 |
| 낮음 | 기온 -5℃ 이상 | 25 |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx ?serviceKey=인증키&areaNo=1100000000&time=2017030806 |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <code>A08</code>                  <areaNo>1100000000</areaNo>                  <date>2017030806</date>                  <h3>25</h3>                  <h6>25</h6>                  <h9>25</h9>                  <h12>25</h12>                  <h15>50</h15>                  <h18>50</h18>                  <h21>75</h21>                  <h24>100</h24>                  <h27>100</h27>                  <h30>100</h30>                  <h33>50</h33>                  <h36>25</h36>                  <h39>50</h39>                  <h42>75</h42>                  <h45>100</h45>                  <h48>100</h48>                  <h51>100</h51>                  <h54>50</h54>                  <h57>50</h57>                  <h60>75</h60>                  <h63>75</h63>                  <h66>100</h66>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer  {  public static void main(String[] args) throws IOException  {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("serviceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*서울지점\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2017030806", "UTF-8")); /\*2017년3월8일6시\*/  urlBuilder.append("&" + URLEncoder.encode("\_type","UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")); /\*xml, json 선택(미입력시 xml)\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300)  {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else  {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null)  {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx'; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('serviceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*서울지점\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017030806'); /\*2017년3월8일6시\*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\*xml, json 선택(미입력시 xml)\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function ()  {  if (this.readyState == 4)  {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx'; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('serviceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*서울지점\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2017030806'); /\*2017년3월8일6시\*/  $queryParams .= '&' . urlencode('\_type') . '=' . urlencode('xml'); /\*xml, json 선택(미입력시 xml)\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **C++ Code** |
| /\* C++ 샘플 코드 \*/  using System;  using System.Net;  using System.Net.Http;  using System.IO;  namespace ConsoleApp1  {  class Program  {  static HttpClient client = new HttpClient();  static void Main(string[] args)  {  string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx"; // URL  url += "?ServiceKey=" + "서비스키"; // Service Key  url += "&serviceKey=-";  url += "&areaNo=1100000000";  url += "&time=2017030806";  url += "&\_type=xml";  var request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "GET";  string results = string.Empty;  HttpWebResponse response;  using (response = request.GetResponse() as HttpWebResponse)  {  StreamReader reader = new StreamReader(response.GetResponseStream());  results = reader.ReadToEnd();  }  Console.WriteLine(results);  }  }  } |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx?serviceKey=인증키(URL Encode)&numOfRows=10&pageNo=1&areaNo=1100000000&time=2017030806&\_type=xml' |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@" http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"serviceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2017060806" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"\_type" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"xml" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  }3 |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx'  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('serviceKey') => '-',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2017030806',CGI::escape('\_type') => 'xml' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Python Code** |
| /\* Python 샘플 코드 \*/  from urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx'  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('serviceKey') : '-', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2017030806', quote\_plus('\_type') : 'xml' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx';  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('serviceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 서울지점 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017030806'); /\* 2017년3월8일6시 \*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\* xml, json 선택(미입력시 xml) \*/  request(  {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>)  {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getFreezeIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2017030806", "UTF-8")) /\* 2017년3월8일6시 \*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

4) [자외선지수조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 4 | **상세기능 유형** | 조회 (목록) |
| **상세기능명(국문)** | 자외선지수조회 | | |
| **상세기능 설명** | 지점코드, 발표시간에 오늘, 내일, 모레의 예측값을 조회하는 기능 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx | | |
| **최대 메시지 사이즈** | [1517] byte | | |
| **평균 응답 시간** | [100] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2017060806 | 2017년 6월 8일 6시 발표 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A07\_1 | 자외선지수 코드 |
| areaNo | 지점코드 | 12 | 1 | 1100000000 | 지점정보(‘서울’의미) |
| date | 발표시간 | 1 | 1 | 2017060806 | 2017년 6월 8일 6시 발표 |
| today | 오늘 예측값 | 1 | 1 | 6 | 오늘 예측값 |
| tomorrow | 내일 예측값 | 1 | 1 | 8 | 내일 예측값 |
| todayaftertomorrow | 모레 예측값 | 1 | 1 | 4 | 모레 예측값 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 연중 (최근 발표자료로 제한)

#자외선지수 단계 및 범위

|  |  |
| --- | --- |
| **단계** | **지수범위** |
| 위험 | 11 이상 |
| 매우높음 | 8~10 |
| 높음 | 6~7 |
| 보통 | 3~5 |
| 낮음 | 0~2 |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx ?serviceKey=인증키&areaNo=1100000000&time=2017060806 |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <code>A07\_1</code>                  <areaNo>1100000000</areaNo>                  <date>2017060806</date>                  <today>6</today>                  <tomorrow>8</tomorrow>                  <theDayAfterTomorrow>4</theDayAfterTomorrow>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer  {  public static void main(String[] args) throws IOException  {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("serviceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*서울지점\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")); /\*2017년6월8일6시\*/  urlBuilder.append("&" + URLEncoder.encode("\_type","UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")); /\*xml, json 선택(미입력시 xml)\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300)  {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else  {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null)  {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx’; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('serviceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*서울지점\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\*2017년6월8일6시\*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\*xml, json 선택(미입력시 xml)\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function ()  {  if (this.readyState == 4)  {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx'; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('serviceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*서울지점\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2017060806'); /\*2017년6월8일6시\*/  $queryParams .= '&' . urlencode('\_type') . '=' . urlencode('xml'); /\*xml, json 선택(미입력시 xml)\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **C++ Code** |
| /\* C++ 샘플 코드 \*/  using System;  using System.Net;  using System.Net.Http;  using System.IO;  namespace ConsoleApp1  {  class Program  {  static HttpClient client = new HttpClient();  static void Main(string[] args)  {  string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx"; // URL  url += "?ServiceKey=" + "서비스키"; // Service Key  url += "&serviceKey=-";  url += "&areaNo=1100000000";  url += "&time=2017060806";  url += "&\_type=xml";  var request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "GET";  string results = string.Empty;  HttpWebResponse response;  using (response = request.GetResponse() as HttpWebResponse)  {  StreamReader reader = new StreamReader(response.GetResponseStream());  results = reader.ReadToEnd();  }  Console.WriteLine(results);  }  }  } |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx?serviceKey=인증키(URL Encode)&numOfRows=10&pageNo=1&areaNo=1100000000&time=2017060806&\_type=xml' |

|  |
| --- |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@ "http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"serviceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2017060806" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"\_type" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"xml" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  } |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx'  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('serviceKey') => '-',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2017060806',CGI::escape('\_type') => 'xml' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Python Code** |
| /\* Python 샘플 코드 \*/  from urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx'  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('serviceKey') : '-', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2017060806', quote\_plus('\_type') : 'xml' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx';  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('serviceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 서울지점 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\* 2017년6월8일6시 \*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\* xml, json 선택(미입력시 xml) \*/  request(  {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>)  {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getUVIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")) /\* 2017년6월8일6시 \*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  Do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

5) [대기확산지수조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 5 | **상세기능 유형** | 조회 (목록) |
| **상세기능명(국문)** | 대기확산지수조회 | | |
| **상세기능 설명** | 지점코드, 발표시간부터 3시간 단위로 66시간 후 까지의 예측값을 조회하는 기능 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx | | |
| **최대 메시지 사이즈** | [4000] byte | | |
| **평균 응답 시간** | [500] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2017060806 | 2017년 6월 8일 6시 발표 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A09 | 대기확산지수 코드 |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 지점정보 |
| date | 발표시간 | 10 | 1 | 2017060806 | ‘17년 6월 8일 6시발표 |
| h3 | 3시간 후 예측값 | 1 | 1 | 25 | 3시간 후 예측값 |
| h6 | 6시간 후 예측값 | 1 | 1 | 25 | 6시간 후 예측값 |
| h9 | 9시간 후 예측값 | 1 | 1 | 25 | 9시간 후 예측값 |
| h12 | 12시간 후 예측값 | 1 | 1 | 25 | 12시간 후 예측값 |
| h15 | 15시간 후 예측값 | 1 | 1 | 50 | 15시간 후 예측값 |
| h18 | 18시간 후 예측값 | 1 | 1 | 50 | 18시간 후 예측값 |
| h21 | 21시간 후 예측값 | 1 | 1 | 75 | 21시간 후 예측값 |
| h24 | 24시간 후 예측값 | 1 | 1 | 100 | 24시간 후 예측값 |
| h27 | 27시간 후 예측값 | 1 | 1 | 100 | 27시간 후 예측값 |
| h30 | 30시간 후 예측값 | 1 | 1 | 100 | 30시간 후 예측값 |
| h33 | 33시간 후 예측값 | 1 | 1 | 50 | 33시간 후 예측값 |
| h36 | 36시간 후 예측값 | 1 | 1 | 25 | 36시간 후 예측값 |
| h39 | 39시간 후 예측값 | 1 | 1 | 50 | 39시간 후 예측값 |
| h42 | 42시간 후 예측값 | 1 | 1 | 75 | 42시간 후 예측값 |
| h45 | 45시간 후 예측값 | 1 | 1 | 100 | 45시간 후 예측값 |
| h48 | 48시간 후 예측값 | 1 | 1 | 100 | 48시간 후 예측값 |
| h51 | 51시간 후 예측값 | 1 | 1 | 100 | 51시간 후 예측값 |
| h54 | 54시간 후 예측값 | 1 | 1 | 50 | 54시간 후 예측값 |
| h57 | 57시간 후 예측값 | 1 | 1 | 50 | 57시간 후 예측값 |
| h60 | 60시간 후 예측값 | 1 | 1 | 75 | 60시간 후 예측값 |
| h63 | 63시간 후 예측값 | 1 | 1 | 75 | 63시간 후 예측값 |
| h66 | 66시간 후 예측값 | 1 | 1 | 100 | 66시간 후 예측값 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 연중 (최근 발표자료로 제한)

# 대기확산지수 단계 및 범위

|  |  |
| --- | --- |
| **지수단계** | **데이터 조건** |
| 낮음 | 100 |
| 보통 | 75 |
| 높음 | 50 |
| 매우높음 | 25 |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx ?serviceKey=인증키&areaNo=1100000000&time=2017061106 |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <code>A09</code>                  <areaNo>1100000000</areaNo>                  <date>2017060806</date>                  <h3>25</h3>                  <h6>25</h6>                  <h9>25</h9>                  <h12>25</h12>                  <h15>50</h15>                  <h18>50</h18>                  <h21>75</h21>                  <h24>100</h24>                  <h27>100</h27>                  <h30>100</h30>                  <h33>50</h33>                  <h36>25</h36>                  <h39>50</h39>                  <h42>75</h42>                  <h45>100</h45>                  <h48>100</h48>                  <h51>100</h51>                  <h54>50</h54>                  <h57>50</h57>                  <h60>75</h60>                  <h63>75</h63>                  <h66>100</h66>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer {  public static void main(String[] args) throws IOException {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("serviceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*서울지점\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")); /\*2017년6월8일6시\*/  urlBuilder.append("&" + URLEncoder.encode("\_type","UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")); /\*xml, json 선택(미입력시 xml)\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300) {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null) {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx’; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('serviceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*서울지점\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\*2017년6월8일6시\*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\*xml, json 선택(미입력시 xml)\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function () {  if (this.readyState == 4) {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx’; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('serviceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*서울지점\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2017060806'); /\*2017년6월8일6시\*/  $queryParams .= '&' . urlencode('\_type') . '=' . urlencode('xml'); /\*xml, json 선택(미입력시 xml)\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **C++ Code** |
| /\* C++ 샘플 코드 \*/  using System;  using System.Net;  using System.Net.Http;  using System.IO;  namespace ConsoleApp1  {  class Program  {  static HttpClient client = new HttpClient();  static void Main(string[] args)  {  string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx";  // URL  url += "?ServiceKey=" + "서비스키"; // Service Key  url += "&serviceKey=-";  url += "&areaNo=1100000000";  url += "&time=2017060806";  url += "&\_type=xml";  var request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "GET";  string results = string.Empty;  HttpWebResponse response;  using (response = request.GetResponse() as HttpWebResponse)  {  StreamReader reader = new StreamReader(response.GetResponseStream());  results = reader.ReadToEnd();  }  Console.WriteLine(results);  }  }  } |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx?serviceKey=인증키(URL Encode)&numOfRows=10&pageNo=1&areaNo=1100000000&time=2017061106&\_type=xml' |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@" http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"serviceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2017060806" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"\_type" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"xml" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  } |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx’  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('serviceKey') => '-',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2017060806',CGI::escape('\_type') => 'xml' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Python Code** |
| /\* Python 샘플 코드 \*/  From urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx’  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('serviceKey') : '-', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2017060806', quote\_plus('\_type') : 'xml' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url =’ http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx’;  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('serviceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 서울지점 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\* 2017년6월8일6시 \*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\* xml, json 선택(미입력시 xml) \*/  request( {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>) {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getAirDiffusionIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("110000000", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")) /\* 2017년6월8일6시 \*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

6) [더위체감지수조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 6 | **상세기능 유형** | 조회 (상세) |
| **상세기능명(국문)** | 더위체감지수조회 | | |
| **상세기능 설명** | 지점코드, 발표시간부터 3시간 단위로 66시간 후 까지의 예측값을 제공한다 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx | | |
| **최대 메시지 사이즈** | [4000] byte | | |
| **평균 응답 시간** | [500] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2017060806 | 2017년 6월 8일 6시 발표 |
| requestCode | 서비스대상 요청코드 | 10 | 1 | A20 | 요청코드(서비스대상)  A20 : 일반인  A21 : 노인  A22 : 어린이  A23 : 실외작업장  A24 : 농촌  A25 : 비닐하우스  A26 : 취약거주환경  \*하단의 서비스 대상정의 참조 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A20 | 입력한 더위체감지수 코드 |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 지점정보 |
| date | 발표시간 | 10 | 1 | 2017060806 | ‘17년 6월 8일 6시발표 |
| h3 | 3시간 후 예측값 | 1 | 1 | 28 | 3시간 후 예측값 |
| h6 | 6시간 후 예측값 | 1 | 1 | 30 | 6시간 후 예측값 |
| h9 | 9시간 후 예측값 | 1 | 1 | 32 | 9시간 후 예측값 |
| h12 | 12시간 후 예측값 | 1 | 1 | 33 | 12시간 후 예측값 |
| h15 | 15시간 후 예측값 | 1 | 1 | 30 | 15시간 후 예측값 |
| h18 | 18시간 후 예측값 | 1 | 1 | 27 | 18시간 후 예측값 |
| h21 | 21시간 후 예측값 | 1 | 1 | 27 | 21시간 후 예측값 |
| h24 | 24시간 후 예측값 | 1 | 1 | 21 | 24시간 후 예측값 |
| h27 | 27시간 후 예측값 | 1 | 1 | 18 | 27시간 후 예측값 |
| h30 | 30시간 후 예측값 | 1 | 1 | 25 | 30시간 후 예측값 |
| h33 | 33시간 후 예측값 | 1 | 1 | 29 | 33시간 후 예측값 |
| h36 | 36시간 후 예측값 | 1 | 1 | 32 | 36시간 후 예측값 |
| h39 | 39시간 후 예측값 | 1 | 1 | 30 | 39시간 후 예측값 |
| h42 | 42시간 후 예측값 | 1 | 1 | 27 | 42시간 후 예측값 |
| h45 | 45시간 후 예측값 | 1 | 1 | 25 | 45시간 후 예측값 |
| h48 | 48시간 후 예측값 | 1 | 1 | 26 | 48시간 후 예측값 |
| h51 | 51시간 후 예측값 | 1 | 1 | 27 | 51시간 후 예측값 |
| h54 | 54시간 후 예측값 | 1 | 1 | 32 | 54시간 후 예측값 |
| h57 | 57시간 후 예측값 | 1 | 1 | 30 | 57시간 후 예측값 |
| h60 | 60시간 후 예측값 | 1 | 1 | 27 | 60시간 후 예측값 |
| h63 | 63시간 후 예측값 | 1 | 1 | 20 | 63시간 후 예측값 |
| h66 | 66시간 후 예측값 | 1 | 1 | 18 | 66시간 후 예측값 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 5월 ~ 9월 (최근 발표자료로 제한)

# 더위체감지수 단계 및 범위

|  |  |
| --- | --- |
| 지수단계 | 데이터 조건 |
| 관심 | 21미만 |
| 주의 | 21이상 25미만 |
| 경고 | 25이상 28미만 |
| 위험 | 28이상 31미만 |
| 매우위험 | 31이상  \*노인, 농촌 비닐하우스, 실외작업장(도로, 건설현장, 조선소)은 30 이상 |

# 서비스 대상에 대한 정의

|  |  |
| --- | --- |
| **서비스대상** | **정의** |
| 일반인  (A20) | 기상관측소의 환경을 가정 |
| 노인  (A21) | 일반 환경에서 노년의 사람이 서 있는 상태를 가정 |
| 어린이  (A22) | 어린이의 야외활동이 많은 운동장(모래바닥)의 지상 1~1.5m 높이의 환경을 가정 |
| 실외작업장 (A23) | 햇볕이 내리쬐는 철판 바닥 위에서 작업하는 환경을 가정 |
| 농촌  (A24) | 농작업을 하는 야외 노지로 그늘이 없고 바람이 드는 환경을 가정 |
| 비닐하우스 (A25) | 농작물 수확을 위해 지어진 백색의 비닐하우스로 통풍 및 환기가 되지 않는 환경을 가정 |
| 취약거주환경 (A26) | 창문이 없거나 작아 환기가 안되고 바람이 거의 없는 쪽방, 강한 햇볕 아래 직접적 영향을 받는 옥탑방 환경을 가정 |
| - 상세 설명은 기상청 홈페이지 참조 (www.weather.go.kr > 기상+ > 생활기상정보 > 도움말) | |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx ?serviceKey=인증키&areaNo=1100000000&time=2017060806&requestCode=A20 |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <areaNo>1100000000</areaNo>                  <date>2017060806</date>                  <code>A20</code>                  <h3>28</h3>                  <h6>30</h6>                  <h9>32</h9>                  <h12>33</h12>                  <h15>30</h15>                  <h18>27</h18>                  <h21>27</h21>                  <h24>21</h24>                  <h27>18</h27>                  <h30>25</h30>                  <h33>29</h33>                  <h36>32</h36>                  <h39>30</h39>                  <h42>27</h42>                  <h45>25</h45>                  <h48>26</h48>                  <h51>27</h51>                  <h54>32</h54>                  <h57>30</h57>                  <h60>27</h60>                  <h63>20</h63>                  <h66>18</h66>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer  {  public static void main(String[] args) throws IOException  {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("pageNo","UTF-8") + "=" + URLEncoder.encode("1", "UTF-8")); /\*페이지번호\*/  urlBuilder.append("&" + URLEncoder.encode("numOfRows","UTF-8") + "=" + URLEncoder.encode("10", "UTF-8")); /\*한 페이지 결과 수\*/  urlBuilder.append("&" + URLEncoder.encode("dataType","UTF-8") + "=" + URLEncoder.encode("XML", "UTF-8")); /\*요청자료형식(XML/JSON)\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*지점코드\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")); /\*2017년 6월 8일 6시 발표\*/  urlBuilder.append("&" + URLEncoder.encode("requestCode","UTF-8") + "=" + URLEncoder.encode("A20", "UTF-8")); /\*서비스대상 요청코드\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300)  {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else  {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null)  {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx'; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); /\*페이지번호\*/  queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); /\*한 페이지 결과 수\*/  queryParams += '&' + encodeURIComponent('dataType') + '=' + encodeURIComponent('XML'); /\*요청자료형식(XML/JSON)\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*지점코드\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\*2017년 6월 8일 6시 발표\*/  queryParams += '&' + encodeURIComponent('requestCode') + '=' + encodeURIComponent('A20'); /\*서비스대상 요청코드\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function () {  if (this.readyState == 4) {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx'; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('ServiceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('pageNo') . '=' . urlencode('1'); /\*페이지번호\*/  $queryParams .= '&' . urlencode('numOfRows') . '=' . urlencode('10'); /\*한 페이지 결과 수\*/  $queryParams .= '&' . urlencode('dataType') . '=' . urlencode('XML'); /\*요청자료형식(XML/JSON)\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*지점코드\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2017060806'); /\*2017년 6월 8일 6시 발표\*/  $queryParams .= '&' . urlencode('requestCode') . '=' . urlencode('A20'); /\*서비스대상 요청코드\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **C++ Code** |
| /\* C++ 샘플 코드 \*/  using System;  using System.Net;  using System.Net.Http;  using System.IO;  namespace ConsoleApp1  {  class Program  {  static HttpClient client = new HttpClient();  static void Main(string[] args)  {  string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx"; // URL  url += "?ServiceKey=" + "서비스키"; // Service Key  url += "&ServiceKey=-";  url += "&pageNo=1";  url += "&numOfRows=10";  url += "&dataType=XML";  url += "&areaNo=1100000000";  url += "&time=2017060806";  url += "&requestCode=A20";    var request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "GET";    string results = string.Empty;  HttpWebResponse response;  using (response = request.GetResponse() as HttpWebResponse)  {  StreamReader reader = new StreamReader(response.GetResponseStream());  results = reader.ReadToEnd();  }  Console.WriteLine(results);  }  }  } |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx?ServiceKey=서비스키&ServiceKey=-&pageNo=1&numOfRows=10&dataType=XML&areaNo=1100000000&time=2017060806&requestCode=A20' |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@"http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"pageNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"numOfRows" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"10" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"dataType" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"XML" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2017060806" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"requestCode" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"A20" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  } |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx'  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('ServiceKey') => '-',CGI::escape('pageNo') => '1',CGI::escape('numOfRows') => '10',CGI::escape('dataType') => 'XML',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2017060806',CGI::escape('requestCode') => 'A20' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Python Code** |
| /\* Python 샘플 코드 \*/  from urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx'  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('ServiceKey') : '-', quote\_plus('pageNo') : '1', quote\_plus('numOfRows') : '10', quote\_plus('dataType') : 'XML', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2017060806', quote\_plus('requestCode') : 'A20' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx';  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); /\* 페이지번호 \*/  queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); /\* 한 페이지 결과 수 \*/  queryParams += '&' + encodeURIComponent('dataType') + '=' + encodeURIComponent('XML'); /\* 요청자료형식(XML/JSON) \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 지점코드 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\* 2017년 6월 8일 6시 발표 \*/  queryParams += '&' + encodeURIComponent('requestCode') + '=' + encodeURIComponent('A20'); /\* 서비스대상 요청코드 \*/  request(  {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>)  {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatFeelingIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("1168053100", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2019111111", "UTF-8")) /\* 2017년6월8일6시 \*/ urlBuilder.append("&" + URLEncoder.encode("requestCode","UTF-8") + "=" + URLEncoder.encode("A20", "UTF-8")); /\*서비스대상 요청코드\*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

7) [열지수조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 7 | **상세기능 유형** | 조회 (상세) |
| **상세기능명(국문)** | 열지수조회 | | |
| **상세기능 설명** | 지점코드, 발표시간부터 3시간 단위로 66시간 후 까지의 예측값을 제공한다 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx | | |
| **최대 메시지 사이즈** | [4000] byte | | |
| **평균 응답 시간** | [500] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2017060806 | 2017년 6월 8일 6시 발표 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A05 | 열지수 코드 |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 지점정보 |
| date | 발표시간 | 10 | 1 | 2017060806 | ‘17년 6월 8일 6시발표 |
| h3 | 3시간 후 예측값 | 1 | 1 | 35 | 3시간 후 예측값 |
| h6 | 6시간 후 예측값 | 1 | 1 | 42 | 6시간 후 예측값 |
| h9 | 9시간 후 예측값 | 1 | 1 | 55 | 9시간 후 예측값 |
| h12 | 12시간 후 예측값 | 1 | 1 | 46 | 12시간 후 예측값 |
| h15 | 15시간 후 예측값 | 1 | 1 | 33 | 15시간 후 예측값 |
| h18 | 18시간 후 예측값 | 1 | 1 | 27 | 18시간 후 예측값 |
| h21 | 21시간 후 예측값 | 1 | 1 | 24 | 21시간 후 예측값 |
| h24 | 24시간 후 예측값 | 1 | 1 | 19 | 24시간 후 예측값 |
| h27 | 27시간 후 예측값 | 1 | 1 | 29 | 27시간 후 예측값 |
| h30 | 30시간 후 예측값 | 1 | 1 | 38 | 30시간 후 예측값 |
| h33 | 33시간 후 예측값 | 1 | 1 | 50 | 33시간 후 예측값 |
| h36 | 36시간 후 예측값 | 1 | 1 | 45 | 36시간 후 예측값 |
| h39 | 39시간 후 예측값 | 1 | 1 | 36 | 39시간 후 예측값 |
| h42 | 42시간 후 예측값 | 1 | 1 | 32 | 42시간 후 예측값 |
| h45 | 45시간 후 예측값 | 1 | 1 | 28 | 45시간 후 예측값 |
| h48 | 48시간 후 예측값 | 1 | 1 | 32 | 48시간 후 예측값 |
| h51 | 51시간 후 예측값 | 1 | 1 | 40 | 51시간 후 예측값 |
| h54 | 54시간 후 예측값 | 1 | 1 | 56 | 54시간 후 예측값 |
| h57 | 57시간 후 예측값 | 1 | 1 | 62 | 57시간 후 예측값 |
| h60 | 60시간 후 예측값 | 1 | 1 | 49 | 60시간 후 예측값 |
| h63 | 63시간 후 예측값 | 1 | 1 | 38 | 63시간 후 예측값 |
| h66 | 66시간 후 예측값 | 1 | 1 | 30 | 66시간 후 예측값 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 6월 ~ 9월 (최근 발표자료로 제한)

# 열지수 단계 및 범위

|  |  |
| --- | --- |
| 지수단계 | 데이터 조건 |
| 위험 | 66 이상 |
| 매우높음 | 54 이상 66 미만 |
| 높음 | 41 이상 54 미만 |
| 보통 | 32 이상 41 미만 |
| 낮음 | 32 미만 |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx ?serviceKey=인증키&areaNo=1100000000&time=2017060806 |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <areaNo>1100000000</areaNo>                  <date>2017060806</date>                  <h3>30</h3>                  <h6>42</h6>                  <h9>55</h9>                  <h12>46</h12>                  <h15>33</h15>                  <h18>27</h18>                  <h21>24</h21>                  <h24>19</h24>                  <h27>29</h27>                  <h30>38</h30>                  <h33>50</h33>                  <h36>45</h36>                  <h39>36</h39>                  <h42>32</h42>                  <h45>28</h45>                  <h48>32</h48>                  <h51>40</h51>                  <h54>56</h54>                  <h57>62</h57>                  <h60>49</h60>                  <h63>38</h63>                  <h66>30</h66>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer  {  public static void main(String[] args) throws IOException  {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*서울지점\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2017060806", "UTF-8")); /\*2017년6월8일6시\*/  urlBuilder.append("&" + URLEncoder.encode("\_type","UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")); /\*xml, json 선택(미 입력시 xml)\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300)  {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else  {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null) {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx’; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*서울지점\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\*2017년6월8일6시\*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\*xml, json 선택(미 입력시 xml)\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function ()  {  if (this.readyState == 4)  {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx'; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('ServiceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*서울지점\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2017060806'); /\*2017년6월8일6시\*/  $queryParams .= '&' . urlencode('\_type') . '=' . urlencode('xml'); /\*xml, json 선택(미 입력시 xml)\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **C++ Code** |
| / C++ 샘플 코드 \*/  using System;  using System.Net;  using System.Net.Http;  using System.IO;  namespace ConsoleApp1  {  class Program  {  static HttpClient client = new HttpClient();  static void Main(string[] args)  {  string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx"; // URL  url += "?ServiceKey=" + "서비스키"; // Service Key  url += "&ServiceKey=-";  url += "&areaNo=1100000000";  url += "&time=2017060806";  url += "&\_type=xml";  var request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "GET";  string results = string.Empty;  HttpWebResponse response;  using (response = request.GetResponse() as HttpWebResponse)  {  StreamReader reader = new StreamReader(response.GetResponseStream());  results = reader.ReadToEnd();  }  Console.WriteLine(results);  }  }  } |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  ‘http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx?serviceKey=인증키(URL Encode)&numOfRows=10&pageNo=1&areaNo=1100000000&time=2017060806&\_type=xml' |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@"http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2017060806" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"\_type" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"xml" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  } |
| **Python Code** |
| /\* Python 샘플 코드 \*/  rom urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx'  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('ServiceKey') : '-', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2017060806', quote\_plus('\_type') : 'xml' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx';  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 서울지점 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2017060806'); /\* 2017년6월8일6시 \*/  queryParams += '&' + encodeURIComponent('\_type') + '=' + encodeURIComponent('xml'); /\* xml, json 선택(미 입력시 xml) \*/  request( {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx'  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('ServiceKey') => '-',CGI::escape('pageNo') => '1',CGI::escape('numOfRows') => '10',CGI::escape('dataType') => 'XML',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2017060806' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>)  {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getHeatIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("1168053100", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2019111111", "UTF-8")) /\* 2017년6월8일6시 \*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

8) [체감온도(여름철)조회] 상세기능명세

a) 상세기능정보

|  |  |  |  |
| --- | --- | --- | --- |
| **상세기능 번호** | 8 | **상세기능 유형** | 조회 (상세) |
| **상세기능명(국문)** | 체감온도(여름철)조회 | | |
| **상세기능 설명** | 지점코드, 발표시간부터 3시간 단위로 66시간 후 까지의 예측값을 제공한다 | | |
| **Call Back URL** | http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx | | |
| **최대 메시지 사이즈** | [4000] byte | | |
| **평균 응답 시간** | [500] ms | **초당 최대 트랙잭션** | [30] tps |

b) 요청 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| serviceKey | 인증키 | 100 | 1 | 인증키 | 공공데이터포털에서 발급받은 인증키 |
| numOfRows | 한 페이지 결과 수 | 4 | 0 | 10 | 한 페이지 결과 수  Default: 10 |
| pageNo | 페이지 번호 | 4 | 0 | 1 | 페이지 번호  Default: 1 |
| dataType | 응답자료형식 | 4 | 0 | XML | 요청자료형식(XML/JSON)  Default: XML |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 서울지점  공백일때: 전체지점조회 |
| time | 발표시각 | 10 | 1 | 2020062506 | 2020년 6월 25일 6시 발표 |
| requestCode | 서비스대상 요청코드 | 10 | 1 | A41 | 요청코드(서비스대상)  A41 : 노인  A42 : 어린이  A44 : 농촌  A45 : 비닐하우스  A46 : 취약거주환경  A47 : 도로  A48 : 건설현장  A49 : 조선소  \*하단의 서비스 대상정의 참조 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n)

c) 응답 메시지 명세

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **항목명(영문)** | **항목명(국문)** | **항목크기** | **항목구분** | **샘플데이터** | **항목설명** |
| numOfRows | 한 페이지 결과 수 | 4 | 1 | 10 | 한 페이지당 표출  데이터 수 |
| pageNo | 페이지 번호 | 4 | 1 | 1 | 페이지 번호 |
| totalCount | 데이터 총 개수 | 10 | 1 | 1 | 데이터 총 개수 |
| resultCode | 응답메시지 코드 | 2 | 1 | 00 | 응답 메시지코드 |
| resultMsg | 응답메시지 내용 | 100 | 1 | NORMAL SERVICE | 응답 메시지 설명 |
| dataType | 데이터 타입 | 4 | 1 | XML | 응답자료형식 (XML/JSON) |
| code | 지수코드 | 3 | 1 | A41 | 노인 지수 코드 |
| areaNo | 지점코드 | 10 | 1 | 1100000000 | 지점정보 |
| date | 발표시간 | 10 | 1 | 2020062506 | ‘20년 6월 25일 6시발표 |
| h3 | 3시간 후 예측값 | 1 | 1 | 28 | 3시간 후 예측값 |
| h6 | 6시간 후 예측값 | 1 | 1 | 30 | 6시간 후 예측값 |
| h9 | 9시간 후 예측값 | 1 | 1 | 32 | 9시간 후 예측값 |
| h12 | 12시간 후 예측값 | 1 | 1 | 33 | 12시간 후 예측값 |
| h15 | 15시간 후 예측값 | 1 | 1 | 30 | 15시간 후 예측값 |
| h18 | 18시간 후 예측값 | 1 | 1 | 27 | 18시간 후 예측값 |
| h21 | 21시간 후 예측값 | 1 | 1 | 27 | 21시간 후 예측값 |
| h24 | 24시간 후 예측값 | 1 | 1 | 21 | 24시간 후 예측값 |
| h27 | 27시간 후 예측값 | 1 | 1 | 18 | 27시간 후 예측값 |
| h30 | 30시간 후 예측값 | 1 | 1 | 25 | 30시간 후 예측값 |
| h33 | 33시간 후 예측값 | 1 | 1 | 29 | 33시간 후 예측값 |
| h36 | 36시간 후 예측값 | 1 | 1 | 32 | 36시간 후 예측값 |
| h39 | 39시간 후 예측값 | 1 | 1 | 30 | 39시간 후 예측값 |
| h42 | 42시간 후 예측값 | 1 | 1 | 27 | 42시간 후 예측값 |
| h45 | 45시간 후 예측값 | 1 | 1 | 25 | 45시간 후 예측값 |
| h48 | 48시간 후 예측값 | 1 | 1 | 26 | 48시간 후 예측값 |
| h51 | 51시간 후 예측값 | 1 | 1 | 27 | 51시간 후 예측값 |
| h54 | 54시간 후 예측값 | 1 | 1 | 32 | 54시간 후 예측값 |
| h57 | 57시간 후 예측값 | 1 | 1 | 30 | 57시간 후 예측값 |
| h60 | 60시간 후 예측값 | 1 | 1 | 27 | 60시간 후 예측값 |
| h63 | 63시간 후 예측값 | 1 | 1 | 20 | 63시간 후 예측값 |
| h66 | 66시간 후 예측값 | 1 | 1 | 18 | 66시간 후 예측값 |

※ 항목구분 : 필수(1), 옵션(0), 1건 이상 복수건(1..n), 0건 또는 복수건(0..n), 코드표별첨

# 자료제공 기간 : 5월 15일 ~ 10월 15일(여름철 방재기간, 변경될 수 있음)

# 체감온도(여름철) 단계 및 범위

|  |  |
| --- | --- |
| 지수단계 | 데이터 조건 |
| 노인 | - : 29미만  관심 : 29이상 31 미만  주의 : 31이상 34미만  경고 : 34이상 37미만  위험 : 37이상 |
| 어린이 |
| 취약거주환경 |
| 농촌 | - : 31미만  관심 : 31이상 33 미만  주의 : 33이상 35미만  경고 : 35이상 38미만  위험 : 38이상 |
| 비닐하우스 |
| 도로 |
| 건설현장 |
| 조선소 |

# 서비스 대상에 대한 정의

|  |  |
| --- | --- |
| **서비스대상** | **정의** |
| 노인  (A41) | 일반 환경에서 노년의 사람이 서 있는 상태를 가정 |
| 어린이  (A42) | 어린이의 야외활동이 많은 운동장(모래 바닥)의 지상 1~1.5m 높이의 환경을 가정 |
| 취약거주환경 (A46) | 창문이 없거나 작아 환기가 안되고 바람이 거의 없는 쪽방, 강한 햇볕 아래 직접적 영향을 받는 옥탑방 환경을 가정 |
| 농촌  (A44) | 농작업을 하는 야외 노지로 그늘이 없고 바람이 드는 환경을 가정 |
| 비닐하우스 (A45) | 농작물 수확을 위해 지어진 백색의 비닐하우스로 통풍 및 환기가 되지 않는 환경을 가정 |
| 도로(A47) | 햇볕이 내리쬐는 포장된 도로 위에서 작업하는 환경을 가정 |
| 건설현장(A48) | 작업자 주변에 철근이나 구조물 등이 설치된 건설현장에서 작업하는 환경을 가정 |
| 조선소(A49) | 햇볕이 내리쬐는 철판 바닥 위에서 작업하는 환경을 가정 |
| - 상세 설명은 기상청 홈페이지 참조 (www.weather.go.kr > 생활과산업 > 생활기상정보 > 도움말) | |

d) 요청/응답 메시지 예제

|  |
| --- |
| **요청메시지** |
| http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx ?serviceKey=인증키&areaNo=1100000000&time=2020062506&requestCode=A41 |
| **응답메시지** |
| <?xml version="1.0" encoding="UTF-8"?>  <response>      <header>          <resultCode>00</resultCode>          <resultMsg>NORMAL\_SERVICE</resultMsg>      </header>      <body>          <dataType>XML</dataType>          <items>              <item>                  <areaNo>1100000000</areaNo>                  <date>2020062506</date>                  <code>A41</code>                  <h3>25</h3>                  <h6>27</h6>                  <h9>26</h9>                  <h12>25</h12>                  <h15>25</h15>                  <h18>25</h18>                  <h21>24</h21>                  <h24>24</h24>                  <h27>26</h27>                  <h30>27</h30>                  <h33>28</h33>                  <h36>27</h36>                  <h39>25</h39>                  <h42>25</h42>                  <h45>24</h45>                  <h48>24</h48>                  <h51>26</h51>                  <h54>28</h54>                  <h57>30</h57>                  <h60>28</h60>                  <h63>27</h63>                  <h66>25</h66>              </item>          </items>          <numOfRows>10</numOfRows>          <pageNo>1</pageNo>          <totalCount>1</totalCount>      </body>  </response> |

e) 그 외 활용 예제코드

|  |
| --- |
| **Java Code** |
| /\* Java 샘플 코드 \*/  import java.io.InputStreamReader;  import java.net.HttpURLConnection;  import java.net.URL;  import java.net.URLEncoder;  import java.io.BufferedReader;  import java.io.IOException;  public class ApiExplorer  {  public static void main(String[] args) throws IOException  {  StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx"); /\*URL\*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey","UTF-8") + "=서비스키"); /\*Service Key\*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey","UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")); /\*공공데이터포털에서 받은 인증키\*/  urlBuilder.append("&" + URLEncoder.encode("pageNo","UTF-8") + "=" + URLEncoder.encode("1", "UTF-8")); /\*페이지번호\*/  urlBuilder.append("&" + URLEncoder.encode("numOfRows","UTF-8") + "=" + URLEncoder.encode("10", "UTF-8")); /\*한 페이지 결과 수\*/  urlBuilder.append("&" + URLEncoder.encode("dataType","UTF-8") + "=" + URLEncoder.encode("XML", "UTF-8")); /\*요청자료형식(XML/JSON)\*/  urlBuilder.append("&" + URLEncoder.encode("areaNo","UTF-8") + "=" + URLEncoder.encode("1100000000", "UTF-8")); /\*지점코드\*/  urlBuilder.append("&" + URLEncoder.encode("time","UTF-8") + "=" + URLEncoder.encode("2020062506", "UTF-8")); /\*2020년 6월 25일 6시 발표\*/  urlBuilder.append("&" + URLEncoder.encode("requestCode","UTF-8") + "=" + URLEncoder.encode("A41", "UTF-8")); /\*서비스대상 요청코드\*/  URL url = new URL(urlBuilder.toString());  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("GET");  conn.setRequestProperty("Content-type", "application/json");  System.out.println("Response code: " + conn.getResponseCode());  BufferedReader rd;  if(conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300)  {  rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));  } else  {  rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));  }  StringBuilder sb = new StringBuilder();  String line;  while ((line = rd.readLine()) != null)  {  sb.append(line);  }  rd.close();  conn.disconnect();  System.out.println(sb.toString());  }  } |
| **Javascript Code** |
| /\* Javascript 샘플 코드 \*/  var xhr = new XMLHttpRequest();  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx'; /\*URL\*/  var queryParams = '?' + encodeURIComponent('ServiceKey') + '='+'서비스키'; /\*Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\*공공데이터포털에서 받은 인증키\*/  queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); /\*페이지번호\*/  queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); /\*한 페이지 결과 수\*/  queryParams += '&' + encodeURIComponent('dataType') + '=' + encodeURIComponent('XML'); /\*요청자료형식(XML/JSON)\*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\*지점코드\*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2020062506'); /\*2020년 6월 25일 6시 발표\*/  queryParams += '&' + encodeURIComponent('requestCode') + '=' + encodeURIComponent('A41'); /\*서비스대상 요청코드\*/  xhr.open('GET', url + queryParams);  xhr.onreadystatechange = function () {  if (this.readyState == 4) {  alert('Status: '+this.status+' Headers: '+JSON.stringify(this.getAllResponseHeaders())+' Body: '+this.responseText);  }  };  xhr.send(''); |
| **PHP Code** |
| /\* PHP 샘플 코드 \*/  $ch = curl\_init();  $url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx'; /\*URL\*/  $queryParams = '?' . urlencode('ServiceKey') . '=서비스키'; /\*Service Key\*/  $queryParams .= '&' . urlencode('ServiceKey') . '=' . urlencode('-'); /\*공공데이터포털에서 받은 인증키\*/  $queryParams .= '&' . urlencode('pageNo') . '=' . urlencode('1'); /\*페이지번호\*/  $queryParams .= '&' . urlencode('numOfRows') . '=' . urlencode('10'); /\*한 페이지 결과 수\*/  $queryParams .= '&' . urlencode('dataType') . '=' . urlencode('XML'); /\*요청자료형식(XML/JSON)\*/  $queryParams .= '&' . urlencode('areaNo') . '=' . urlencode('1100000000'); /\*지점코드\*/  $queryParams .= '&' . urlencode('time') . '=' . urlencode('2020062506'); /\*2020년 6월 25일 6시 발표\*/  $queryParams .= '&' . urlencode('requestCode') . '=' . urlencode('A41'); /\*서비스대상 요청코드\*/  curl\_setopt($ch, CURLOPT\_URL, $url . $queryParams);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, 'GET');  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); |
| **C++ Code** |
| /\* C++ 샘플 코드 \*/  using System;  using System.Net;  using System.Net.Http;  using System.IO;  namespace ConsoleApp1  {  class Program  {  static HttpClient client = new HttpClient();  static void Main(string[] args)  {  string url = "http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx"; // URL  url += "?ServiceKey=" + "서비스키"; // Service Key  url += "&ServiceKey=-";  url += "&pageNo=1";  url += "&numOfRows=10";  url += "&dataType=XML";  url += "&areaNo=1100000000";  url += "&time=2020062506";  url += "&requestCode=A41";    var request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "GET";    string results = string.Empty;  HttpWebResponse response;  using (response = request.GetResponse() as HttpWebResponse)  {  StreamReader reader = new StreamReader(response.GetResponseStream());  results = reader.ReadToEnd();  }  Console.WriteLine(results);  }  }  } |
| **Curl Code** |
| /\* cURL 샘플 코드 \*/  curl --include --request GET \  'http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx?ServiceKey=서비스키&ServiceKey=-&pageNo=1&numOfRows=10&dataType=XML&areaNo=1100000000&time=2020062506&requestCode=A41' |
| **Object C Code** |
| /\* Objective-C 샘플 코드 \*/  #import <Foundation/Foundation.h>  #import <CoreFoundation/CoreFoundation.h>  @implementation NSString (URLEncoding)  //function used to encode query and matrix parameters  -(NSString \*)urlEncodeUsingEncoding:(NSStringEncoding)encoding  {  return (\_\_bridge NSString \*)CFURLCreateStringByAddingPercentEscapes(NULL,  (CFStringRef)self,  NULL,  (CFStringRef)@"!\*'\\"();:@&=+$,/?%#[]% ",  CFStringConvertNSStringEncodingToEncoding(encoding));  }  @end  int main (int argc, const char \* argv[])  {  @autoreleasepool  {  NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];  NSString \*endpoint = [NSString stringWithFormat:@"http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx"]; /\*URL\*/  NSString \*queryParams = [NSString stringWithFormat:@"?%@&%@&%@&%@&%@&%@&%@&%@",  [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"서비스키" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"ServiceKey" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"-" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"pageNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"numOfRows" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"10" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"dataType" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"XML" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"areaNo" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"1100000000" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"time" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"2020062506" urlEncodeUsingEncoding:NSUTF8StringEncoding]]  , [NSString stringWithFormat:@"%@=%@", [@"requestCode" urlEncodeUsingEncoding:NSUTF8StringEncoding], [@"A41" urlEncodeUsingEncoding:NSUTF8StringEncoding]]];  [request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@",endpoint,queryParams]]];  [request setHTTPMethod:@"GET"];  NSHTTPURLResponse \*urlResponse = nil;  NSError \*error = [[NSError alloc] init];  NSData \*responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&urlResponse error:&error];  NSString \*result = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];  NSLog(@"Response Code: %d", [urlResponse statusCode]);  if ([urlResponse statusCode] >= 200 && [urlResponse statusCode] < 300)  {  NSLog(@"Response: %@", result);  }  return 0;  }  } |
| **Ruby Code** |
| /\* Ruby 샘플 코드 \*/  require 'rubygems'  require 'rest\_client'  require 'cgi'  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx'  headers = { :params => { CGI::escape('ServiceKey') => '서비스키',CGI::escape('ServiceKey') => '-',CGI::escape('pageNo') => '1',CGI::escape('numOfRows') => '10',CGI::escape('dataType') => 'XML',CGI::escape('areaNo') => '1100000000',CGI::escape('time') => '2020062506',CGI::escape('requestCode') => 'A41' } }  response = RestClient::Request.execute :method => 'GET', :url => url , :headers => headers  puts response |
| **Python Code** |
| /\* Python 샘플 코드 \*/  from urllib2 import Request, urlopen  from urllib import urlencode, quote\_plus  url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx'  queryParams = '?' + urlencode({ quote\_plus('ServiceKey') : '서비스키', quote\_plus('ServiceKey') : '-', quote\_plus('pageNo') : '1', quote\_plus('numOfRows') : '10', quote\_plus('dataType') : 'XML', quote\_plus('areaNo') : '1100000000', quote\_plus('time') : '2020062506', quote\_plus('requestCode') : 'A41' })  request = Request(url + queryParams)  request.get\_method = lambda: 'GET'  response\_body = urlopen(request).read()  print response\_body |
| **Node.js Code** |
| /\* NodeJs 샘플 코드 \*/  var request = require('request');  var url = 'http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx';  var queryParams = '?' + encodeURIComponent('ServiceKey') + '=서비스키'; /\* Service Key\*/  queryParams += '&' + encodeURIComponent('ServiceKey') + '=' + encodeURIComponent('-'); /\* 공공데이터포털에서 받은 인증키 \*/  queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); /\* 페이지번호 \*/  queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); /\* 한 페이지 결과 수 \*/  queryParams += '&' + encodeURIComponent('dataType') + '=' + encodeURIComponent('XML'); /\* 요청자료형식(XML/JSON) \*/  queryParams += '&' + encodeURIComponent('areaNo') + '=' + encodeURIComponent('1100000000'); /\* 지점코드 \*/  queryParams += '&' + encodeURIComponent('time') + '=' + encodeURIComponent('2020062506'); /\* 2020년 6월 25일 6시 발표 \*/  queryParams += '&' + encodeURIComponent('requestCode') + '=' + encodeURIComponent('A41'); /\* 서비스대상 요청코드 \*/  request(  {  url: url + queryParams,  method: 'GET'  }, function (error, response, body)  {  //console.log('Status', response.statusCode);  //console.log('Headers', JSON.stringify(response.headers));  //console.log('Reponse received', body);  }); |
| **Kotlin Code** |
| /\* Kotlin 예제 \*/  import java.io.BufferedReader  import java.io.InputStreamReader  import java.net.HttpURLConnection  import java.net.URL  import java.net.URLEncoder  object OpenAPI  {  fun main(args: Array<String>)  {  val urlBuilder = StringBuilder("http://apis.data.go.kr/1360000/LivingWthrIdxService/getSenTaIdx") /\* URL \*/  urlBuilder.append("?" + URLEncoder.encode("ServiceKey", "UTF-8") + "=서비스키") /\* Service Key \*/  urlBuilder.append("&" + URLEncoder.encode("ServiceKey", "UTF-8") + "=" + URLEncoder.encode("-", "UTF-8")) /\* 공공데이터포털에서 받은 인증키 \*/  urlBuilder.append("&" + URLEncoder.encode("areaNo", "UTF-8") + "=" + URLEncoder.encode("1168053100", "UTF-8")) /\* 지점코드 \*/  urlBuilder.append("&" + URLEncoder.encode("time", "UTF-8") + "=" + URLEncoder.encode("2020062506", "UTF-8")) /\* 2020년6월25일6시 \*/ urlBuilder.append("&" + URLEncoder.encode("requestCode","UTF-8") + "=" + URLEncoder.encode("A41", "UTF-8")); /\*서비스대상 요청코드\*/  urlBuilder.append("&" + URLEncoder.encode("\_type", "UTF-8") + "=" + URLEncoder.encode("xml", "UTF-8")) /\* xml, json (미입력시 xml) \*/  val url = URL(urlBuilder.toString())  val conn = url.openConnection() as HttpURLConnection  conn.requestMethod = "GET"  conn.setRequestProperty("Content-type", "application/json")  System.out.println("Response code: " + conn.responseCode)  val br: BufferedReader  br = if (conn.responseCode >= 200 && conn.responseCode <= 300)  {  BufferedReader(InputStreamReader(conn.inputStream))  } else  {  BufferedReader(InputStreamReader(conn.errorStream))  }  val sb = StringBuilder()  var line: String?  do  {  line = br.readLine()  sb.append(line)  } while (line != null)  br.close()  conn.disconnect()  System.out.println(sb.toString())  }  } |

# 생활기상지수 생산주기, 생산시간, 예측기간, 기상청 홈페이지 서비스 기간

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **지 수 명** | **생산주기** | **생산시간** | **예측기간** | **제공기간** |
| 체감온도 | 일8회 | 03~24(3hr) | 오늘,내일,모레  (+3h-+66h) | 11월~3월 |
| 불쾌지수 | 일8회 | 03~24(3hr) | 오늘,내일,모레  (+3h-+66h) | 6월~9월 |
| 자외선지수 | 일2회 | 06, 18시 | 07시(오늘,내일,모레)  19시(내일,모레) | 연중 |
| 동파가능지수 | 일8회 | 03~24(3hr) | 오늘,내일,모레  (+3h-+66h) | 11월~3월 |
| 대기확산지수 | 일8회 | 03~24(3hr) | 오늘,내일,모레  (+3h-+66h) | 연중 |
| 더위체감지수 | 일2회 | 06, 18시 | 오늘,내일,모레  (+3h-+66h) | 5월~9월 |
| 열지수 | 일8회 | 03~24(3hr) | 오늘,내일,모레  (+3h-+66h) | 6월~9월 |
| 체감온도(여름철) | 일2회 | 06, 18시 | 오늘,내일,모레  (+3h-+66h) | 5월15일~  10월 15일  (변경될수있음) |

# 첨부. Open API 에러 코드 정리

|  |  |  |
| --- | --- | --- |
| **에러코드** | **에러메세지** | **설명** |
| 00 | NORMAL\_SERVICE | 정상 |
| 01 | APPLICATION\_ERROR | 어플리케이션 에러 |
| 02 | DB\_ERROR | 데이터베이스 에러 |
| 03 | NODATA\_ERROR | 데이터없음 에러 |
| 04 | HTTP\_ERROR | HTTP 에러 |
| 05 | SERVICETIME\_OUT | 서비스 연결실패 에러 |
| 10 | INVALID\_REQUEST\_PARAMETER\_ERROR | 잘못된 요청 파라메터 에러 |
| 11 | NO\_MANDATORY\_REQUEST\_PARAMETERS\_ERROR | 필수요청 파라메터가 없음 |
| 12 | NO\_OPENAPI\_SERVICE\_ERROR | 해당 오픈API서비스가 없거나 폐기됨 |
| 20 | SERVICE\_ACCESS\_DENIED\_ERROR | 서비스 접근거부 |
| 21 | TEMPORARILY\_DISABLE\_THE\_SERVICEKEY\_ERROR | 일시적으로 사용할 수 없는 서비스 키 |
| 22 | LIMITED\_NUMBER\_OF\_SERVICE\_REQUESTS\_EXCEEDS\_ERROR | 서비스 요청제한횟수 초과에러 |
| 30 | SERVICE\_KEY\_IS\_NOT\_REGISTERED\_ERROR | 등록되지 않은 서비스키 |
| 31 | DEADLINE\_HAS\_EXPIRED\_ERROR | 기한만료된 서비스키 |
| 32 | UNREGISTERED\_IP\_ERROR | 등록되지 않은 IP |
| 33 | UNSIGNED\_CALL\_ERROR | 서명되지 않은 호출 |
| 99 | UNKNOWN\_ERROR | 기타에러 |