audio commons

Deliverable D4.8
Second prototype tool for the automatic semantic description of music pieces

Grant agreement nr	688382
Project full title	Audio Commons: An Ecosystem for Creative Reuse of Audio Content
Project acronym	AudioCommons
Project duration	36 Months (February 2016 - January 2019)
Work package	WP4
Due date	31 July 2018 (M30)
Submission date	15 August 2018 (M31)
Report availability	Public (X), Confidential ( )
Deliverable type	Report ( ), Demonstrator (X), Other ( )
Task leader	QMUL
Authors	Johan Pauwels
Document status	Draft ( ), Final (X)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

Executive Summary

As part of the Audio Commons Ecosystem, a number of tools are provided for the automatic analysis of audio content without the need for human intervention. These tools are designed for extracting i) musical audio properties for music pieces and music samples, and ii) non-musical audio properties for any kind of sounds. Work-in-progress versions of these tools have been released in parallel for the first prototype in the Audio Commons Ecosystem (ACE).

To improve the integration into the Audio Commons Ecosystem, the second prototype of this tool for the semantic description of music pieces has improved upon the work previously detailed in D4.3 and D4.5 and made it accessible as a web service. Its goal is to provide audio-based descriptors for music pieces to complement already available metadata and make retrieval based on the audio descriptors possible. This web service is not meant to be used by end-users directly, but should be integrated with the mediator developed in WP2 during the final part of the project, when it will also be merged into the Audio Commons Ontology.

Finally, although the current audio analysis service is entirely developed in house, it is also being used as a test case to explore how future third parties can best offer their analysis software as a service in the Audio Commons Ecosystem. This has the potential for creating a marketplace not only for content but also for services within the ACE. Deployment considerations relevant for this case are also discussed in this report.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1 Audio analysis service principles

The analysis service is created following the function-as-a-service (FaaS) paradigm. It is a natural fit because requesting a descriptor for a music piece is a triggered event that does not require a long running process. The server load is also variable, depending on the number of requests, and FaaS programs are inherently easy to scale to adapt to the demand at any time.

The FaaS framework that is being used is OpenFaaS (https://www.openfaas.com/), which relies heavily on the Docker container platform. This allows to easily turn any piece of software, regardless of programming language or dependency, into a web function. Since Docker is already used for other parts of AudioCommons, reuse of previously developed components will be comparatively easy.

In practice, the audio analysis service takes the form of a single API endpoint. For development and demonstrative purposes, it can currently be called directly, but it isn't intended to be exposed to end users as such. Rather it should be integrated with the mediator developed in WP2 by the end of the project, which can then call the service whenever the descriptors it provides are required.

The focus of this service is to demonstrate how to make the large number of audio descriptors already available in our toolkits (including the Essentia and Vamp frameworks) easily integrated into the Audio Commons Ecosystem. This includes content analysis algorithms coupled with confidence measures as reported in D4.5 and algorithms with a complex set of dependencies, such as deep learning-based instrument identification reported in D4.3.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

2 Using the audio analysis service

2.1 General

The current version of the service can be called by using the endpoint http://c4dm-xenserv-virt5:8080/function/ac-analysis. One of the advantages of offering an audio analysis web service as an API is that it can be used by a wide variety of setups and programming languages. We will illustrate the usage of the service in this report by calling the API manually using the command-line tool curl, which simply prints the response of the query. In more realistic use-cases, the API will be called programatically and its output processed further.

The API call always takes three parameters:

- a content provider (either "jamendo" or "freesound")
- an id in the content provider's namespace
- a descriptor (allowed values in the current version are "chords", "instruments", "beats-beatroot" or "keys")

Depending on the deployment configuration (discussed in section 4), the audio gets downloaded from the content provider on-demand or will be read locally. It then gets analysed by the tool that corresponds to the requested descriptor, as explained in section 3. All descriptors get stored in a database such that they only need to be calculated once and such that the generated metadata can be published in the Audio Commons Ecosystem (ACE). This database gets filled on-demand, but for some applications it would be useful to have a database of descriptors for an entire collection. To achieve this scenario, it suffices to call the API once for each audio file in the collection, optionally asynchronously as described in section 5. This initial API call could for instance be integrated in the upload mechanism of Audio Commons, such that as soon as a file is published in the ACE, all its descriptors get calculated.

The service is closely tied to the content providers in the ACE. As such, no provision is made to analyse files that are not already made available in the ACE, for instance local files. For those use-cases, the local analysis tool developed in D4.7 can be used.

2.2 Content type of the response

An example call to the API requesting the chords of the file on Jamendo with `id` 1498355 looks as follows:

```
curl -v
"http://c4dm-xenserv-virt5:8080/function/ac-analysis?provider=jamendo&id=1498355
&descriptor=chords"
```

And has the following response:

```
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8081 (#0)
> GET
/function/ac-analysis?provider=jamendo&id=1498355&descriptor=chord
s HTTP/1.1
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

```
> Host: localhost:8081
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Length: 125
< Content-Type: text/plain; charset=utf-8
< Date: Fri, 06 Jul 2018 16:16:37 GMT
< X-Call-Id: 9bb7324a-79cd-467a-a6a4-45cfdb0be33a
< X-Duration-Seconds: 0.345971
< X-Start-Time: 1530893797042208353
<
Unknown content type "None" requested. Allowed content types are:
['application/json', 'text/plain', 'text/rdf', 'text/csv']
* Connection #0 to host localhost left intact
```

As you can see from the error message, a content-type also needs to be requested. Following the common web convention, it is done by passing an HTTP header. This leads to the desired result:

```
curl -v
"http://c4dm-xenserv-virt5:8080/function/ac-analysis?provider=jamendo&id=1498354
&descriptor=chords" -H "Content-Type: application/json"
```

```
{'confidence': 0.8284330681425113,
 'duration': 289.04,
 'chordSequence': [{'start': 0.0, 'end': 16.85, 'label': 'Dbmaj'},
{'start': 16.85, 'end': 23.05, 'label': 'Ebmin7'}, {'start':
23.05, 'end': 25.15, 'label': 'Abmaj'},
...
{'start': 286.55, 'end': 289.0, 'label': 'Fmaj7'}]}
```

3 Web service components

Since some audio descriptors may take a substantial amount of time to compute, a caching mechanism has been implemented such that a descriptor for a file needs to be computed only once and subsequent requests can be served from the caching database. Therefore there are three components that form the audio analysis service

- a dispatcher function
- a caching database
- a number of worker function that do the actual descriptor calculation

The actual endpoint is a lightweight dispatcher function that inspects incoming requests, verifies if the desired descriptor is available in the caching database and retrieves it from there if possible. If not, it forwards the request to the worker function that is appropriate for the requested combination of content provider and descriptor. The newly computed descriptor is then stored in the caching database for future reuse and returned as response to the query.

The current caching database is implemented using a MongoDB (https://www.mongodb.com/) instance because it's fast, easy to use and its document-oriented organisation is a natural fit for the file-id-based organisation of the content providers. The choice of database is not critical however, and could be relatively easily swapped for a different database technology if required (for instance to integrate into an already existing setup). Once the caching database has been filled with an entire collection, it can also be used to make new retrieval scenarios and pattern mining based on the descriptors possible.

Due to the use of Docker as underlying infrastructure for the worker functions, a great deal of flexibility is available to match descriptors to worker functions. A single function can calculate one or many descriptors and the service can be made up from one to many worker functions, but it makes most sense to group descriptors into containers according to technical requirements. Descriptors that require the same runtime environments should be grouped together in one container, and can be isolated from other descriptors with conflicting environments. Increasing the number of functions improves code isolation and therefore maintainability, but at the same time the output of multiple functions needs to be kept in sync such that it's not noticeable that they were produced by different functions. Therefore a balance needs to be struck between more and less worker functions.

For the current demonstration purposes, three worker functions calculating four descriptors are available:

- A container with sonic-annotator and all officially released Vamp plugins as detailed on the Vamp website (https://vamp-plugins.org/download.html). In the prototype, only a "keys" and a "beats-beatroot" descriptor are exposed, but it's trivial to add more Vamp plugin outputs to the service.
- A container with the "chords" descriptor that includes a confidence measure as previously developed for the AC project and described in D4.5
- A container with another instance of sonic-annotator and the VamPy instrument identification plugin described in D4.3 because of its specific and demanding runtime environment

Some example calls including different content types and content providers are displayed below.

```
curl -v
"http://c4dm-xenserv-virt5:8080/function/ac-analysis?provider=freesound&id=26317
1_4338788&descriptor=beats-beatroot" -H "Content-Type: text/csv"
```

Calling the "beats-beatroot" descriptor returns the location in seconds of the beat onsets in a file:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

```
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8081 (#0)
> GET
/function/ac-analysis?provider=freesound&id=263171_4338788&descriptor=beats-beat
root HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Type: text/csv
>
< HTTP/1.1 200 OK
< Content-Length: 543
< Content-Type: text/csv
< Date: Mon, 06 Aug 2018 12:06:03 GMT
< X-Call-Id: c93ba0b3-d095-4f20-bf5b-61d1ff72a0c2
< X-Duration-Seconds: 1.360694
< X-Start-Time: 1533557162086853471
<
"http://freesound.org/data/previews/263/263171_4338788-hq.ogg",0.050000000
,0.440000000
,0.880000000
…
,15.000000000

* Connection #0 to host localhost left intact
```

```
curl -v
"http://c4dm-xenserv-virt5:8080/function/ac-analysis?provider=jamendo&id=1498354
&descriptor=instruments" -H "Content-Type: text/rdf"
```

The “instruments” descriptor returns a 26-dimensional vector that gives the probabilities of the file containing the following instruments: ['Shaker', 'Electronic Beats', 'Drum Kit', 'Synthesizer', 'Female', 'Male', 'Violin', 'Flute', 'Harpsichord', 'Electric Guitar', 'Clarinet', 'Choir', 'Organ', 'Acoustic Guitar', 'Viola', 'French Horn', 'Piano', 'Cello', 'Harp', 'Conga', 'Synthetic Bass', 'Electric Piano', 'Acoustic Bass', 'Electric Bass']

```
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8081 (#0)
> GET /function/ac-analysis?provider=jamendo&id=1498354&descriptor=instruments
HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Type: text/rdf
>
< HTTP/1.1 200 OK
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 688382

```
< Content-Length: 1887
< Content-Type: text/rdf
< Date: Fri, 06 Jul 2018 16:20:34 GMT
< X-Call-Id: 25414794-f8f5-492d-b605-23bd206750bc
< X-Duration-Seconds: 0.860671
< X-Start-Time: 1530894033564568323
<
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix mo: <http://purl.org/ontology/mo/> .
@prefix af: <http://purl.org/ontology/af/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix event: <http://purl.org/NET/c4dm/event.owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#> .
@prefix vamp: <http://purl.org/ontology/vamp/> .
@prefix : <#> .


:transform_0_instrument-probabilities a vamp:Transform ;
    vamp:plugin
<file:///home/app/transforms/instrument-probabilities.n3#transform_plugin> ;
    vamp:step_size "512"^^xsd:int ;
    vamp:block_size "2048"^^xsd:int ;
    vamp:sample_rate "22050"^^xsd:float ;
    vamp:plugin_version """2""" ;
    vamp:output [ vamp:identifier "instrument-probabilities" ] .

:event_type_1 rdfs:subClassOf event:Event ;
    dc:title "Instrument probabilities" ;
    dc:format "" ;
    dc:description "Probabilities of all possible instruments" .

<https://flac.jamendo.com/download/track/1498354/flac> a mo:AudioFile ;
    mo:encodes :signal_2.

:signal_2 a mo:Signal ;
    mo:time [
        a tl:Interval ;
        tl:onTimeLine :signal_timeline_2
    ] .

:signal_timeline_2 a tl:Timeline .

:event_3 a :event_type_1 ;
    event:time [
        a tl:Interval ;
        tl:onTimeLine :signal_timeline_2 ;
        tl:beginsAt "PT111.827301587S"^^xsd:duration ;
        tl:duration "PT5.015510204S"^^xsd:duration ;
    ] ;
    vamp:computed_by :transform_0_instrument-probabilities ;
    af:feature "0.00255305 0.0390834 0.598441 0.136329 0.00411661 0.0029797
0.00774392 0.000163014 0.00569463 0.00287672 0.000373543 0.0258499 0.00308203
0.00193787 0.000533148 0.00029215 0.00255066 0.00721903 0.000228554 0.000688069
0.147141 0.000246749 0.0049805 0.00489513" .

* Connection #0 to host localhost left intact
```

The three components of the service can be configured into multiple architectures, each with different benefits, which will be discussed in the next section.

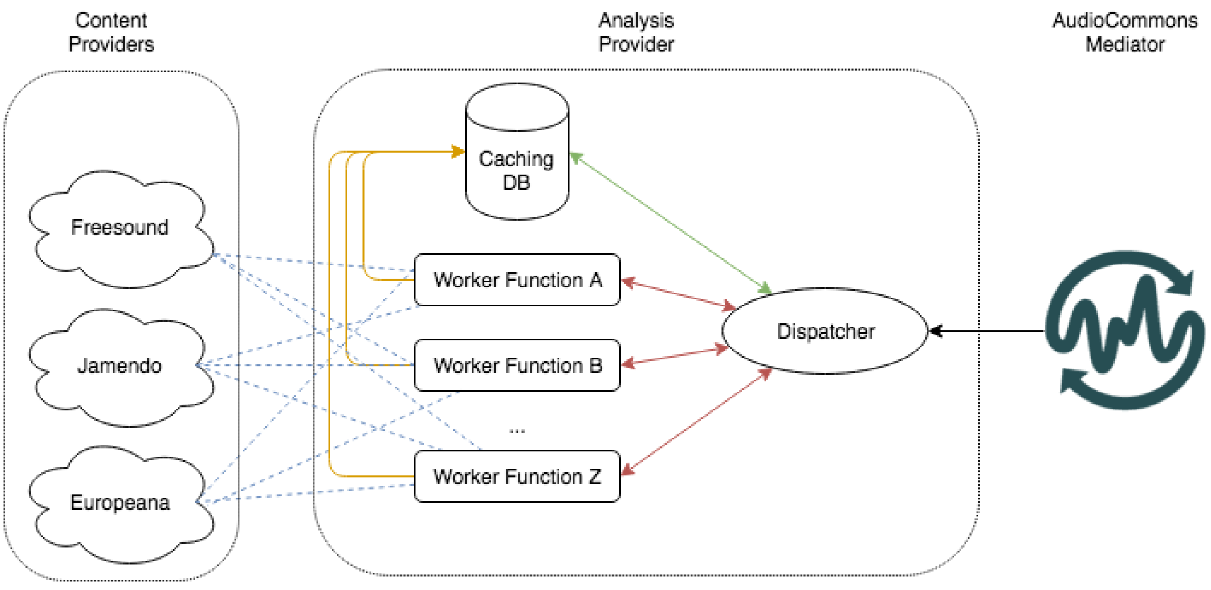
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

4 Suggested deployment configurations

Because of practical and legal reasons, the hosting of the different components of the analysis service can be spread over multiple physical sites. Especially in the light of potential third parties wanting to offer their analysis software as a service in the Audio Commons Ecosystem, it is important to be able to accommodate different requirements. A number of potential setups are therefore possible, which will be discussed in this section. In short, the three components -- the dispatcher, the worker functions and the caching database -- can be hosted centrally or distributed to the content providers in different combinations. Common to all cases is that the audio content is hosted by the respective providers and the dispatcher is called by the mediator.

4.1 Fully centralised

The analysis provider takes care of all hosting and retrieves the audio files from the different content providers.



4.1.1 Advantages

- + No FaaS infrastructure and database hosting required at content providers
- + Centrally maintained, so no risk of different content providers running different versions of analysis software and easier to update
- + Central database can be used for monitoring and data mining applications

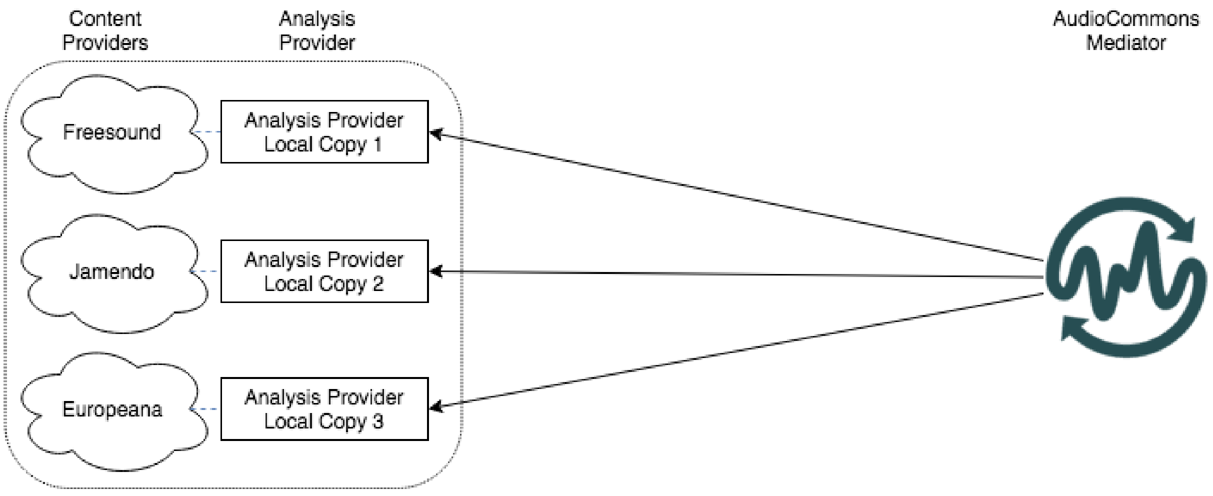
4.1.2 Disadvantages

- - Audio content needs to be transferred over the internet (bandwidth and/or latency issues)
- - Needs database hosting and FaaS infrastructure by analysis provider

4.2 Maximally distributed

Essentially the same as running multiple service providers, but all are copies of the same service and are restricted to processing audio of one specific content provider.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382


Content Providers
Analysis Provider
AudioCommons Mediator
Freesound
Analysis Provider Local Copy 1
Jamendo
Analysis Provider Local Copy 2
Europeana
Analysis Provider Local Copy 3

4.2.1 Advantages

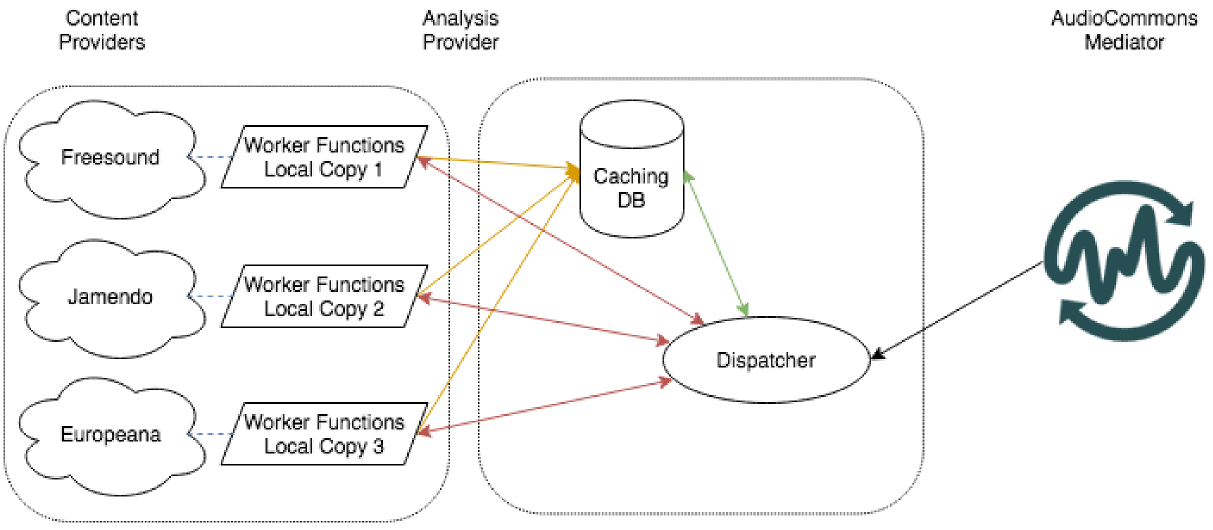
+ No database hosting or FaaS infrastructure required at analysis provider
+ Audio content can be processed locally, which avoids transfer of audio over internet (faster, less bandwidth required and potentially legally more straightforward).

4.2.2 Disadvantages

- FaaS infrastructure and database hosting required at content providers
- Updates need to be rolled out separately at each of the content providers, which means more work, but can also lead to multiple versions of the analysis software being present in the ecosystem if updates don't happen concurrently.

4.3 Central database and dispatcher

A hybrid version where the worker functions run at the content providers, but are centrally dispatched.



4.3.1 Advantages

+ Audio content can be processed locally, which avoids transfer of audio over internet (faster, less bandwidth required and potentially legally more straightforward).
+ Central database can be used for monitoring and data mining applications

4.3.2 Disadvantages

- FaaS infrastructure needed both at content providers and analysis provider
- Updates need to be rolled out separately at each of the content providers, which means more work, but can also lead to multiple versions of the analysis software being present in the ecosystem if updates don't happen concurrently.

4.4 Configuration

Further mixtures between these configurations are possible, for instance by hosting some worker functions at the content providers and keeping some central. For the ongoing development, the fully centralised approach is chosen, but the code can be adapted to any other setup. The configuration only involves specifying the available content providers and the URI format for their respective audio files, whether they are local paths or web-accessible URLs.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

5 Asynchronous calling

Another benefit of using OpenFAAS is that it comes with a lot of useful features out-of-the-box such as scaling, monitoring using Prometheus, etc. One of the most relevant is that it also supports asynchronous calling, which can be useful because some of the descriptors can take minutes to calculate (the first time, that is, the next ones will be retrieved from the caching DB).

One potential scenario would be to make an asynchronous query for all descriptors whenever a new file is uploaded to the AudioCommons ecosystem, such that the descriptors are calculated and ready for synchronous retrieval out of the cache. The example below shows how to do asynchronous querying, with the option to provide a callback url.

```
curl -v -X POST
"http://c4dm-xenserv-virt5:8080/async-function/ac-analysis?provider=jamendo&id=1
498356&transform=chords" -H "X-Callback-Url: http://notify.me.when.done" -H
"Content-Type: application/json"
```

```
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8081 (#0)
> POST /async-function/ac-analysis?provider=jamendo&id=1498356&transform=chords
HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.54.0
> Accept: */*
> X-Callback-Url: http://notify.me.when.done
> Content-Type: application/json
>
< HTTP/1.1 202 Accepted
< Date: Fri, 06 Jul 2018 16:26:10 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

6 Conclusion and future work

In this report, the next generation of the tool for automatic semantic description of music pieces is presented, which are now provided as a web service to ease the integration into the Audio Commons Ecosystem. It is based on the function-as-a-service paradigm and consists of three components: a dispatcher function, a caching database and multiple worker functions that reuse work previously developed for Audio Commons. Future work will focus on hardening the service against abuse such that it can be made world-accessible (although it is not meant to be accessed directly by end-users anyway).

Furthermore, the output of the different worker functions needs to be made more uniform and integrated with the Audio Commons mediator. After this stage, more descriptors can be added and the work from the MTG on music samples can be included as well. Finally, the use of the caching database for enabling new sorts of content-based queries needs to be explored.

In the next deliverable D4.11, such a new user application that relies on a database of chord descriptors will be discussed to demonstrate and evaluate the usefulness of the analysis service.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382