SEARCH by Sound
Spectralmind Audio Intelligence Platform

API Documentation

Platform version: 1.3
Document version: 1.1.1

Authors: Wolfgang Jochum, Ewald Peiszer
Version

This is the documentation for the API version that is given on the cover page of this document. Version information can be retrieved using the provided API method.

Overview

Spectralmind’s SEARCH by Sound API is designed to minimize the integration effort of our customers and partners. Spectralmind has chosen REST\(^1\) as a paradigm to provide application developers with a simple interface that is easy to understand, implement and use.

The API can be queried by using HTTP requests. Specific parameters and options are added as query string to the request.

\[
\text{http://.../smintapi/[method]?[parameters/options]}
\]

In order to add or modify items we follow the REST principle to use HTTP PUT or POST requests. Likewise the API call to delete tracks needs to be a HTTP DELETE request.

Access

Currently access is granted for IP ranges. Different Access Methods can be provided by request.

Upload / Adding new Items

The system supports the \text{track/add} method that needs access to a given resource via HTTP.

API Response

The current version of the API returns XML documents. If the API call succeeds the HTTP status code will be 2xx. In case of an error a corresponding HTTP status code will be the result. For example if a track does not exist the API will return a 404 status code.

Future versions of the API may support more result document MIME types such as HTML – \text{text/html} or RDF - application/rdf+xml.

\(^1\) \url{http://en.wikipedia.org/wiki/Representational_State_Transfer}
Error Messages
All Error Messages are returned in the requested format. That means if an XML response was requested the error message will also be an XML document. Additionally the appropriate HTTP Status Code\(^2\) is used. The following list of status codes gives an overview on frequently used HTTP status codes of the SEARCH by Sound API. The list is not intended to be exhaustive.

HTTP Status Codes
- 200 OK: Successful request.
- 403 Forbidden: The request was legal, but could not be executed for other reasons. Possible reasons might be: overwriting a existing resource is forbidden, the user might not be allowed to execute this request, ...
- 404 Not Found: The URI is invalid or the resource requested does not exist.

Example of an error response:
```xml
<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <error code="403">
    Forbidden
  </error>
  <errorDetail>23505, 7, ERROR: duplicate key value violates unique constraint "smafejob_external_key_key"
  </errorDetail>
</smint>
```

Encoding
All data is and has to be UTF-8 encoded.

Request and URL Encoding / Parameter encoding
All parameters have to be converted to UTF-8 and URL encoded\(^3\).

Response Encoding
All response resources are UTF-8 encoded. This includes all error responses.


\(^3\) URL encoding / Percent encoding: [http://en.wikipedia.org/wiki/Percent_encoding](http://en.wikipedia.org/wiki/Percent_encoding)
Return Values
All return and request values are formatted corresponding to the following specification.

General Types:

<table>
<thead>
<tr>
<th>type</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>a url</td>
<td><a href="http://test.com/my/">http://test.com/my/</a></td>
</tr>
<tr>
<td>integer</td>
<td>a number</td>
<td>12345</td>
</tr>
<tr>
<td>boolean</td>
<td>boolean</td>
<td>true, false</td>
</tr>
<tr>
<td>string</td>
<td>string</td>
<td>this is a string</td>
</tr>
</tbody>
</table>

Specific Types:

<table>
<thead>
<tr>
<th>value</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>smint_track_id</td>
<td>any String</td>
<td>17943286224c73d9934d07d8.05617896</td>
</tr>
<tr>
<td>ext_track_id</td>
<td>any String</td>
<td>mytrack1234</td>
</tr>
</tbody>
</table>
API Methods

track/:smint_track_id
Returns a list of tracks that are similar to the given trackid.

- If the track exists and has been analyzed the result will be a list of similar tracks.
- If the track does not exist a 404 status code will be returned.
- If the track exists, but has not been analyzed yet the result will be a successful HTTP status code 200 and the result document will include a status element with information on the current status. (see XML Status Codes on next page)

The result contains an ordered list of related tracks, starting with the most relevant result. The value attribute describes the normalized distance between the query track and the result track. If the tracklist is empty a status element provides further details on the reason why there are no tracks in the result.

HTTP method:
GET

Parameters:

<table>
<thead>
<tr>
<th>name</th>
<th>required</th>
<th>data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>no</td>
<td>integer</td>
<td>(Maximum) number of related tracks to retrieve. Overwrites the server-side default count for the number of related track results to deliver.</td>
</tr>
<tr>
<td>distance_values</td>
<td>no</td>
<td>boolean</td>
<td>Determines whether to include distance_value for every track in the result set. If set to ‘false’ a ranked list will be retrieved. Overwrites the server default behavior.</td>
</tr>
<tr>
<td>external_keys</td>
<td>no</td>
<td>boolean</td>
<td>Determines whether to include the external_key value in the result document.</td>
</tr>
<tr>
<td>collection</td>
<td>no</td>
<td>string</td>
<td>Limits the query to a specific collection.</td>
</tr>
</tbody>
</table>
Examples:
The following examples uses the smint_track_id 5033184664c742a9fca9134.58351154.

/track/5033184664c742a9fca9134.58351154?count=3&distance_values=true

Result:
<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <query id="5033184664c742a9fca9134.58351154" collection="_d">
    <result id="8803880104c742aaf3c5516.04064305" value="238.243012"/>
    <result id="6295626594c742aa750c9d4.61222556" value="238.243012"/>
    <result id="286532314c742ab570d248.69055179" value="249.356536"/>
  </query>
</smint>

If there was a problem with the file or if the server has not finished analyzing the file a status element will be returned.

<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <query id="17793737764c742abc199276.47424858" collection="_d">
    <status status_code="2">job is processed by process: mydeamon</status>
  </query>
</smint>

Error Codes:

<table>
<thead>
<tr>
<th>HTTP Status Code</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>404 Not Found</td>
<td>If a track with given id does not exist.</td>
</tr>
</tbody>
</table>

XML Status Codes:

<table>
<thead>
<tr>
<th>XML Status Code</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Unknown</td>
</tr>
<tr>
<td>0</td>
<td>Successful. If no similar tracks (results) are in the result document this indicates, that the track was processed, but no similar tracks are found in the system.</td>
</tr>
<tr>
<td>1</td>
<td>Waiting for processing.</td>
</tr>
<tr>
<td>2</td>
<td>Ongoing processing.</td>
</tr>
<tr>
<td>3</td>
<td>Indexing.</td>
</tr>
</tbody>
</table>
**track_external_key/:external_key**
Returns a list of tracks that are similar to the given external key.

This method is identical to the track method that is used with smint_track_id but

- always returns the external_key values
- includes the external_key value in the query element

**Examples:**
The following example uses the external_key mykey2-A.

`track_external_key/mykey2-A?count=3`

**Result:**
```xml
<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <query id="17327124294c742abbeb5f53.34342440" external_key="mykey2-A" collection="_d">
    <result id="8015415404c742abbd14835.88000070" external_key=""/>
    <result id="17349204894c742abc0566b3.81365905" external_key="myotherotherkey1-d"/>
    <result id="6295626594c742aa750c9d4.61222556" external_key="url3"/>
  </query>
</smint>```
**track/add**
Add a Track by sending a URL where the track a) can be downloaded or b) providing a local file location (on the same server as the API is running).

**HTTP method:**
**PUT, POST**

The API does not distinguish between PUT and POST. But since there are some problems with clients that do not support PUT properly the API allows POST to be used instead.

**Parameters:**

<table>
<thead>
<tr>
<th>name</th>
<th>required</th>
<th>data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url (or file)</td>
<td>yes</td>
<td>URL</td>
<td>Defines the location where the file can be downloaded.</td>
</tr>
<tr>
<td>file (or url)</td>
<td>yes</td>
<td>File</td>
<td>A local file location.</td>
</tr>
<tr>
<td>external_key</td>
<td>no</td>
<td>String</td>
<td>This parameter allows adding an additional key to an uploaded track. If no external_key is given the track will not be accessible using external_key related requests, but only using the track/:smint_track_id method.</td>
</tr>
<tr>
<td>collection</td>
<td>no</td>
<td>String</td>
<td>Name of the collection the track should be added. If the track should be added to multiple collections the API call needs to be repeated for each collection.</td>
</tr>
</tbody>
</table>

**Example**:  
```bash
http://localhost/smintapi/track/add/
```

**Result:**
```
<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <track_added
    smint_track_id="4438668954c74000dee3580.04549407"
    url="http://www.soundpark.at/mp3/2005-07/moped2067_fight_your_fight_137691.mp3"
    external_key="mykey1234"/>
</smint>
```

---

4 Post/Put/Delete Examples are given using cURL. See [http://en.wikipedia.org/wiki/CURL](http://en.wikipedia.org/wiki/CURL) for more details on cURL.
Example:
curl -H "Accept: application/xml" -d "external_key=mykey4321&file=%2FUsers%2FShared%2Ftestsmall.mp3"
http://localhost/smintapi/track/add/

Result:
<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <track_added smint_track_id="13010202134c75a5eb9e11e5.44314348"
    file="/Users/Shared/testsmall.mp3"
    external_key="mykeylocal1234"/>
</smint>

The result contains the information sent to the server and the smint_track_id of the track. The track will be available as soon as the server analyzed the track. The result indicates only that the server accepted and queued the request, but does not guarantee that it can be analyzed. If the track cannot be analyzed, e.g. if it cannot be downloaded, the server will discard the request. To check if a track was processed simply access the track.

Error Codes:

<table>
<thead>
<tr>
<th>HTTP Status Code</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>403 Forbidden</td>
<td>If a track with the same key already exists.</td>
</tr>
</tbody>
</table>

track/delete/:smint_track_id
Removes a track from the system. This means it will be removed permanently, cannot be queried and will no longer appear as a query result of other tracks.

HTTP method:
DELETE

Example:
curl -X DELETE -H "Accept: application/xml"
"http://localhost/smintapi/track/delete/13010202134c75a5eb9e11e5.44314348"

Result:
<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <track_to_be_deleted
    id="13010202134c75a5eb9e11e5.44314348" collection="_d"/>
</smint>

The result just contains the information sent to the server. The track will be fully removed as soon as the server removed all links to related tracks. A result just means that the server queued the request, but does not guarantee that the track
was already removed from all result sets. To check if a track was removed simply access the track. If the result is a 404 status code the track and all references to the track (results) have been removed.

<table>
<thead>
<tr>
<th>HTTP Status Code</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>404 Not Found</td>
<td>If a track with given key does not exist.</td>
</tr>
</tbody>
</table>

**version**

Returns information on the API including the version.

**HTTP method:**

**GET**

**Example:**

```
/version
```

**Result:**

```
<?xml version="1.0" encoding="UTF-8"?>
<smint>
  <version major="1" minor="5" revision="134"/>
</smint>
```
System Requirements and Capabilities
(on-site installation only)
The core system engine, the database and the API have the following system
requirements (subject to change):

Hardware

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Any multicore Processor. Intel preferred, but not necessary.</td>
</tr>
<tr>
<td>Harddisk</td>
<td>Core system: 250 MB</td>
</tr>
<tr>
<td></td>
<td>Database: ~ 5 GB per 100,000 tracks (more if extensive search results are required)</td>
</tr>
<tr>
<td>Main Memory</td>
<td>Core system: 210 MB per 100,000 tracks (more if extensive search results are required)</td>
</tr>
<tr>
<td></td>
<td>Database: 8 GB (up to 100,000 tracks)</td>
</tr>
<tr>
<td></td>
<td>16 GB (more than 100,000 tracks)</td>
</tr>
<tr>
<td></td>
<td>32 GB (more than 300,000 tracks)</td>
</tr>
</tbody>
</table>

Software – Core Components
Main Platform: Debian Style Linux, (Ubuntu 8.04 or greater, Debian 5)
Additional Platform: Mac OSX running on Intel based Mac 10.5 or greater
Experimental Support: Windows

API
PHP 5.2.x or greater, PDO-Postgres driver installed
Apache 2.x (optional enabled option AllowEncodedSlashes for external_key values containing slashes)

Performance
Since audio analysis requires a reasonable time use the following table to project
your initial runtimes. Note that once a track is indexed there will only be additional computation effort, if a new track is added.

<table>
<thead>
<tr>
<th>Performance</th>
<th>Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk mode (One time catalog injection)</td>
<td>~ 5.8 days per 100,000 tracks (based on mp3 files)</td>
</tr>
<tr>
<td>Regular mode (track update)</td>
<td>(Startup of core component finished)</td>
</tr>
<tr>
<td>1,000 tracks in database</td>
<td>~ 3 seconds / track</td>
</tr>
<tr>
<td>10,000 tracks in database</td>
<td>~ 10 seconds / track</td>
</tr>
<tr>
<td>300,000 tracks in database</td>
<td>~ 50 seconds / track</td>
</tr>
</tbody>
</table>

5 http://httpd.apache.org/docs/2.0/mod/core.html#allowencodedslashes
Limitations
The core software can generally be used with corpora until a size of 500,000. It is continuously adapted and optimized for larger corpora. Please contact our support if you intend to use larger corpora.

Note that, besides technical limitations, license restrictions may limit the number of tracks that can be added.

Audio files types
Spectralmind platform uses ffmpeg/libavcodec to provide support for a wide variety of audio file formats. The ffmpeg package can be optionally installed during setup.
Key formats include mp3, aac and ogg vorbis.
For a complete list see libavcodec documentation\(^6\).

FAQ
Just a short list of common questions:

- Why does my external key with slashes not work?
  - enable the Apache Option AllowEncodedSlashes\(^5\)

---

\(^5\) https://en.wikipedia.org/wiki/Libavcodec#Implemented_audio_codecs

\(^6\) https://en.wikipedia.org/wiki/Libavcodec#Implemented_audio_codecs