



AudioTag

Application Programming Interface (API)

API revision: 2.6

Document revision: 1.53

Modification date: August 20, 2022

1. Table of Contents

1. Table of Contents	2
2. Introduction.....	3
3. Functionality and API calls	4
3.1. Overview.....	4
3.2. API keywords	4
3.3. API error codes	5
4. API queries and examples.....	7
4.1. Getting API information, testing connection.....	7
4.2. Getting account statistics	8
4.3. Recognition of an audio file by file upload	10
4.3.1. Audio recognition request.....	10
4.3.2. Getting audio recognition result	13
4.3.3. A complete example of audio file recognition by upload	17
4.4. Recognition of a remote audio file via direct URL.....	20
5. Contacts.....	28

2. Introduction

AudioTag API is designed to allow full automation of music identification queries to the AudioTag recognition engine. The API allows integrating AudioTag music recognition functionality with 3rd-party systems, services and software applications.

The API functionality can be used only by registered AudioTag users. To use the API, you need to register your account at <https://audiotag.info/apisection>.

Your calls to the API must be directed to <https://audiotag.info/api>. Each API call requires authentication. The authentication is done using a special “API access key”—a sequence of symbols unique to every AudioTag user. Users can obtain the access keys in the control panel of their AudioTag account, under API menu (<https://user.audiotag.info> -> “API keys” tab)

The use of AudioTag API is regulated by AudioTag “Terms of Use” (https://user.audiotag.info/terms_of_use). Each user utilizing AudioTag API must make sure that his use of AudioTag API functionality fully complies with the AudioTag terms of use.

User’s account is charged for certain API operations according to the standard AudioTag pricing (<https://user.audiotag.info> -> “Balance” tab).

3. Functionality and API calls

3.1. Overview

AudioTag API is available at <https://audiotag.info/api>. Users must be registered in order to use the API.

All API queries must be performed via HTTP(S) protocol using HTTP POST method.

Each query must include the user's unique API access key. The key must be valid and active. The user can manage his access API keys in the control panel at his AudioTag account, under API menu (<https://user.audiotag.info> -> "API keys" tab). Keys can be created, deleted, activated and deactivated. API queries can be done only using valid and active keys.

API response represents textual data in a form of JSON string. Example:

```
{"success":true,"status":"normal","api_ver":"2.6","action":"info"}
```

3.2. API keywords

AudioTag API operates with query keywords listed in Table 1.

Each API query must include the '**apikey**' keyword with corresponding API access key data. The supplied key must be valid and active. You can manage (generate, delete, activate, de-activate) the API keys at your AudioTag account, under API menu (<https://user.audiotag.info> -> "API keys" tab).

Each API query must include the '**action**' keyword with corresponding data specifying the requested action.

Table 1. API query keywords

API keyword	Type	Description
apikey	string	API access key. The key must be valid and active. Each API query must include the ' apikey ' keyword with corresponding data.
action	string	Specifies requested API query action. Each API query must include the ' action ' keyword with corresponding data. The following actions can be requested: - 'stat' – request for statistics on the API key usage. Refer to the Table 4. - 'info' – request for API status - 'identify' – request to perform acoustic recognition of an audio file by upload - 'get_result' – get recognition results of a recognition request made using 'identify' - 'identify_offline_stream' – request to perform acoustic recognition of a remote audio file/ offline (pre-recorded) audio stream by its URL - 'get_result_offline_stream' – get recognition results of a recognition request made using 'identify_offline_stream'

url	string	Direct link on a remote audio file. The URL must be an absolute resource URI. This keyword must be used only as part of the 'action:identify_offline_stream' query. The following file formats are accepted: WMA, WAV, OPUS, OGG, MP3, MP2, M4A, FLAC, AMR, ALAC, AIFF, AIF, AAC, 3GPP, 3GP. Maximal allowed file size is 5000 Mb.
file	binary	User's audio file intended for recognition. This keyword must be used only as part of the 'action:identify' query. The file transfer must be performed as Content-Type "multipart/form-data" . The following file formats are accepted: WMA, WAV, OPUS, OGG, MP3, MP2, M4A, FLAC, AMR, ALAC, AIFF, AIF, AAC, 3GPP, 3GP. Maximal file size is 100 Mb.
base_time	String	Absolute time (in format "Hours:Minutes:Seconds", e.g. 23:12:32) of the start of an offline audio stream recording. Recognition results for the offline audio stream/file will be reported relative to this time. Default: "00:00:00". This keyword can be used only as part of the 'action:identify_offline_stream' query.
overwrite	number	Can be used with 'action:identify_offline_stream' only. When overwrite=1 is sent, it forces the system to "forget" any existing recognition results for the specified URL and re-download/re-process the URL from scratch.
start_time	number	Time-point (in seconds) from the beginning of the file to start the identification. Default: 0 seconds (from the very beginning of the file). This keyword can be used only as part of the 'action:identify' query.
time_len	number	Duration to identify (in seconds), starting from the 'start_time' . Default: till the end of file. This keyword can be used only as part of the 'action:identify' query.
token	string	Unique recognition query token (ID) returned by the server upon acceptance of the 'identify' request. This keyword must be used only as part of the 'action:get_result' or 'action:identify_offline_stream' queries.

3.3. API error codes

AudioTag API returns errors via a dedicated 'error' field. The full list of errors is listed in the.

Table 2. API error list

Query / action	Error description
all types of queries	'invalid command or parameter' – wrong or unrecognized API keyword or query type 'api key expired' – API access key has expired 'api key inactive' – API access key is inactive; only active API keys can be used to make queries. 'api key unrecognized' – wrong/non-existing API access key 'api key invalid or empty' – API access key was not specified or wrong 'service temporarily not available' – the API service is not available at the moment 'internal error: no server response' 'internal error: server init error'
action:info	'internal error'
action:stat	'api key stats not found'

action:identify	'credit balance exhausted' – not enough credit point on the user's balance to complete the action 'no audio found, format invalid or unsupported' – the file does not include audio track or file format could not be converted/recognized 'audio duration is too short' – cannot process the file due to its short playback duration 'credit balance insufficient or exhausted' – not enough credit point on the user's balance to complete the action 'internal error: could not communicate to the recognition server' 'internal error: could not process the file'
action:get_result	'no recognition result found for the token' – provided identification query token is not found or expired; this error is return in one of the 3 cases: (i) music recognition for the token did not yield any results (ii) the token is wrong (ii) the token has expired 'internal error: server recognition error' 'internal error: server is not ready'

4. API queries and examples

This section provides comprehensive details on AudioTag API and a list of ready-to-use examples on PHP and Python.

Please note that the examples shown below use exemplary API access key). You must replace the access key with your own, created under your account.

4.1. Getting API information, testing connection

API information can be requested using `'action:info'`.

A comprehensive list of keywords returned by the `'action:info'` query is provided in the Table 3.

Table 3. `'action:info'` reply keywords

Keyword	Type	Description
success	boolean	Query result: true or false
status	string	'normal'
api_ver	string	Current version of the AudioTag API
action	string	'info'

Example 1. PHP example code of `'action:info'`:

```
<?php
$url = 'https://audiotag.info/api';

$apikey = '1234567abcdef1234567890abcdef123';# generate and place here your unique API access key, the key must
be 'active'
$action = 'info';

$ch = curl_init();

$args=[
    'apikey' => $apikey,
    'action' => $action
];

curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
$Result = curl_exec ($ch);
curl_close ($ch);
var_export($Result);

?>
```

Example 2. Python example code of `'action:info'`(Python 2.7):

```
import requests
import json

apikey = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key
must be 'active'

payload = {'action': 'info', 'apikey': apikey}

result = requests.post('https://audiotag.info/api', data=payload)

print(result.text)
json_object = json.loads(result.text);
pretty_print = json.dumps(json_object, indent=4, sort_keys=True)
print(pretty_print);
```

Example server replies:

Successful:

```
{"success":true,"status":"normal","api_ver":"2.6","action":"info"}
```

Erroneous:

```
{"success":false,"error":"api key unrecognized"}
```

4.2. Getting account statistics

Account statistics can be requested using **'action:stat'**.

A comprehensive list of keywords returned by the **'action:stat'** query is provided in the Table 4.

Table 4. **'action:stat'** reply keywords

Keyword	Type	Description
expiration_date	string	Expiration date and time of the API key
queries_count	number	Total count of API queries made using this key
uploaded_duration_sec	number	Total audio duration of audio files passed to identification using this key
uploaded_size_bytes	number	Total size (in bytes) of files passed to identification using this key
credits_spent	number	Total amount of credit points spent using this key
current_credit_balance	number	Current credit balance on account owning this key
identification_free_sec_remainder	number	Remaining budget of free seconds to analyze

Example 3. PHP example code of **'action:stat'** (PHP v5.5 and up):

```
<?php
$url = 'https://audiotag.info/api';
```

```
$apikey = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key must be 'active'
$action = 'stat';

$ch = curl_init();

$args=[
    'apikey' => $apikey,
    'action' => $action
];

curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
$result = curl_exec ($ch);
curl_close ($ch);
print_r($result);

?>
```

Example 4. Python example code of 'action:stat' (Python 2.7):

```
import requests
import json

apikey = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key must be 'active'

payload = {'action': 'stat', 'apikey': apikey}

r = requests.post('https://audiotag.info/api',data=payload)

print(r.text)
j = json.loads(r.text);
print(j)
```

Example server replies:

Successful:

```
{"current_credit_balance":0,"identification_free_sec_remainder":1704,"expiration_date":"2022-05-15 19:06:12","queries_count":23,"uploaded_duration_sec":96,"uploaded_size_bytes":1058656,"credits_spent":0,"success":true}
```

Erroneous:

```
{"success":false,"error":"api key unrecognized"}
```

4.3. Recognition of an audio file by file upload

4.3.1. Audio recognition request

Audio identification of a single file (e.g. music track, short audio recording, etc.) by file upload is performed using **'action:identify'**. This method is mostly designed for recognition of unknown music tracks. If you want recognize long recorded audio stream and get full printing of its contents, use another method as described in 4.4.

The following API keywords are used to form the identification query: **'apikey'**, **'action'**, **'file'**, **'start_time'**, **'time_len'**. Refer to the list of API keywords in the Table 1.

The file transfer must be performed as **Content-Type "multipart/form-data"**.

Native audio format used by the AudioTag engine is RIFF wave (.wav), PCM, 16 bit, 8 KHz, mono. The following file formats are also accepted and are converted internally: WMA, WAV, OPUS, OGG, MP3, MP2, M4A, FLAC, AMR, ALAC, AIFF, AIFAAC, 3GPP, 3GP. Please note that the file you upload must have one of the extensions from the list.

It is highly recommended to upload only the AudioTag's native wave-files (PCM, 16 bit, 8 KHz, mono). It puts less load on the AudioTag server and leads to faster API reaction on your recognition requests. You can use FFmpeg tool (open-source) to convert other audio file formats into AudioTag's native format by executing the following command:

```
ffmpeg -i input_file.ext -ar 8000 -ac 1 -vn converted.wav
```

If you want to recognize an unknown music track please keep in mind that sending the entire track for recognition is suboptimal. Optimal duration for uploaded audio is 15-30 seconds – it is usually enough for the recognition engine. Recognizing music tracks by their short fragments represents the most optimal approach as it saves credit points on your account and does not occupy the recognition server.

Minimal file duration accepted by the API is 5 seconds.

Maximal file size is 100 Mb.

The communication with the AudioTag server is done asynchronously in the sense that the client should not stand by waiting for the identification result in response to the **'action:identify'** query. Instead of the identification result, in response to the query the client immediately receives a unique token(identifier) of the corresponding server job it created and can finish the connection with the server. The token should be used with another query, namely, **'action:get_result'** accompanied with the token (**'token:token'**) in order to check the identification process (job) status and get identification data in case the audio recognition result is ready.

Upon requesting **'action:get_result'**, the client receives a **'result'** which can take the following values:

- **'wait'** – recognition is still in progress
- **'found'** – recognition finished, identification data is provided with data under **'data'** keyword
- **'not found'** – recognition finished, the audio could not be recognized

The token and the recognition data associated with it are valid for 6hours from the job creation time.

A comprehensive list of keywords returned by the **'action:identify'** query is provided in the Table 5.

Table 5. 'action:identify' reply keywords

Keyword	Type	Description
success	boolean	Query result: true or false
token	string	Unique token (identifier) of the identification request to be used with the 'get_result' query
error	string	Error description, refer to the Table 2. In case of error, the 'job_status' keyword is be absent, and the 'success' value is 'false'.
job_status	string	'wait'
start_time	number	Effective time-point (in seconds) from the beginning of the file where the identification starts.
time_len	number	Effective duration that is going to be analyzed by the server (in seconds)

Example 5. PHP example code of 'action:identify' (PHP v5.5 and up):

```
<?php
$url = 'https://audiotag.info/api';

$apikey = '1234567abcdef1234567890abcdef123';# generate and place here your unique API access key, the key must
be 'active'
$action = 'identify';
$filename = 'gazebo_12s.wav';

$args=[
    'apikey' => $apikey,
    'action' => $action,
    'file'   => curl_file_create($filename, 'application/octet-stream', $filename),
];

$ch = curl_init();
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
$Result = curl_exec ($ch);
curl_close ($ch);
var_export($Result);

?>
```

Example 6. Python example code of 'action:identify' (Python 2.7):

```
import requests
import json

filename = 'gazebo_12s.wav'
api_url = 'https://audiotag.info/api'
apikey = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key
must be 'active'

payload = {'action': 'identify', 'apikey': apikey}

result = requests.post(api_url,data=payload,files={'file': open(filename, 'rb')})
```

```
print(result.text)
json_object = json.loads(result.text);
pretty_print = json.dumps(json_object, indent=4, sort_keys=True)
print(pretty_print);
```

Example server replies:

Successful:

```
{"success":true,"error":null,"token":"c5c7f4aa2affc79d39439f2fbe5bebe6","start_time":0,"time_len":12,"job_status":"wait"}
```

Erroneous:

```
{"success":false,"error":"audio duration is too short"}
```

4.3.2. Getting audio recognition result

In order to receive audio recognition result for a file delivered by upload, the client has to perform **'action:get_result'** query using the token obtained from the **'action:identify'** query.

A comprehensive list of keywords returned by the **'action:get_result'** query is provided in the Table 6.

Table 6. **'action:get_result'** reply keywords

Keyword	Type	Description
success	boolean	Query result: true or false
error	string	Error description, refer to the Table 2. In case of error, the 'job_status' keyword is be absent, and the 'success' value is 'false' .
result	string	'wait' – recognition is still in progress 'found' – recognition is finished, identification data is provided with data under 'data' keyword 'not found' – recognition is finished, the audio could not be recognized
data	array	Array of objects. The object structure has the following elements: 'confidence' – confidence of the candidate in % 'time' – time-interval relative to the beginning of the analyzed audio file where the candidate is recognized 'tracks' – array of objects in which each object contains records in the following order: 'track name' 'artist name' 'album name' 'album year' Refer to the below example server reply showing successful recognition result.

Example 7. PHP example code of **'action:get_result'** (PHP v5.5 and up):

```
<?php
$url = 'https://audiotag.info/api';

$apikey = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key must
be 'active'
$action = 'get_result';
$token = '5a3cd8d32a8a616e5cd28703012ed1de'; # query token received in reponse to the action:identify request

$ch = curl_init();

$args=[];
$args['apikey'] = $apikey;
$args['action'] = $action;
$args['token'] = $token;

$n=1;
while($n < 100){ # query the server periodically, not more than 100 times
```

```
$n++;
usleep(500000); # delay 0.5sec between calls (not required but recognition takes some time to the server
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
$Result=curl_exec ($ch);
if(!$Result){
    echo "error accessing the server\n";
    exit();
}
$Result = json_decode($Result,true);
if(!empty($Result['success']) && $Result['result']!='wait'){
    break;
    # the server has returned result $Result['data'] - see description of the "data" object
}
if(empty($Result['success'])){
    break;
    # error occurred
}
# otherwise -- query the server again
# example of server response in case the recognition result is not yet ready and still awaits processing
in the server's queue
# {
# "success":true,
# "error":null,
# "result": "wait",
# }
}
curl_close ($ch);

if(empty($Result['success']) || empty($Result['data'])){
    echo "Server could not recognize the file\n";
    echo "Error: ".$Result['error']."\n";
}
var_export($Result);

?>
```

Example 8. Python example code of 'action:get_result' (Python 2.7):

```
import requests
import json
import time

apikey = '1234567890abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key must be 'active'
token = '9bd30fc62ab4cbc9016070e9904b25ef'; # query token received in reponse to the action:identify request
url = 'https://audiotag.info/api';

n=1;
while n < 100 :
    time.sleep(0.5) ;
    print('request:%d'%(n));
    n+=1;
    payload = {'action': 'get_result', 'token':token, 'apikey': apikey}
    result = requests.post(url,data=payload)
    print(result.text);
    json_object = json.loads(result.text);
    print(json_object);
    if not json_object.has_key('success') or json_object['success']==False or json_object['result']!='wait' :
        break;

pretty_print = json.dumps(json_object, indent=4, sort_keys=True)
print(pretty_print);
```

Example server replies:

Finished / Successful:

```
{
  "success":true,
  "error":null,
  "result":"found",
  "data": [
    {
      "tracks": [
        [
          "Hello Mr. Monkey",
          "Arabesque",
          "Grand Collection",
          2001
        ],
        [
          "Arabesque - Hello Mr. Monkey",
          "Various - Disco, New Wave",
          "All Stars Disco (3 & 4)",
          2001
        ]
      ],
      "confidence": 321,
      "time": "100 - 200"
    },
    {
      "tracks": [
        [
          "Hello Mr. Monkey",
          "Arabesque",
          "Arabesque-II (City Cats)",
          1979
        ]
      ],
      "confidence": 285,
      "time": "100 - 170"
    }
  ]
}
```

Finished / could not recognize:

```
{
  "success":true,
  "error":null,
  "result":"not found",
}
```

Erroneous (result is not ready yet):

```
{
  "success":true,
  "error":null,
  "result": "wait",
}
```

Erroneous:

```
(  
  "success":true  
  "error":null  
  "result":"no recognition result found for the token"  
)
```

4.3.3. A complete example of audio file recognition by upload

A complete example of audio recognition process for a file delivered by upload using **'action:identify'** and **'action:get_result'** is shown below.

Example 9. PHP example code of the complete audio recognition cycle (PHP v5.5 and up):

```
<?php

$url = 'https://audiotag.info/api';

$filename = 'gazebo_12s.wav';
$apikey = '1234567abcdef1234567890abcdef123';# generate and place here your unique API access key, the key
must be 'active'
$action = 'identify';

$ch = curl_init();

$args=[];
$args['apikey'] = $apikey;
$args['action'] = $action;
$args['start_time'] = 20; # start recognition with offset of 20 sec from the beginning of the file
$args['time_len'] = 50; # analyze only 50 seconds from the start point

$args['file'] = curl_file_create($filename, 'application/octet-stream', $filename);

curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
$Result=curl_exec ($ch);
curl_close ($ch);
# if curl_exec returns false - request to the server failed
if($Result===false){
    echo 'Server does not responding';
    echo ' or file not found';
    exit();
}

echo $Result;
$Result = json_decode($Result,true);
if(!$Result)
    exit('error');

# $Result variable contains object with keywords:
#      'success' [boolean true|false],
#      'token' [string token],
#      'job_status' [string "wait"],
#      'error' [string error],
# Exemplary reply (object in the $Result variable):
# {
#   "success":true,
#   "error":null,
#   "job_status": "wait",
#   "token": "fe57a3d3f9615891fe33e6b5ddaa540c"
# }

if($Result['success'] && $Result['job_status']=='wait'){
    $args=[];
    $args['apikey'] = $apikey;
    $args['action'] = 'get_result';
    $args['token'] = $Result['token'];
    $n=1;
    $job_status = 'wait';
    $ch = curl_init();
    while($n < 100 && $job_status=='wait'){ # query the server periodically, not more than 100 times
```

```

        usleep(500000); # delay 0.5sec between calls (not required but recognition takes some time to the server
        echo "\n request: $n \n";
        $n++;

        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_URL,$url);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
        $Result=curl_exec ($ch);
        if($Result===false){
            echo "request error\n";
            exit();
        }
        if($Result === false)
            break;

        $Result = json_decode($Result,true);

        if($Result && $Result['success']){
            $job_status = $Result['result'];
        }else{
            break;
        }
    }
    curl_close ($ch);
}
var_export($Result);

?>

```

Example 10. Python example code of the complete audio recognition cycle (Python 2.7):

```

import requests
import json
import time

filename = 'gazebo_12s.wav'
apikey = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key
must be 'active'
payload = {'action': 'identify', 'apikey': apikey, 'start_time':20, 'time_len':50}

result = requests.post('https://audiotag.info/api',data=payload, files={'file': open(filename, 'rb')})

print(result.text)
result_object = json.loads(result.text);

print(result_object);

if result_object['success']==True and result_object['job_status']=='wait' :
    token = result_object['token'];
    n=1;
    job_status = 'wait';
    while n < 100 and job_status=='wait':
        time.sleep(0.5) ;
        print('request:%d'%(n));
        n+=1;
        payload = {'action': 'get_result', 'token':token, 'apikey': apikey}
        result = requests.post('https://audiotag.info/api',data=payload)
        print(result.text);
        result_object = json.loads(result.text);
        print(result_object);
        if result_object.has_key('success') and result_object['success']==True:
            job_status = result_object['result'];

```

```
        else:
            break;
    else :
        print ('not success');

pretty_print = json.dumps(result_object, indent=4, sort_keys=True)
print(pretty_print);
```

4.4. Recognition of a remote audio file via direct URL

Audio identification of a remote file with a direct URL is performed using `'action:identify_offline_stream'`.

The following API keywords are used to form the identification query: `'apikey'`, `'action'`, `'url'`, `'base_time'`, `'overwrite'`. Refer to the list of API keywords in the Table 1.

The remote file must be publicly accessible via http(s) protocol by the URL specified with `'url'` keyword.

Native audio format used by the AudioTag engine is RIFF wave (.wav), PCM, 16 bit, 8 KHz, mono. The following file formats are also accepted and are converted internally: WMA, WAV, OPUS, OGG, MP3, MP2, M4A, FLAC, AMR, ALAC, AIFF, AIFAAC, 3GPP, 3GP. Please note that the file you upload must have one of the extensions from the list.

It is highly recommended to upload only audio files in AudioTag's native format (raw PCM, 16 bit, 8 KHz, mono). It puts less load on the AudioTag server and leads to faster API reaction on your recognition requests. You can use Ffmpeg tool (open-source) to convert other audio file formats into AudioTag's native format by executing the following command:

```
ffmpeg -i input_file.ext -ar 8000 -ac 1 -vn converted.wav
```

Minimal file duration accepted by the API is 12 seconds.

Maximal remote file size supported is 5000 Mb.

Communication with the AudioTag server is done asynchronously in the sense that the client should not stand by waiting for the file download process and identification result in response to the `'action:identify_offline_stream'` query. Instead of the identification result, in response to the query the client immediately receives a unique token(identifier) of the corresponding server job it created and can finish the connection with the server. The token should be used with another query, namely, `'action:get_result_offline_stream'` accompanied with the token (`'token:token'`) in order to check the identification process (job) status and get identification data in case the audio recognition result is ready. Refer to information in the Table 7.

Table 7. `'action:identify_offline_stream'` **reply** keywords

Keyword	Type	Description
success	boolean	Query result: true or false. False in next cases: see error description; True: all other cases;
token	string	Token to request recognition results using action <code>'get_result_offline_stream'</code>
error	String	This field contains textual information about error in the following cases: - when 'success' field is false - user API access key is not found - file format is not supported - file download has failed - extension of the remote file is invalid or the file is of zero size - remote file is too large - credit balance exhausted on the account - recognition failed

status	String	'wait' – while the request is being processed 'done' – the processing is finished.
--------	--------	---

Upon requesting 'action:get_result_offline_stream', the client receives JSON encoded array that contains the following fields: 'success', 'result', 'status'. Refer to information in the Table 8.

Table 8. 'action:get_result_offline_stream' reply keywords

Keyword	Type	Description
success	boolean	Query result: true or false. False in case of an error; True: all other cases;
error	string	This field contains textual information about an error in the following cases: - when 'success' field is false - error downloading file - file format is not supported - file's duration is too short - credit balance on the user's account is insufficient or exhausted - input file is not an audio file - parse file error - provided token is unrecognized
result	array	'link' – URL recognition result in textual/readable form; 'audio_duration' – duration in format H:M:S of source audio; 'filesize' – size of the source file in bytes; 'source_file' – original URL of the source audio file 'time_consumed' – time consumed by the recognition process (in seconds) 'audio_format' – format of audio file was received; 'json_data' – sub-array of recognition results, each entry is enumerated and contains the following entries: - 'time_start' – start of the recognized music, relative to the base_time as received as part of the - 'time_end' – end of the recognized music, relative to the base_time as received as part of the - 'confidence' – recognition score (reliability) - 'track' – array containing the following fields: track title, album title, artist, year; If nothing has been recognized, the 'json_data' array will be empty.
status	string	'wait' – while the recognition request is being processed 'done' – processing finished
progress	string	'download' – the remote file is being downloaded 'parse' – the request is being parsed number - % of processing done
base_time		absolute time (in format "Hours:Minutes:Seconds", e.g. 23:12:32) of the start of the offline audio stream, as received as part of the 'action:identify_offline_stream' query.

The token and the recognition data associated with it are valid for 48 hours from the job creation time. You can request recognition results for the token within this time after the initial `'action:identify_offline_stream'` query has been sent. If within this time the `'action:identify_offline_stream'` query is sent with the same URL, the query won't be reprocessed again since the result of its recognition already exists. If you want to force the system to re-download/re-process the audio for a URL, you can use `'overwrite'` keyword (send `overwrite=1`).

Upon repeated `'action:get_result_offline_stream'` query using the same `'url'` value, the API returns current status value (`'wait'` or `'done'`). Processing will occur again if there was any error with the last query.

A comprehensive example demonstrating remote file recognition is shown below. The example implements the both queries: `'identify_offline_stream'` and `'get_result_offline_stream'`. The example is divided into two logical parts correspondingly.

In the first part, `'remote_url'` specifies absolute URL of the remote file to be recognized. Its recognition is performed using the `'identify_offline_stream'` query. The query returns a result in JSON format with entries as described in the Table 7.

In case of success of the above query and non-empty content of the `'token'` field, the `'status'` field contains `'wait'` which is later replaced with `'done'` after the recognition is finished (or in case of an error).

In the second part of the example code, the `'status'` field is monitored while waiting for `'done'`.

Then, `'get_result_offline_stream'` is used to get the recognition results.

Example 11. PHP example code of the complete audio recognition cycle (PHP v5.5 and up):

```
<?php
# PART1 -----

# URL of a remote file to be recognized
$remote_url = 'https://audiotag.info/test-streams/audiotag-test-stream-40min-mono8khz.wav';

$url      = 'https://audiotag.info/api';
$action   = 'identify_offline_stream';
$RESULT   = [];

$args     = [];
$args['action'] = $action;
$args['apikey'] = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access
key, the key must be 'active'
$args['url'] = $remote_url;
$args['base_time'] = '13:00:00';

$ch = curl_init();
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
$Result = curl_exec ($ch);
$Error = curl_error($ch);
curl_close ($ch);

if(!$Result || json_decode($Result,true)===false){
    echo "no response\n";
}
```

```
        exit();
    }
    $j = json_decode($Result,true);
#print_r($j);
/*
    [success] => 1
    [result] => "accepted"
    [status] => "wait"
    [token] => ...given token used to receive result
*/

if(!$j || empty($j['success'])){
    echo "Error:". $j['error']."\n";
    echo "unsuccess\n";
    print_r($j);
    exit();
}
echo " result: {$j['result']}\n";
echo " token:  {$j['token']}\n";
echo " status: {$j['status']}\n";

$action = 'get_result_offline_stream';
$token  = $j['token'];
$url    = 'https://audiotag.info/api';

# PART2 -----

$RESULT=[];

$args          = [];
$args['action'] = $action;
$args['apikey'] = '1234567abcdef1234567890abcdef123';
$args['token']  = $token;

while(1){

    sleep(1);
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_URL,$url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $args);
    $Result = curl_exec ($ch);
    $Error = curl_error($ch);
    curl_close ($ch);

    if(!$Result || json_decode($Result,true)===false){
        echo " - no response from server\n";
        print_r($Result);
        break;
    }
    $j = json_decode($Result,true);
    if(!$j || !isset($j['status'])){
        print_r($j);
        echo " - receive unexpected response\n";
        break;
    }
    if($j['status']=='wait'){
        $progress = $j['progress'];
        echo " status: {$j['status']} $progress\t\t\r";
    }else{
        echo "\n status: {$j['status']}\t\t\n";
    }
    if($j['status']=='done'){
        print_r($j);
        #[success] => 1
        #[error] =>
```

```

    #[result] => Array
    # (
    #     [link] => https://audiotag.info/rsar/s1587633701d6eccccd859aa606f818
    #     [audio_duration] => 0:37:12
    #     [filesize] => 35708840
    #     [time_consumed] => 113
    #     [json] => [data.....]
    #     [source_file] => https://audiotag.info/streamtest.wav
    #     [audio_format] => WAV / WAVE (Waveform Audio)
    # )
    #
    #[status] => done
    #[progress] => 100
    #[base_time] => 12:00:00

    break;
}
}
exit();

```

Output obtained from running the example:

```

result: accepted
token: s9c43aeb21df90372002f94623c352b
status: done

status: done
Array
(
    [success] => 1
    [error] =>
    [result] => Array
        (
            [link] => https://audiotag.info/rsar/s9c43aeb21df90372002f94623c352b
            [audio_duration] => 0:40:00
            [filesize] => 38406650
            [time_consumed] => 102
            [source_file] => https://audiotag.info/test-streams/audiotag-test-stream-40min-mono8khz.wav
            [json_data] => Array
                (
                    [0] => Array
                        (
                            [time_start] => 13:00:21
                            [time_end] => 13:03:00
                            [track] => Array
                                (
                                    [0] => Cruel Summer (Digital Mix)
                                    [1] => Bananarama
                                    [2] => 12 Inch Collection
                                    [3] => 2006
                                )
                            [confidence] => 5503
                        )
                    [1] => Array
                        (
                            [time_start] => 13:03:00
                            [time_end] => 13:04:16
                            [track] => Array
                                (
                                    [0] => Bananarama - Cruel Summer
                                    [1] => Die Hit-Giganten (CD Series)
                                    [2] => Radioklassiker
                                    [3] => 2008
                                )
                            [confidence] => 1946
                        )
                )
        )
    )

```

```
)
[2] => Array
(
    [time_start] => 13:04:29
    [time_end] => 13:06:00
    [track] => Array
        (
            [0] => A-Ha - Cry Wolf
            [1] => Various - Pop Music (90`s-00`s)
            [2] => Halloween Hits
            [3] => 2008
        )
    [confidence] => 4624
)
[3] => Array
(
    [time_start] => 13:06:00
    [time_end] => 13:09:00
    [track] => Array
        (
            [0] => A Ha- Cry Wolf
            [1] => The Legends of World Rock (CD Series)
            [2] => 20 - Popularity
            [3] => 2001
        )
    [confidence] => 4340
)
[4] => Array
(
    [time_start] => 13:09:00
    [time_end] => 13:12:00
    [track] => Array
        (
            [0] => Forget Me Nots (Single Version)
            [1] => Patrice Rushen
            [2] => Straight From The Heart
            [3] => 1982
        )
    [confidence] => 6663
)
...
...
)
[audio_format] => WAV / WAVE (Waveform Audio)
)
[status] => done
[progress] => 100
[base_time] => 13:00:00
)
```

Example 10. Python example code of the complete audio recognition cycle (Python 2.7):

```
#PART 1 -----
from pprint import pprint
import requests
import json
import time

apikey      = '1234567abcdef1234567890abcdef123'; # generate and place here your unique API access key, the key
must be 'active'
remote_url  = 'https://audiotag.info/test-streams/audiotag-test-stream-40min-mono8khz.wav';

action      = 'identify_offline_stream';
url         = 'https://audiotag.info/api';

payload     = {'action':action, 'apikey': apikey , 'url':remote_url, 'base_time':'12:00:00' }

result = requests.post(url,data=payload)

json_object = json.loads(result.text);
if not json_object.has_key('success') or json_object['success']==False :
    print (result);
    print('unsucces');
    quit()
#pprint(json_object);

print(" result: "+json_object['result']);
print(" token : "+json_object['token']);
print(" status: "+json_object['status']);

#PART 2 -----

action      = 'get_result_offline_stream';
token       = json_object['token'];
url         = 'https://audiotag.info/api';

progress= 0;
n=1;
while n < 100 :
    time.sleep(0.5) ;
    print('request:%d'%(n));
    n+=1;
    payload = {'action': action, 'token':token, 'apikey': apikey}
    result = requests.post(url,data=payload)
#    print(result.text);
    json_object = json.loads(result.text);
    pprint(json_object);
    if not json_object.has_key('success') or json_object['success']==False :
        print('- no response from server')
        pprint(result)
        break;

    if not json_object.has_key('status') :
        pprint(json_object);
        print( " - receive unexpected response");
        break;

    if json_object['status']=='wait' :

        print(" status:"+json_object['status']+"; "+ json_object['progress'] + " \t\t\n");
    else :
        print(" status:"+json_object['status']+"; \t\t\n");

    if json_object['status']=='done' :
        pretty_print = json.dumps(json_object, indent=4, sort_keys=True)
        print(pretty_print);
        #[success] => 1
```

```
#[error] =>
#[result] => Array
# (
#     [link] => https://audiotag.info/rsar/s1587633701d6eccccd859aa606f818
#     [audio_duration] => 0:37:12
#     [filesize] => 35708840
#     [time_consumed] => 113
#     [json] => [data,.....]
#     [source_file] => https://audiotag.info/streamtest.wav
#     [audio_format] => WAV / WAVE (Waveform Audio)
# )
#
#[status] => done
#[progress] => 100
#[base_time] => 12:00:00

break;

print("quit");
quit();
```

5. Contacts

If you have questions, inquiries or error reports, you can drop an email to the following e-mail address:
`mailbox@audiotag.info`