Empath WebAPI Specifications

Mood Analysis API

Overview: API for computation and quantification of emotions by analyzing some audio data
Versions: v1, v2 (recommended version)

Starting April 1st 2017, HTTP requests won't be handled by our servers anymore.
Please change the HTTP requests to HTTPS before March 31st 2017.

Requests and responses

Requests

HTTP request: Depending on the API version, there are several endpoints.

API Version	HTTP request
v1	POST https://api.webempath.net/v1/analyzeWav
v2	POST https://api.webempath.net/v2/analyzeWav

Request header: The following information is expected in the header section.

Key	Type of information	Description
Content-type	multipart/form-data	Example: Content-Type: multipart/form-data; boundary=・・・・・・・

Request body: The following information is expected in the body section.

Key	Type of information	Description
apikey	API Key	API Key generated at the administration page
wav	Audio data	The audio data that can be analyzed by the API has to respect the following specifications: ・PCM WAVE format, 16bit ・Data size of 1.9MB or less ・PCM_FLOAT, PCM_SIGNED or PCM_UNSIGNED format ・The recording time is less than 5.0 seconds ・The sampling frequency is 11025 Hz ・Number of channels : 1 (monophonic sound)

Responses

HTTP status code: The following HTTP status is expected when the API runs normally

Code	Description
200 OK	When the audio data is successfully analyzed, or when the input data contains some errors

Returns: A JSON string is returned with either the quantified emotions in the case of a successful analysis, or error details otherwise

API Version	Examples in case of a successful analysis	Examples in case of an error
v1	{"error":0, "calm":21, "anger":0, "joy":35, "sorrow":24, "energy":20}	{"error": 1001}
v2	{"error":0, "calm":21, "anger":0, "joy":35, "sorrow":24, "energy":20}	{"error":1011, "msg": "wav format must be PCM_FLOAT, PCM_SIGNED, or PCM_UNSIGNED."}

Response items: Details of the composition of the JSON response

Key	Type of information	Description
error	Error code	If the data is successfully analyzed, this key will contain the value '0'. If an error occurs, it will contain the code of the corresponding error (refer to Error codes section).
msg (Since v2)	Error message	If the data is successfully analyzed, this key won't be part of the JSON response. In case of an error, the corresponding value will contain a message describing the error.
calm	Calm	Range: 0～50
anger	Anger	Range: 0～50
joy	Joy	Range: 0～50
sorrow	Sorrow	Range: 0～50
energy	Energy	Range: 0～50

Error codes

Error codes: The errors that may occur and be displayed are as followed. Depending on the version of the API, some error codes may differ.

Code in v1	Code in v2	Description
1001	1001	API Key is null.
	1002	The WAV file couldn't be read properly.
	1003	Content-type doesn't begin with multipart/form-data.
	1011	The sent WAV file doesn't correspond to PCM_FLOAT, PCM_SIGNED or PCM_UNSIGNED.
	1012	The sampling frequency in the input WAV file is different from 11025Hz.
	1013	The sound in the WAV file isn't a monophonic one.
	1014	The length of the WAV file exceeds 5.0 seconds.
	1015	The size of the WAV file exceeds 1.9MB.
	1016	The WAV file couldn't be properly read as a voice sound.
	1017	An invalid API version is being used.
1003	2001	The transmitted API Key exceeds the maximum number of API calls.
1002	2002	The account or the state is invalid.
	2003	The sent API key isn't available.
	2004	The sent API key can't be used or doesn't exist.
1004	3001	Access forbidden from the country.
9999	9999	Internal error.

Sample code(curl)

```
curl -X POST ¥
    -F apikey=YOUR APIKEY ¥
    -F wav=@/PATH/TO/WAVFILE.wav ¥
https://api.webempath.net/v2/analyzeWav
```

Sample code(Java)

The following sample code checks some operations using Java SE 8.
Also, in order to use this sample code, it is necessary to put those jar in the build path : httpmime-4.5.3.jar, httpclient-4.5.3.jar, httpcore-4.4.6.jar, commons-logging-1.2.jar, commons-codec-1.9.jar

If Maven can be used, those jars can be got by setting the pom.xml as described later.

WebEmpathV2Client.java

```
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.mime.HttpMultipartMode;
import org.apache.http.entity.mime.MultipartEntityBuilder;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

/**
 * This file demonstrates how to invoke WebEmpath's analyzeWav API version2 from Java
 */
public class WebEmpathV2Client {
    public static final String API_ENDPOINT = "https://api.webempath.net/v2/analyzeWav";
    public static void main(String[] arg) throws Exception {

        MultipartEntityBuilder builder = MultipartEntityBuilder.create()
                .setMode(HttpMultipartMode.STRICT)
                .setContentType(ContentType.MULTIPART_FORM_DATA)
                .addTextBody("apikey", "YOUR APIKEY")
                .addBinaryBody("wav", new java.io.File("/PATH/TO/WAVFILE.wav"));

        HttpPost httpPost = new HttpPost(API_ENDPOINT);
        httpPost.setEntity(builder.build());
        try (CloseableHttpClient client = HttpClients.createDefault();
                CloseableHttpResponse resp = client.execute(httpPost)) {

            if (resp.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
                System.out.println(EntityUtils.toString(resp.getEntity()));
            }
        }
    }
}
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>empath-api-sampleclient-J</groupId>
    <artifactId>empath-api-sampleclient-J</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>empath-api-sampleclient-J</name>
    <description>How to invoke WebEmpath's analyzeWav API version2 from Java.</description>
    <build>
        <sourceDirectory>src</sourceDirectory>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpmime</artifactId>
            <version>4.5.3</version>
        </dependency>
    </dependencies>
</project>
```

Sample code (Node.js)

The following code performs some operation checking using node v7.8.0 and npm 4.2.0.

package.json

```
{
  "name": "empath-demo",
  "version": "2.0.0",
  "description": "Example code of WebEmpath API. For node.js",
  "main": "index.js",
  "scripts": {
    "start": "npm install && node index.js"
  },
  "author": "smc_admin@webempath.net",
  "license": "MIT",
  "dependencies": {
    "request": "^2.62.0"
  }
}
```

index.js

```
/**
 * This file demonstrates how to invoke WebEmpath's analyzeWav API from node.js
 */
var fs = require('fs');
var request = require('request');
const API_ENDPOINT = 'https://api.webempath.net/v2/analyzeWav';
var formData = {
  apikey: "YOUR APIKEY",
  wav: fs.createReadStream("/PATH/TO/WAVFILE.wav")
};

request.post({ url: API_ENDPOINT, formData: formData }, function(err, response) {
  if (err) {
    console.trace(err);
  } else {
    var respBody = JSON.parse(response.body);
    console.log("result: " + JSON.stringify(respBody));
  }
});
```

Sample code (Python)

The following sample code performs some operation checking using Python 2.7.13, pip 9.0.1 and poster-0.8.1.

Installation of poster

```
# pip install poster==0.8.1

```

EmpathAPI_Sample.py

```
from poster.encode import multipart_encode, MultipartParam
from poster.streaminghttp import register_openers
import urllib2

url="https://api.webempath.net/v2/analyzeWav"
register_openers()
items = []
items.append(MultipartParam('apikey', "YOUR_APIKEY"))
items.append(MultipartParam.from_file('wav', "/PATH/TO/WAVFILE.wav"))
datagen, headers = multipart_encode(items)
request = urllib2.Request(url, datagen, headers)
response = urllib2.urlopen(request)
if response.getcode() == 200:
    print(response.read())
else:
    print("HTTP status %d" % (response.getcode()))
```

Sample code (PHP)

The following sample code performs some operation checking using PHP 5.4.

EmpathAPI_Sample.php

```
<?php
$url = 'https://api.webempath.net/v2/analyzeWav';
$apikey = 'YOUR APIKEY';
$wav = '@/PATH/TO/WAVFILE.wav;type=audio/wav';
$postfields = array(
 "apikey" => $apikey,
 "wav" => $wav
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $postfields);
$result = curl_exec($ch);
$result = json_decode($result);
print_r($result);
curl_close($ch);
```