Deliverable 5.3 First LinkedTV End-to-end Platform

Jan Thomsen, Ali Sarioglu, Sacha Weinberg, Rolf Fricke (CONDAT)

Version: 08.04.2013

Work Package 5: LinkedTV Platform

LinkedTV

Television Linked To The Web

Integrated Project (IP)

FP7-ICT-2011-7. Information and Communication Technologies

Grant Agreement Number 287911

Dissemination level[1]	*PU*
Contractual date of delivery	*31st March 2013*
Actual date of delivery	*08.04.2013*
Deliverable number	*D5.3*
Deliverable name	*First LinkedTV End-to-end Platform*
File	*LinkedTV_D5* 3.docx
Nature	*Prototype*
Status & version	*V1.0*
Number of pages	*43*
WP contributing to the deliverable	*WP5*
Task responsible	*CONDAT*
Authors	*Jan Thomsen (Condat)*
Other contributors	*Ali Sarioglu, Sacha Weinberg, Rolf Fricke (Condat)*
Reviewer	*José Luis Redondo Garcia* *EURECOM*
EC Project Officer	*Thomas Küpper*
Keywords	*Integration Platform, Architecture, ConnectedTV, Workflow, Services*
Abstract (for dissemination)	*This Deliverable describes the first version of the LinkedTV end-to-end platform, which intergrates a whole workflow from video ingestion over video analysis, annotated media fragment generation, content enrichment to personalized playout by a dedicated media player.*

[1] • PU = Public
• PP = Restricted to other programme participants (including the Commission Services)
• RE = Restricted to a group specified by the consortium (including the Commission Services)
• CO = Confidential, only for members of the consortium (including the Commission Services)

Table of Contents

List of Figures

List of Tables

1 Introduction

This Deliverable describes the First LinkedTV End-to-end-Platform as it is due M18 (March 2013). The End-to-end Platform consists of the whole LinkedTV workflow with the steps: 1) video ingestion and import of metadata into the platform, 2) video analysis, 3) generation of annotated media fragment URIs generation, 4) enrichment, 5) retrieval of related content, 6) hyperlinking of related content, 7) playout of enriched and hyperlinked videos, and 8) personalisation and contextualisation. The first version of the end-to-end platform comprises a first integration of most of the above mentioned workflow steps which basically demonstrates the workflow from video ingestion to enriched playout, while not yet incorporating some features of the final platform (due M24), most importantly the full automization of the process and the personalisation and contextualisation features.

This Deliverable mainly concentrates on the technical description of the components and interfaces of the platform, thus also functioning as an integration manual and reference for dealing with the LinkedTV platform. For a thorough description of the overall LinkedTV workflow and architecture refer to Deliverable *D5.1 The LinkedTV Platform Architecture*. This document, however, does not cover the technical details of the single components and services within these different workflow steps themselves (e.g. the different tools and services for video analysis or personalisation). These again are subject of the respective Deliverables. However, these components will be listed and references to their documentation are given.

1.1 Related Deliverables

D1.2 Visual, text and audio information analysis for hypervideo, first release

D2.2 Specification of lightweight metadata models for multimedia annotation

D2.3 Specification of Web mining process for hypervideo concept identification

D3.4 LinkedTV interface and presentation engine version 1

D5.1 LinkedTV platform and architecture

D5.2 LinkedTV front-end: video player and MediaCanvas API

1.2 History of the document

Date	Version	Name	Comment
2013/01/12	0.1	Jan Thomsen	Document created
2013/03/17	0.2	Jan Thomsen	prefinal version uploaded to the Wiki
2013/03/18	0.3	Sacha Weinberg	Proofreading and corrections
2013/03/25	0.4	Jan Thomsen	Final version for internal QA
2013/04/02	0.5	Jan Thomsen	Final version, internal QA feedback included
2013/04/08	1.0	Jan Thomsen	Feedback from Scientific Coordinator included

Table 1: History of the document

2 The LinkedTV architecture

As depicted in Figure 1 the LinkedTV system basic can be understood as consisting of horizontal, or temporal, dimension following the workflow of the processing from ingestion to enriched playout, and a vertical dimension, consisting of the basically three conceptual layers: a) the backend LinkedTV Platform with Metadata Repository, Integration Components and REST API interfaces; b) the different specialied automatic services for analysis, annotation, Media Fragment generation, etc. and c) a user layer which includes all the tools and components for user interaction and consumption, be it domain experts, editors or the end user or viewer, respectively.

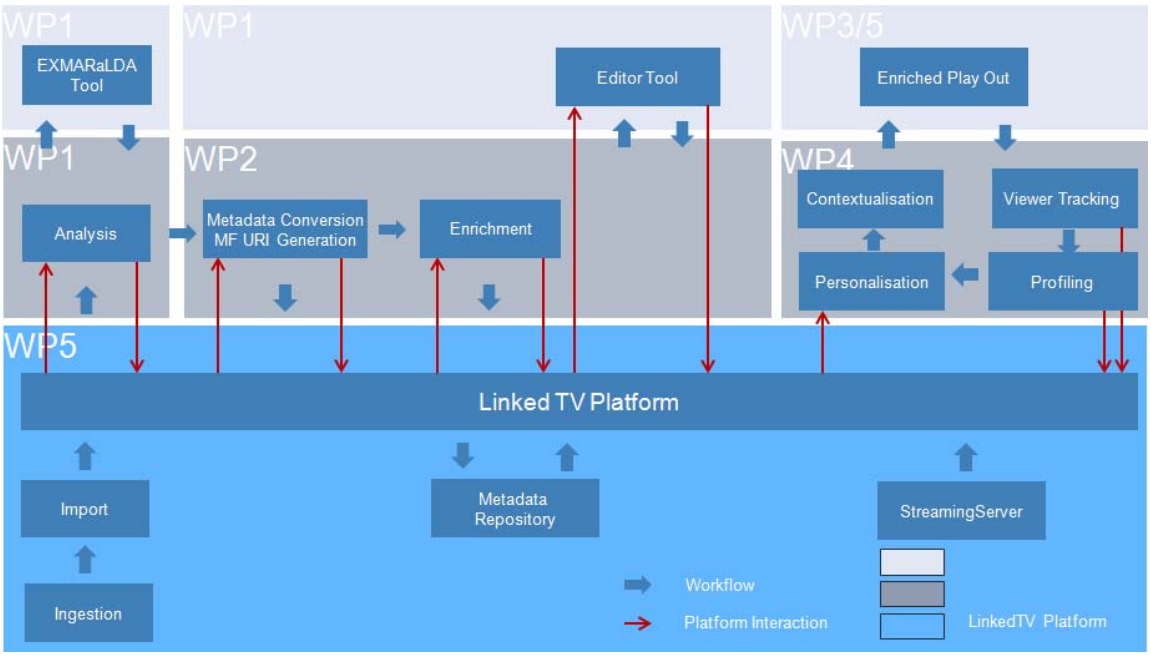

Figure 1: LinkedTV Platform Architecture Overview

Subject of this Deliverable is the description of the current state of development of the lower level, here labelled “WP5”, together with the interaction with the other components (i.e. mainly the red arrows).

The components of the LinkedTV Platform Backend are mainly the following:

a) The Linked Media Service Bus, an integration framework based on Enterprise Service Bus technology. This covers the functionalities for importing, file transfer, notification, message routing, event publishing, service integration, scheduled processing and more.

b) The Metadata Repository, which is basically an RDF Triple Store (more precisely, a Quad Store)

c) REST API interfaces for accessing and updating all data within the platform

Additionally, internal components are used such as an SQL Database for the administration and status management of complete media resources.

The following table provides an overview of the components employed by the LinkedTV Platform.

Component	Technology
RDF Triple Store	Openlink Virtuoso Universal Server Release 6.4
SQL Database	Openlink Virtuoso Universal Server Release 6.4
ESB/Integration	Apache Camel 2.10 (planned)
Libraries used	ELDA, Jersey, Junit, JSTL, Jackson, Spring, SLF4J, Hibernate, Tuckey

Table 2: Components of the LinkedTV Backend Platform

3 Platform Integration

The following Section describes the details of the first end-to-end platform, i.e. the different components of the backend platform, as well as the integration aspects for each basic workflow step or tool.

3.1 General Integration Strategy

This subsection outlines the general strategy for the integration of services and tools within the LinkedTV end-to-end platform.

Cascading Integration and Distribution

While conceptually the platform interfaces are designed in such a way that services can be integrated individually, the LinkedTV services are grouped together in functional collections which are first integrated within these groups. E.g. for the different services for video analysis, like automatic speech recognition, visual concept detection or subtitle extraction are integrated through an own orchestration service, and interact only through entry and exit points with the platform (for more details cf. Section 3.2).

These functional collections are grouped along the work package structure of LinkedTV, mainly within the workpackages 1) WP1 for video analysis, 2) WP2 for metadata generation and content enrichment, and 3) WP4 for personalisation and contextualisation.

This approach reflects also the distributed, cloud-oriented nature of the LinkedTV services: the different services can be hosted by the respective partner, and also external services can be integrated into the workflow as well.

REST API Compliance

All platform services are exposed via REST API functions under the domain http://api.linkedtv.eu/. This includes the basic REST Services for Create, Retrieve, Update and Delete operations and also requires that all external LinkedTV services which need to be called from the LinkedTV platform components themselves also offer a REST API to interact with.

The REST API of the LinkedTV platform not only provides the basic platform interaction functionality, but moreover ensures the integration with the overall platform services like monitoring, event notification, logging, authentication, etc.[2]

[2] These overall services are not part of the First End-to-end Platform.

SPARQL Endpoint

Additionally, a SPARQL endpoint is provided for direct SPARQL queries under http://data.linkedtv.eu/sparql. However, this SPARQL endpoint should be used only for retrieval as there is no integration with the above mentioned platform services taking place here when using it for updating the Triple Store; updating via direct SPARQL update is only advisable under very limited and controlled conditions during test phases.

Open integration by event notification and message routing

The LinkedTV platform is on open integration platform. This means that rather than through a central workflow orchestration, individual or grouped services are integrated via an event and notification based mechanism. Table 3 summarizes the events as currently defined; they can be extended in the future. Events are qualified by the component which issues the event.

Component	Event	Description
MediaResourceImporter	MEDIA_RESOURCE_IMPORTED	A new MediaResource has been downloaded or uploaded to the LinkedTV FTP-Server
MediaResourceImporter	MEDIA_RESOURCE_CREATED	The Metadata belonging to the MediaResource has been created; the MR is now ready for video analysis as well as for transfer and encoding to the playout StreamingServer; the Platform internal MediaResourceID is generated and can be retrieved
MediaServer	MEDIA_RESOURCE_TRANSFERRED	A copy of the MR has been transferred to the StreamingServer
StreamingServer	MEDIA_RESOURCE_ENCODED	A new encoding of the MediaResource has been generated; the URI has to be returned
MediaResourceAnalyzer	ANALYSIS_STARTED	The video analysis component has started the processing of the MediaResource; during the processing further individual events can be issued (shot segmentation, face recognition, etc.)
MediaResourceAnalyzer	ANALYSIS_COMPLETED	The video analysis component has finished the analysis

Component	Event	Description
MediaResourceAnalyzer	EXMARALDA_FILE_GENERATED	the according EXMARaLDA file has been generated and URI has been stored in the Database
MetadataConverter	MF_URI_GENERATION_STARTED	The generation of Media Fragment URIs has started
MetadataConverter	MF_URI_GENERATION_FINISHED	The generation of MediaFragment URIs has finished, named entities and annotations have been generated
MetadataConverter	RDF_DATA_STORED	The according RDF data has been stored in the TripleStore

Table 3: LinkedTV service events

Figure 2 shows a high-level sequence diagram for the analysis and annotation workflow.

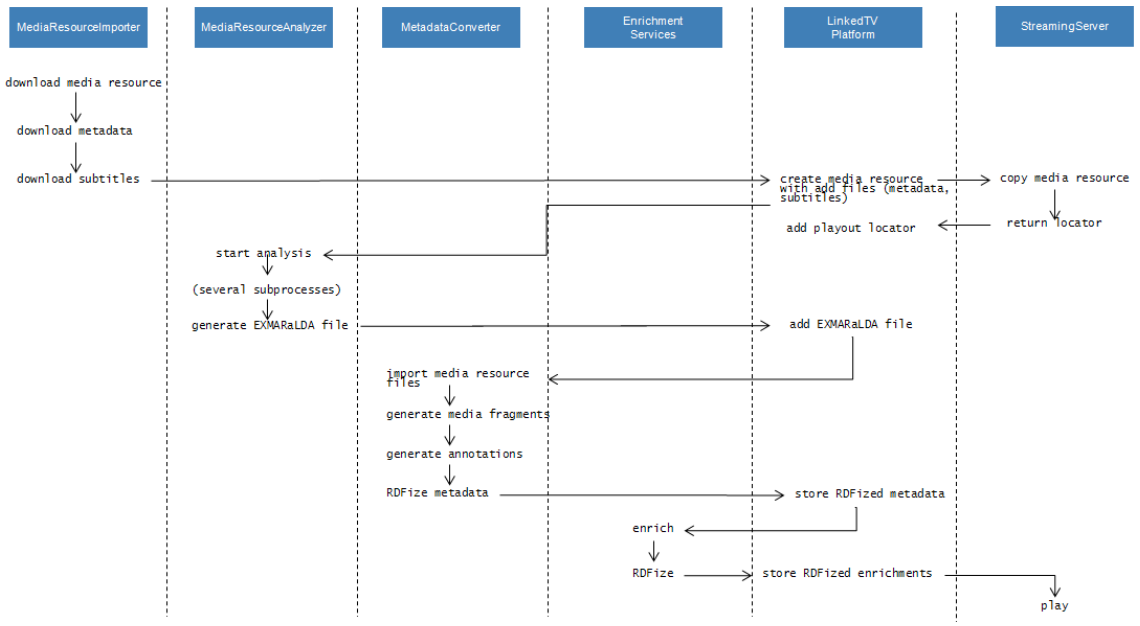

Figure 2: Sequence Diagram for the Analysis and Annotation Workflow

At this moment this list of events contains only events of the beginning of the process. The events for subsequent steps including enrichment, hyperlinking, editor's tool and personalization will be defined during the further integration of these components.

Notification Mechanisms

In order to get notified about a relevant event the LinkedTV platform provides three basic types of event consumption and notification:

1. *Scheduled Pull*: all resources and their respective status are exposed via a REST API which can then be called on a timed basis. Example: when the MediaResourceAnalyzer component wants to know when it can start processing a new media resource, it just needs to implement a feature which issues a REST call `GET /mediaresource?status=MEDIA_RESOURCE_IMPORTED` on a scheduled basis (e.g. daily during development, every 5 minutes in a final production environment) to get all the respective media resources. This is the easiest way for a triggered execution of processing steps.
2. *Direct Messaging*: Secondly, the LinkedTV platform will support the routing of event-based messages directly to service endpoints which are registered beforehand. With the underlying Apache Camel framework a lot of standard endpoints from SMTP, FTP, over HTTP and RESTful services up to Amazon or Google cloud services are already supported. In the example above this means that the MediaResource-Analyzer has to provide a REST service like
   `PUT http://analyzer.linkedtv.eu/mediaresource/{id}`
   which will then be called directly upon the event `MEDIA_RESOURCE_CREATED`.
3. *Syndicated Push/Pull*: As a third method the LinkedTV Platform will provide a syndication service based on the Atom protocol[3]. Every relevant event will be published on a LinkedTV feed site under http://atom.linkedtv.eu. A single entry looks like (concrete feed URI and entry details subject to change):

```
<entry>
    <id>...</id>
    <title>MEDIA_RESOURCE_IMPORTED</title>
    <updated>$date</updated>
    <author>MediaResourceImporter</author>
    <ltv:mediaresourceid>$id</ltv:mediaresourceid>
    ...
</entry>
```

   Every client that wants to get informed about relevant events can poll that feed and select the respective events or authoring components. So, while this from the platform (or producer) side this is a push service, from the client/consumer side this is pull service.

[3] http://www.atomenabled.org/

Events and status updates by LinkedTV services

The methods described above apply to the direction LinkedTV Platform →LinkedTV services. For the opposite direction there is only one method foreseen, which is the general LinkedTV REST API. I.e. for every communication of an external component or service towards the platform there is a respective RESTful URI with POST, PUT or DELETE HTTP methods providing the functionality.[4] The platform integration components then themselves handle all subsequent steps, e.g. publishing the respective event in the Atom feed.[5]

Following the example above, this means that upon finishing processing the MediaResource-Analyzer calls `POST http://api.linkedtv.eu/mediaresource/{id}` with the information about the event, uri of the generated EXMARaLDA file, etc. in the body.[6]

3.2 Integration of LinkedTV services

Within this section the integration details of the different LinkedTV services following the general LinkedTV workflow are described.

3.2.1 Video Ingestion and Import

The first step is the ingestion of media resources into the LinkedTV.[7] Within the first end-to-end platform this is done via a custom download component which is also usable for uploading of video content to the platform by content providers. Currently, there are daily news shows from rbb being downloaded to the platform in order to start the LinkedTV workflow. Content from the AVRO Tussen&Kunst show is uploaded on an irregular basis.

The directory structure is as follows:[8]

```
\data\{provider}          currently rbb or sv
        \mp4              the video source files
        \tva              TVA metadata
        \srt              subtitles
        \exm              Exmaralda file generated by the Analysis workflow
```

Additional subdirectories might be added in the future.

[4] As the full REST API is not functional within the first end-to-end platform, updates by directly using the SPARQL interface are also possible, but will be replaced in the final end-to-end platform by RESTful updates.

[5] Although the Atom protocol as an HTTP-based method also allows for PUT, POST and DELETE operations, it principally would be possible that external services directly update the LinkedTV feed. However, in order to avoid synchronisation problems this method will not be provided.

[6] For the REST API specification cf. Section 5.

[7] mainly performed by WP5.

[8] Access to the directory is restricted to members of the Consortium only.

With the final end-to-end platform the video ingestion process will be fully integrated into the Linked Media Service Bus by defining the providers as Apache Camel endpoints.

The video URI and according metadata are automatically stored in the LinkedTV Platform Administration Database. They are accessible under the URI http://api.linkedtv.eu/mediaresource. For consumption by REST clients different formats are exposed by adding an extension, e.g.

http://api.linkedtv.eu/mediaresource.html	returns HTML
http://api.linkedtv.eu/mediaresource.xml	returns XML
http://api.linkedtv.eu/mediaresource.json	returns JSON

The one media resource which has been created can be retrieved via the call

http://api.linkedtv.eu/mediaresource/{id}

For getting specific mediaresources the calls can be filtered by indicating relevant properties, e.g.
http://api.linkedtv.eu/mediaresource.json?publisher=rbb&status=MEDIA_RESOURCE_CREATED.json returns the list of all mediaresources from publisher *rbb* with status MEDIA_RESOURCE_CREATED in JSON format.

At this stage the metadata is not yet RDFized, so no RDF format is available here. For an overview of the properties of media resources and which can be retrieved via cf. subsection 4.1.1. Of course, only those properties will be inserted into the AdministrationDB which are available through attached metadata. Figure 3 shows the HTML version of the media resource overview, and Figure 4 the detail view.

When the import of the media resource is finished this is notified by a status update

`MediaResourceImporter.MEDIA_RESOURCE_CREATED <uri>`

After that the following subsequent steps can be performed in parallel:

1. `MediaResourceAnalyzer.Init(uri):` start the video analysis processing of the file
2. `StreamingServer.CopyEncode(uri)`: transfer a copy of the video to the final playout media server and encode

In a later production environment initial location and playout location may be the same, in which case the MediaResourceImporter just consists of inserting the metadata and this step would be omitted completely.

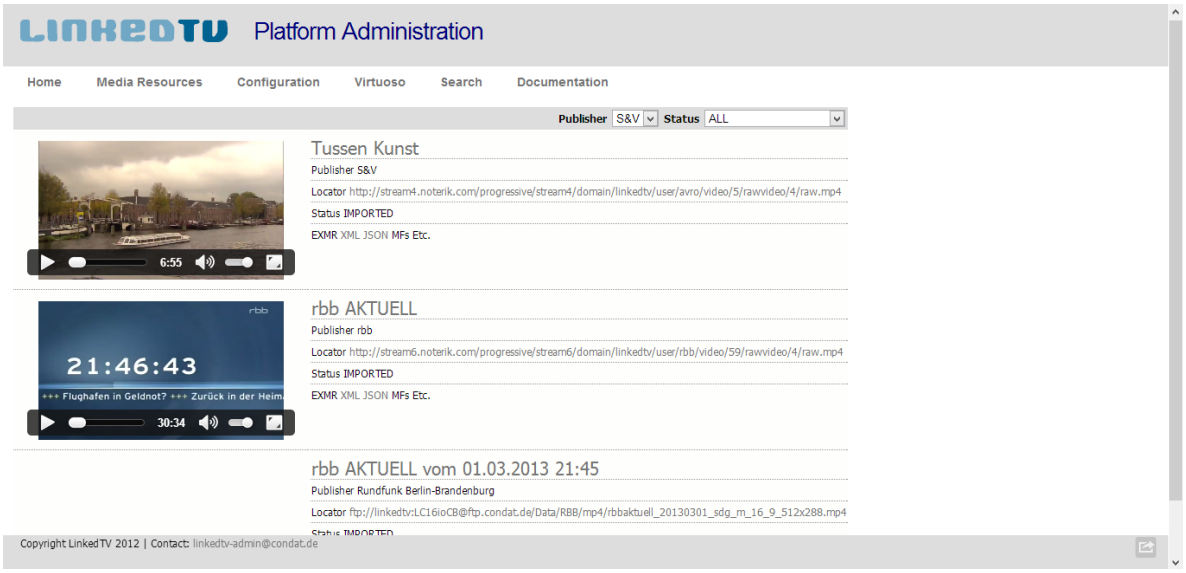

Figure 3: (Screenshot) Media Resources at LinkedTV Platform Administration

Figure 4: (Screenshot) Media Resources Detail View with Related Content

3.2.2 Video analysis

The general strategy for the integration of the different video analysis tools provided by WP1 for automatic speech recognition, concept recognition, subtitle extraction, etc. follows the cascading integration approach as described in Section 3.1. The main integration of the different video analysis components is done by an own orchestration service, the LinkedTV Analysis Manager (LAM, also known as Exmaralda Service) which also handles the communication with the platform. As the analysis tools heavily depends on the video sources themselves which have to be locally available, there is no real distributed processing possible, because this always would require a physical transfer of the video sources. Furthermore, some components are implemented as submodules of the EXMARaLDA tool.

Figure 5 provides an overview of the communication architecture.

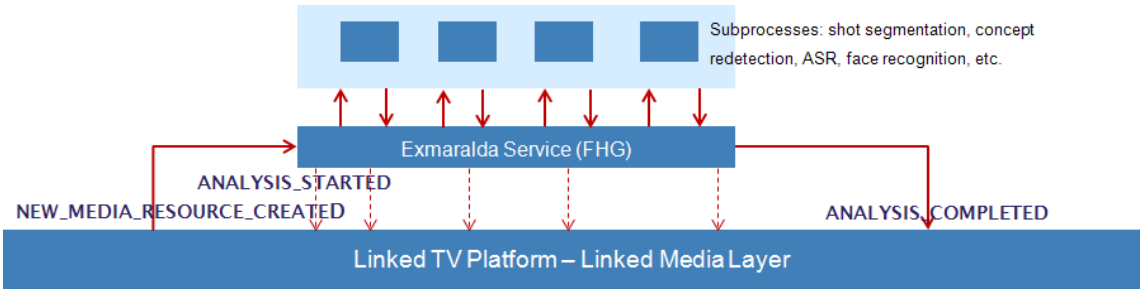

Figure 5: Integration of Video Analysis Components

The interaction process between the LinkedTV Backend Platform (Linked Media Layer) and the video analysis components looks as follows:

1. LAM receives a `NEW_MEDIA_RESOURCE_CREATED` message by one of three methods described above, e.g. by being called via
   `PUT http://analyzer.linkedtv.eu/mediaresource/{id}`
2. LAM transfers/copies media resource to a local server (optional)
   LAM issues event `MEDIA_RESOURCE_COPIED`
3. LAM issues event `ANALYSIS_STARTED`, maybe parametrized with the types of analyses performed
4. LAM calls internal administration tools, EXMARaLDA subcomponents or services; no data interaction with the LinkedTV Backend Platform occurs during processing; individual status updates can be issued, however (e.g. `ASR_COMPLETED`)
5. Manual processing via EXMARaLDA tool (optional)
6. When completed and consolidated:
   - LAM generates final EXMARaLDA file and stores it via ftp in the platform subdirectory `\data\{publisher}\exm`

- LAM calls
  `POST /mediaresource/{id}?agent=LAM&status=MEDIA_ANALYSIS_COMPLETED`
  additional properties like URI for EXMARaLDA file in the body

The following table lists the different tools which are integrated by the LinkedTV Analysis Manager.

Component	Responsible Partner	Technology
EXMARaLDA	Beeld en Geluid	Standalone tool based on Python
Keyword Extractor	UEP	Web based service, REST API
Face Recognition	EURECOM	C++ Tool; REST API or EXMARaLDA module possible
Object Redetector	CERTH	EXMARaLDA module
Video/Shot Segmentation	CERTH	EXMARaLDA module
Concept Detector	CERTH	EXMARaLDA module
Audio Analyzer	Fraunhofer	EXMARaLDA module

Table 4: Video Analysis subcomponents

3.2.3 Transfer and encoding of media resources to the playout server

Within the initial LinkedTV platform locations for video sources for analysis and playout differ. For playout media resources are made available through a dedicated Media Streaming Server which is part of the platform itself and hosted by LinkedTV partner Noterik. Parallel to the video analysis the original video sources have to be transferred to the Streaming Server and to be encoded into different formats or resolutions. For the subsequent steps it is important that the locator information of the final playout URI is known. Thus the process looks as follows:

1. The StreamingServer API receives the `NEW_MEDIA_RESOURCE_CREATED` message by one of three methods described above, e.g. by being called via
   `PUT http://StreamingServer.linkedtv.eu/mediaresource/{id}`
2. The StreamingServer transfers a copy of the media resource
   StreamingServer issues an event `MEDIA_RESOURCE_COPIED`

3. The StreamingServer API creates a new media resource on the LinkedTV platform by calling
   `PUT http://api.linkedtv.eu/mediaresource?locator=<uri>`[9]
   If the return code is successful (200) then an id is returned
4. The StreamingServer API relates the original media resource and the playout resource by calling (POST, not all attributes are given here):
   `http://api.linkedtv.eu/mediaresource/{oldID}?relatedContentID={newID}&..`
   `http://api.linkedtv.eu/mediaresource/{newID}?relatedContentID={oldID}&..`

Figure 6 shows the general workflow between analysis, transfer and media fragment URI generation.

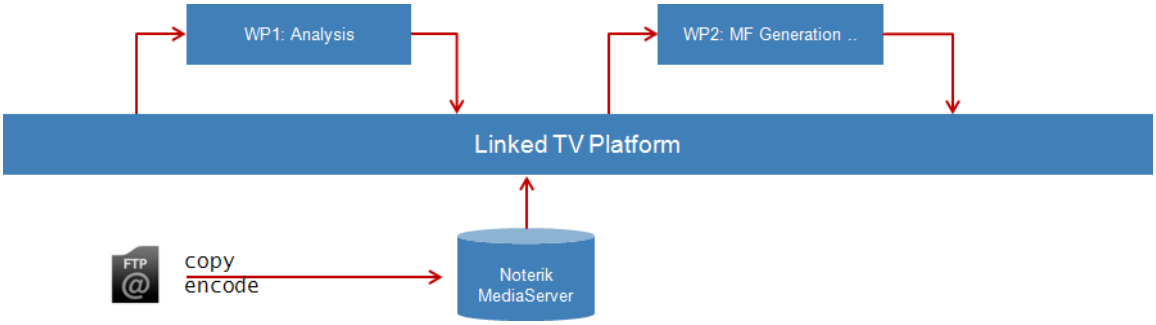

Figure 6: General Workflow between Analysis, Transfer and Media Fragment Generation

3.2.4 Metadata Generation and Enrichment

After the completion of the video analysis and the copy of video sources to the streaming server the next important step of the LinkedTV workflow is the generation of annotated media fragment URIs according to the LinkedTV ontology which relies (among other) on MF URI 1.0 and Open Annotation 1.0 standards.[10,11]

The integration of services for generation of annotated media fragment URIs and enrichment also follows a cascading integration strategy. In contrast to the video analysis process, however, these do not rely on the video sources themselves but on textual metadata only. By

[9] At present, also instances of the same content, the copy of the transferred media resource will be treated as a new media resource on its own; this approach might be changed in the future if appropriate.

[10] http://www.w3.org/2008/WebVideo/Fragments/WD-media-fragments-spec/

[11] http://www.openannotation.org/

this nature they can be arbitrarily distributed and cloud based. The services employed by this process in fact use all kind of resources on the web, mainly Linked Open Data resources.

The internal integration of the different services for the generation of annotated media fragment URIs, named entity recognition, entity enrichment and hyperlinking will be described in detail in the upcoming Deliverables *D2.4 Annotation and retrieval module of media fragments* and *D2.5 Specification of the Linked Media Layer* (both due September 2013).

As all of the services operate over RDF data by just constantly adding new RDFized information, the interaction with the LinkedTV backend platform is quite simple by using the LinkedTV SPARQL endpoint or the RDF REST API.

The general workflow has two main parts:

1. *Metadata Generation*: The initial generation of annotated media fragments, which is done once (Metadata Generation Service)
2. *Content Enrichment:* The subsquent enrichment and hyperlinking process which can be done several times (Content Enrichment Service)

The workflow for the first part looks as follows:

1. For the starting of the Metadata Generation Service two conditions must be met:
   a) The video analysed is finished (`EXMARALDA_FILE_GENERATED=TRUE`); this includes the availability of subtitle and metadata files.
   b) The media resource has been copied to the Streaming Server and the playout URI is known (`MEDIA_RESOURCE_ENCODED=TRUE`)
   The Metadata Generation Service gets notified of these starting conditions by one of the methods described above, e.g. through a listening service like
   `GET http://api.linkedtv.eu/mediaresource?status=MEDIA_RESOURCE_ENCODED`
   or by providing REST API which can be called, e.g.
   `PUT http://linkedtv.eurecom.fr/mediaresource/{id}`
2. The Metadata Generation Service retrieves the EXMARaLDA, as well as subtitles and metadata files as available.
3. The Metadata Generation Service the process of generating the media fragment URIs from the EXMARaLDA file, the annotations, named entities etc. and finally RDFizing the results according to the LinkedTV Data Model as described in D2.3 .
4. The Metadata Generation Service stores the generated RDF data in the LinkedTV Virtuoso Triple Store either directly through the SPARQL endpoint or through the RDF Update REST API[12] and issues a `MF_URIs_GENERATED` event:
   `POST /mediaresource/{id}?status=MF_URIs_GENERATED&mediaID={mediaID}`

[12] not available in the first version of the end-to-end platform

`mediaID` is the UUID generated by the metadata converter and which will be used to relate media fragments to the media resources.

5. The LinkedTV Platform API updates the status of the media resource accordingly and relates the `mediaID` with the original `ID`.

The workflow for the Content Enrichment Service is looks as follows:

1. Several individual services like the Mediafinder, the Unstructured Search Module or the Linked Service (cf. Table 5) are running either permanently in the background or can be called on demand.
2. Through a merging component (the MediaServer) the results of the different services are serialized according to the general Linked TV model as described in D2.3 and related to the named entities recognized by the Metadata Generation Service.
3. The merged results are RDFized and stored as additional annotations over the pre-existing media fragments according to the LinkedTV ontology in the Metadata Repository.
4. Due to the often very time-critical nature of related content to events, people, news etc. it might be sometimes more appropriate to not store RDFized permanently in the Metadata Repository but to retrieve it on demand. This will be further investigated and specified within D2.5.

Figure 7 gives an overview of the whole process.

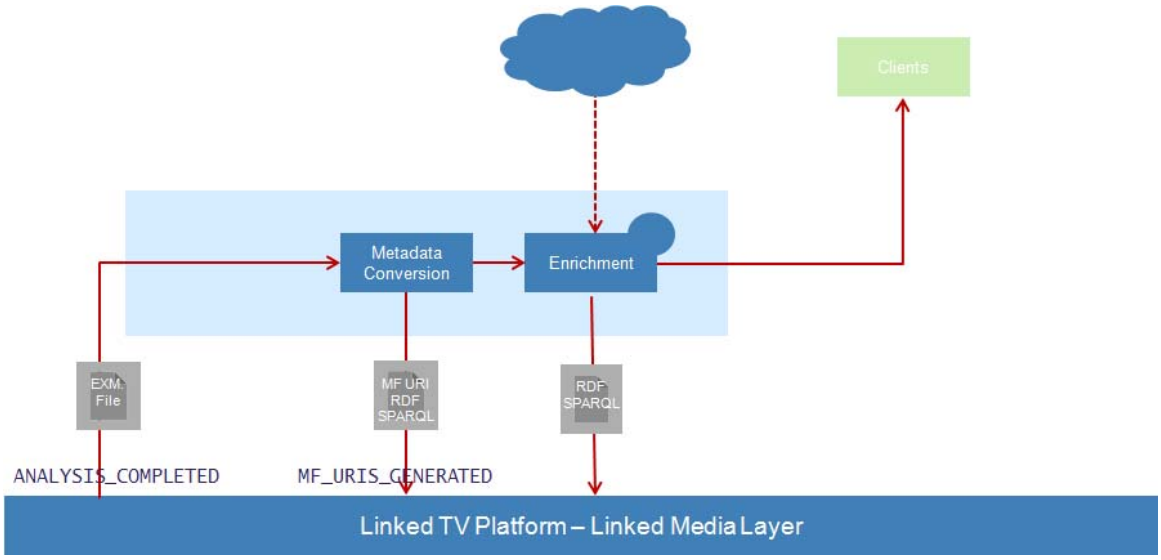

Figure 7: Integration of services for metadata generation and content enrichment

The following table gives a preliminary overview over the tools and services as employed by WP2; more services will be added in due course.

Component	Responsible Partner	Technology
NERD	EURECOM	REST API
Metadata Converter	EURECOM	REST API+SPARQL
Targeted Hypernym Discovery	UEP	REST API
Unstructured Search Module	UEP	REST API
MediaServer	EURECOM	node.js Service, REST API
MediaFinder	EURECOM	REST API, not ready for public disclosure
Linked Service Infrastructure	STI	REST API

Table 5: Services and Tools for Metadata Generation, Enrichment and Hyperlinking

3.2.5 MediaPlayer

Within this subsection the general integration of the LinkedTV Media Player is described. For a description of the MediaPlayer itself cf. *D5.2 LinkedTV front-end: video player and MediaCanvas API.* The following description does not include the personalisation and contextualisation features; these are described in Section 3.2.6. The MediaPlayer is developed within WPs 3 and 5.

The simple, non-personalized version of the Media Player is a read-only component which basically does two things with respect to the platform:

1. Retrieve and display the media resource from the LinkedTV Media Streaming Server
2. Retrieve and display the related content, i.e. annotations and related (hyperlinked) content

The general workflow here is:

1. A user wants to see a video which is delivered over the Internet, i.e. by some URI as `http://sample.com/video.mp4`, which the player starts to display
2. The MediaPlayer retrieves the LinkedTV mediaID by the call (Base URI for all subsequent calls is `http://data.linked.tv`
   `GET /mediaresource?locator=http://sample.com/video.mp4` which returns (when existing) an ID like `6d4f04ba-92dc-4787-9a0a-48ba60fe0b2d`

3. While playing, the player retrieves the annotations and the related content via the associated media fragments from the LinkedTV Platform API; e.g. for time interval from seconds 30 to seconds 50 it first retrieves the mediafragments of the given media resource by: `GET /mediaresource/{id}/mediafragment&min-temporalStart=30.0&max-temporalEnd=50.0`
   and then retrieves the related annotations by
   `GET /mediafragment/{id}/annotation`
4. The related content can be retrieved by
   `GET /mediafragment/{id}/relatedcontent`
5. The user may interacting via clicking, pausing, playing forward, etc.

Figure 8 shows an overview of this process.

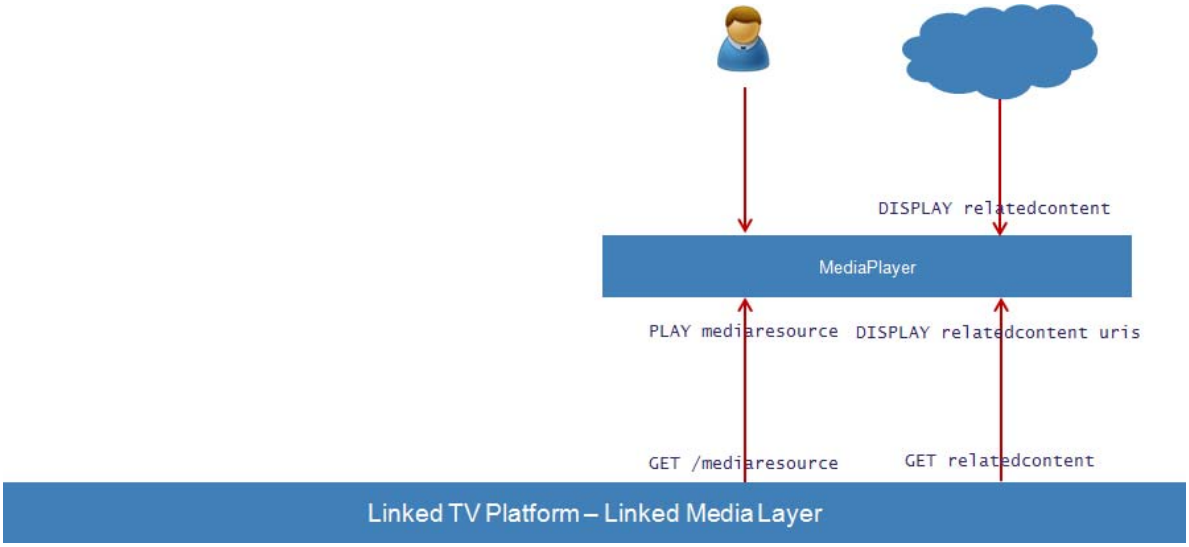

Figure 8: The general interaction between the MediaPlayer and the backend platform

3.2.6 Personalisation and Contextualisation

Within this subsection the integration of the personalisation and contextualiation is described. Personalisation and contextualisation[13] consists mainly of the following two subprocesses:

1. *Data gathering*: user behavior tracking, automatic or manual getting user interests, getting viewing context data
2. *Personalisation and contextualisation*: adapting the displayed content through different parameters to the user's preferences.

[13] Performed by WP4.

For a detailed description cf. deliverables *D4.1 Specification of user profiling and contextualisation*. *D4.2 User profile schema and profile capturing*, *D4.3 Content and concept filter v1, D4.4 User profile and contextual adaptation* (Sep 2013).

The process described in the following is still very preliminarily and abstract as personalisation and contextualisation is not integrated within the first end-to-end platform and it also has yet to be specified what data is stored and computed on the platform or on the client, respectively, and how the exact interaction between client and platform will look like exactly.

The general, preliminary workflow is described as follows:

1. Qualified User watches video content
2. While displaying video the MediaPlayer retrieves related content on a timed basis (or caches it in advance)
3. All related content is filtered through a personalisation service
4. MediaPlayer displays related content dependent on viewing context (e.g. 2nd Screen, overlay annotations)
5. User interacts (clicking, pausing, forward, etc.)
6. Profiling Data achieved through
   Automatic User Behaviour Tracking
   Explicit Feedback (Like/Dislike, Starred, Rated)
   Personal Editing, requires registration.

Figure 9 shows an overview of the process.

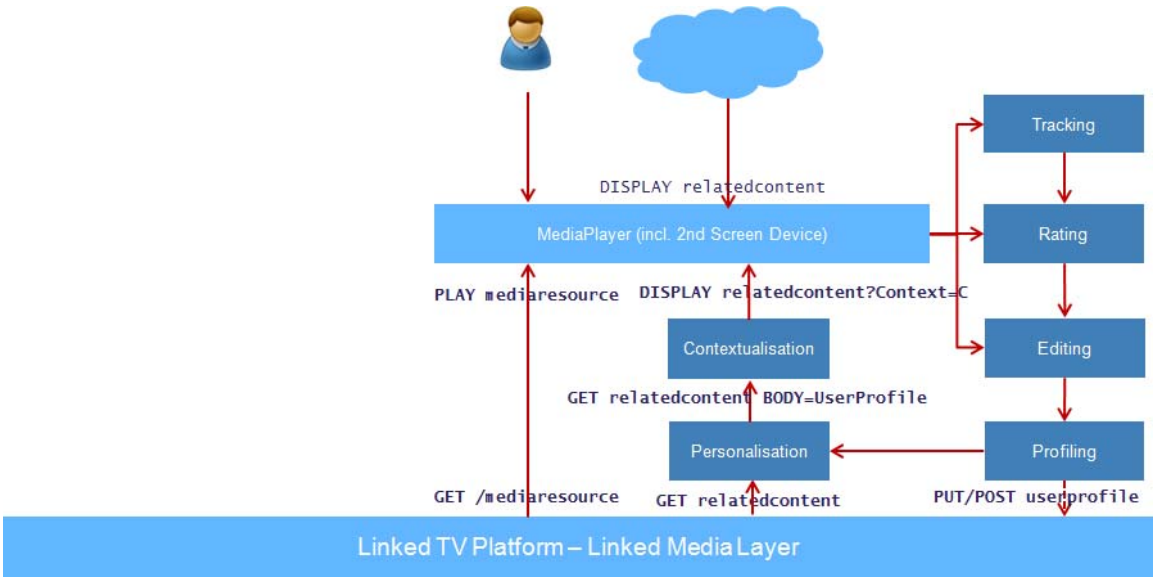

Figure 9: The personalisation and conceptualisation interaction

Component	Responsible Partner	Technology
LUMO	CERTH	Ontology
GAIN	UEP	REST API
I:ZI Miner	UEP	REST API
LUME	Fraunhofer	Editing Tool
f-pocketKRHyper	CERTH	REST API planned
Interest Tracker	UMONS	C++ Tool
Beancounter	STI	REST API
Profile Learner / Profile API	CERTH	to be defined

Table 6: Services and Tools for Personalization and Contextualization

3.2.7 Editor Tool

The LinkedTV Editor Tool is a specialized Web-based tool which allows broadcast editors to check results of video analysis, metadata generation and enrichment and mark these as false or irrelevant, to edit and add additional annotations and related content to media fragments, and to provide further functionalities. Currently, requirements are collected and the functionalities are being specified. The editor tool will be realized by LinkedTV partner Beeld en Geluid until Sep 2013.

The Editor Tool presupposes the results of the Metadata Generation Service and writes back additional annotations to the platform which are qualified by the provenance information. Figure 10 shows the overall integration of the Editor Tool into the Platform workflow.

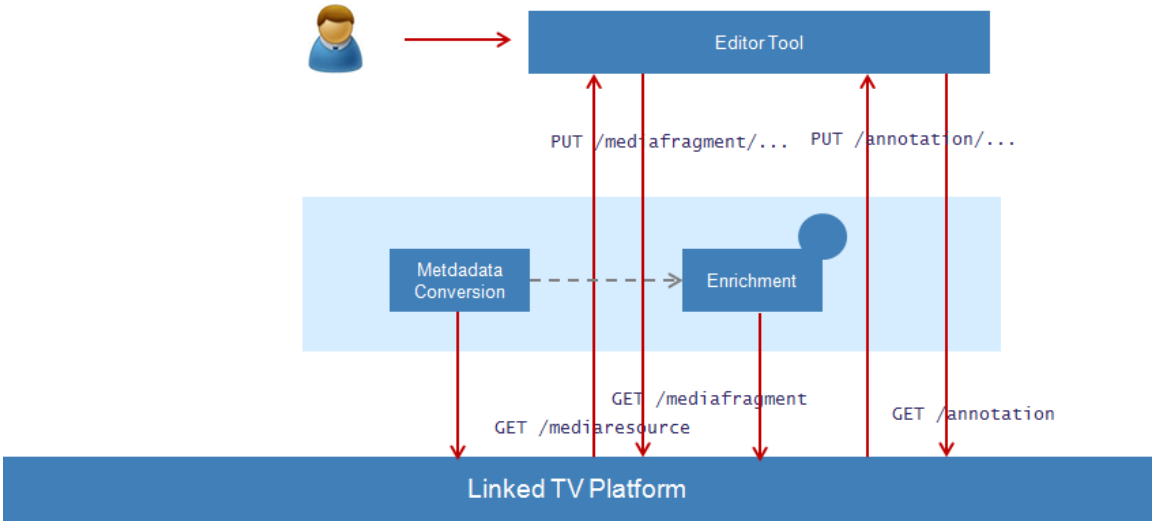

Figure 10: Integration of the Editor Tool

4 LinkedTV Platform Components

The LinkedTV Platform consists of the following components for the first end-to-end platform:

Component	Description
AdminDB	SQL Database for administrative Data
Admin REST API	REST services for accessing the administration services
Admin Tool	Web Application; the GUI for the Admin Tool
Import Agent	Importer for external video sources
Openlink Virtuoso Universal Server	the Repository for RDF Metadata
Media Server	The repository for storing and streaming media resources
Linked Media Service Bus	The core integration component, based on Apache Camel; if needed the framework

Table 7: Components of the LinkedTV Backend Platform

The components are depicted in the following image:

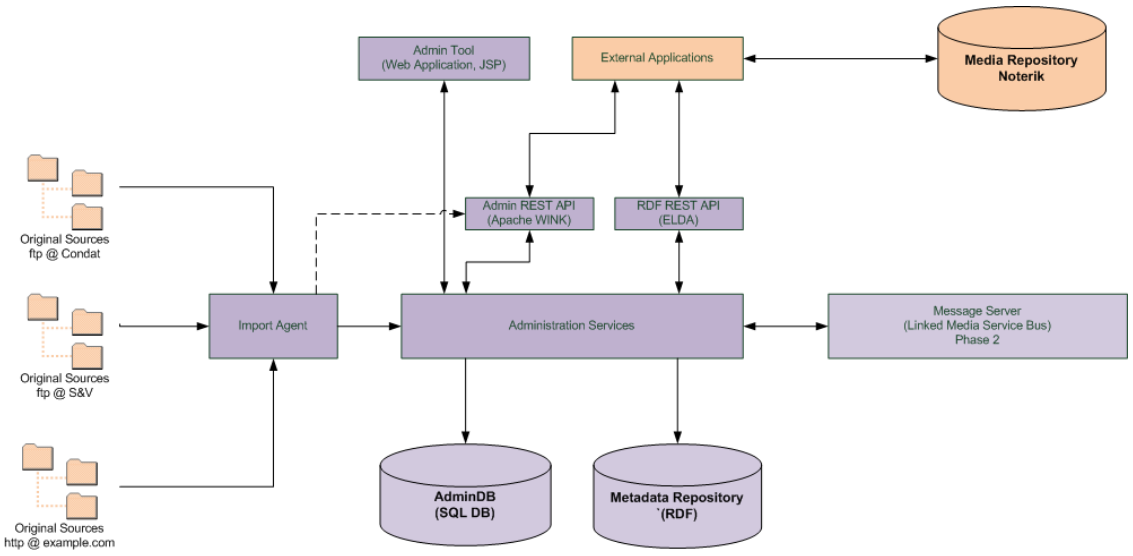

Figure 11: Administration Components

The final end-to-end platform additionally includes the LinkedTV Service Bus.

Security and Authorization

At present, access to the media resources is restricted to the Consortium members only, access to the ingested resources and the API functions for requesting complete media resources requires a login key. This holds also for the Media Player. With the final integration platform (M24) an OAuth mechanism will be provided which requires registration and demands User Agents receiving a key which must be included into the HTTP requests. By this it will be ensured that all access to the media resources is fully compliant with the terms and conditions of the content providers and can not be distributed freely.

4.1 LinkedTV Administration Database (AdminDB)

For the administration of video processing an SQL DB is better suited than an RDF triple store, because it is quite homogeneously structured information. Here SQL DB of Virtuoso Universal Server is used.

The LinkedTV AdminDB is also be updatable via REST API functions, which can be called from tools from other work packages (such as the EXMARaLDA tool).

The schema is designed to be compliant to the W3C Ontology for Media Resources 1.0[14] (MedOnt10) as much as possible, and in particular, with the class `ma:MediaResource`. However, only of a subset of the Core Elements is used, others can be added if needed.

4.1.1 Object Model

MEDIA_RESOURCE

Required fields are underlined.

Attribute	Description
Identifier	The unique id (key) of the Media Resource
TitleName	String; title of the Media Resource; if not available by default the filename (URI)
Language	URI | String
Locator	The logical address at which the resource can be accessed (e.g. a URL, or a DVB URI).

[14] http://www.w3.org/TR/mediaont-10/

Attribute	Description
DateInserted	DateTime of import into the media repository
Collection	The collection where the media resource belongs to; in our case the original ftp directory
Publisher	URI, String; eg.. “rbb”, or http://www.rbb-online.de, or “dbpedia:rbb”
FrameSizeWidth	Decimal, e,g. 720
FrameSizeHeight	Decimal, e.g. 480
Compression	URI, String; e.g. ogg, mpeg4
Format	URI, String; MIME Type
Duration	Decimal; in seconds
hasRelations	MediaResourceRelation; handles the relations to other related resources, like EXMARaLDA file, playout versions in different formats or metadata files
LinkedTVStatus	Not part of MedOnt10; used to describe the workflow status of this media resource in LinkedTV; -> LINKEDTV_STATUS

Table 8: Object Model of the AdminDB

Not yet used: *DateCreated, Contributor, Creator, Location, Description, Keyword, Genre, Rating, Copyright, Policy, TargetAudience, Fragment, NamedFragment* (these will be dealt with separately)*, SamplingRate, FrameRate, AverageBitRate, NumTracks* (this one might become important later).

MEDIA_RESOURCE_RELATION

This table stores the 1:n relational information of a media resource to other objects; these might be other files, like metadata files, other versions of the same media resource. Each Media Resource can have arbitrary many relations. At the moment, we will not cover fragment relations here; these are dealt with in the RDF repository (maybe subject to later change).

Attribute	Description
MediaResource.Identifier	The unique identifier of the Media Resource

Attribute	Description
RelationTarget	URI; Relation of this media resource to another one, e.g. the compressed file hosted by Noterik; this could also be the Identifier of another LinkedTV Media Resource
RelationType	URI, String; e.g. “720p mp4 playout” or “EXMARaLDA Result”; if the Target is a LinkedTV Media Resource, the type should be “LinkedTV Media Resource”;

Table 9: Table MEDIA_RESOURCE_RELATION

LINKEDTV_STATUS

For a list of the different defined LinkedTV status cf. Table 3 (p.6).

5 REST API

The REST API of the LinkedTV Backend Platform consists of two parts:

1. The REST API for setting and getting information on complete media resources
2. The REST API to access the RDF Triplestore

The base URL for both REST API URIs is `http://api.linkedtv.eu`

5.1 REST API for MediaResources

The following table lists the REST API functions for getting and setting complete media resources.

Operation	HTTP Method	URI	Request Body	Response Body
Get all media resources	GET	/mediaresource	None	XML of all media resources
Create a media resource	POST	/mediaresource	XML of a media resource	XML of the new created media resources
Get a media resource	GET	/mediaresource/{id}	None	XML of a media resources
Replace a media resource	PUT	/mediaresources/{id}	XML of a media resource	None
Delete a media resource	DELETE	/mediaresource/{id}	None	None
Get all relations for a media resource	GET	/mediaresource/{id}/relation	None	XML of all relations for a media resource
Create a media resource relation	POST	/mediaresource/{id}/relation	XML of a media resource relation	None

Operation	HTTP Method	URI	Request Body	Response Body
Get a media resource relation	GET	/mediaresource/{id}/relation/{id}	None	XML of a media resource relation
Replace a media resource relation	PUT	/mediaresource/{id}/relation/{id}	XML of a media resource relation	None
Delete a media resource relation	DELETE	/mediaresource/{id}/relation/{id}	None	None

Table 10: Overview of REST calls for complete media resources

In the following the REST API methods are described in detail by giving examples of request and return values.[15]

List available MediaResources

Request:

```
Method URL:
  GET http://api.linkedtv.eu/mediaresource

Header:
    Authorization: Basic YWRtaW46bGlua2VkdHY=
    Accept: application/xml

Body:
    --- Empty ---
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml

Body:
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
     <mediaResources>
        <mediaResource>
```

[15] The REST API documentation is continouosly updated on http://www.linkedtv.eu/wiki/index.php/REST_API (accessible to Consortium members only).

```
            <id>1</id>
            <titleName>Tussen Kunst</titleName>
            <language>Dutch</language>
<locator>{streamlocator}</locator>
            <dateInserted>2012-11-30T17:19:29+01:00</dateInserted>
            <collection>
            </collection>
            <publisher>S&amp;amp;V</publisher>
            <frameSizeWidth>1920</frameSizeWidth>
            <frameSizeHeight>1080</frameSizeHeight>
            <compression>
            …
```

Returns a XML/JSON collection of available MediaResources with attributes.

`Identifier, titleName, language, dateInserted, collection, publisher, frameSizeWidth, frameSizeHeight, compression, format, duration, linkedTVStatus(id, name, description)`

Curl example:

```
curl -X GET -H "Authorization: Basic YWRtaW46bGlua2VkdHY=" -H "Accept: application/xml" -H
"Content-Type: application/xml" "http://api.linkedtv.eu/mediaresource"
```

Create a MediaResource

Create a new media resource. Required: titleName, locator, publisher. The body of the response return the complete media resource with generated id linkedTVStatus with IMPORTED state and dateInserted with the current date. The id is important, you can grab it here for later use. You will need it for create media resource relations.

Request:

```
Method URL:
  POST http://api.linkedtv.eu/mediaresource

Header:
   Authorization: Basic dXNlcjpwYXNzd29yZA==
   Accept: application/xml
   Content-Type: application/xml
Body:
   <?xml version="1.0" encoding="UTF-8"?>
   <mediaResource>
     <titleName>WAS! KOMPAKT</titleName>
     <language>German</language>
<locator>.../Data/RBB/12_03_RBB/was_20120109_kompakt2_m_16_9_512x288.mp4</locator>
     <collection />
     <publisher>rbb</publisher>
     <frameSizeWidth>512</frameSizeWidth>
     <frameSizeHeight>288</frameSizeHeight>
     <compression />
     <format />
```

```
   </mediaResource>
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml

Body:
   <?xml version="1.0" encoding="UTF-8"?>
   <mediaResource>
     <id>11</id>
     <titleName>WAS! KOMPAKT</titleName>
     <language>German</language>
     <locator>.../Data/RBB/12_03_RBB/was_20120109_kompakt2_m_16_9_512x288.mp4</locator>
     <dateInserted>2012-10-12T13:00:00+02:00</dateInserted>
     <collection />
     <publisher>rbb</publisher>
     <frameSizeWidth>512</frameSizeWidth>
     <frameSizeHeight>288</frameSizeHeight>
     <compression />
     <format />
     <linkedTVStatus>
       <id>1</id>
       <name>IMPORTED</name>
       <description>Media Resource Source file is imported into AdminDB; available for further
processing</description>
     </linkedTVStatus>
   </mediaResource>
```

Get a MediaResource

Get the metadata of a MediaResource.

Request:

```
Method URL:
  GET http://api.linkedtv.eu/mediaresource/{id}

Header:
    Authorization: Basic YWRtaW46bGlua2VkdHY=
    Accept: application/xml

Body:
    --- Empty ---
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml
```

```
Body:
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <mediaResource>
        <id>1</id>
        <titleName>Tussen Kunst</titleName>
        <language>Dutch</language>
        <locator>{streamlocator}</locator>
        <dateInserted>2012-11-30T17:19:29+01:00</dateInserted>
        <collection>
        </collection>
        <publisher>S&amp;amp;V</publisher>
        <frameSizeWidth>1920</frameSizeWidth>
        <frameSizeHeight>1080</frameSizeHeight>
        <compression>
        </compression>
        <format>mp4</format>
        <duration>0</duration>
        <linkedTVStatus>
            <id>1</id>
            <name>IMPORTED</name>
            <description>Media Resource Source file is imported into AdminDB; available for
further processing</description>
        </linkedTVStatus>
    </mediaResource>
```

Update metadata of a MediaResource

With HTTP Method PUT metadata of media resources can be updated (currently not yet available).

Request:

```
Method URL:
  PUT http://api.linkedtv.eu/mediaresource/{id}

Header:
   Authorization: Basic YWRtaW46bGlua2VkdHY=
   Accept: application/xml
   Content-Type: application/xml
Body:
    <?xml version="1.0" encoding="UTF-8"?>
    <mediaResource>
      <titleName>WAS! KOMPAKT</titleName>
      <language>German</language>
      <locator>.../Data/RBB/12_03_RBB/was_20120109_kompakt2_m_16_9_512x288.mp4</locator>
      <collection />
      <publisher>rbb</publisher>
```

```
      <frameSizeWidth>512</frameSizeWidth>
      <frameSizeHeight>288</frameSizeHeight>
      <compression />
      <format />
    </mediaResource>
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml

Body:
    --- EMPTY ---
```

Delete a MediaResource

Use HTTP Method DELETE to delete a media resource. MediaResources can only be deleted if they have NO relations to media resource relations. You have first delete all relations before you can delete a media resource.

Request:

```
Method URL:
    DELETE http://api.linkedtv.eu/mediaresource/{id}

Header:
    Authorization: Basic YWRtaW46bGlua2VkdHY=
Body:
    --- EMPTY ---
```

Response:

```
Header:
    Status Code: 302 Moved Temporarily

Body:
    --- EMPTY ---
```

Get media resource relations

Get relations of a MediaResource.

Request:

```
Method URL:
    GET http://api.linkedtv.eu/mediaresource/{id}/mediaResourceRelations
```

```
Header:
    Authorization: Basic YWRtaW46bGlua2VkdHY=
    Accept: application/xml
Body:
    --- EMPTY ---
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml

Body:
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <mediaResourceRelations>
        <mediaResourceRelation>
            <id>1</id>
            <relationTarget>.../Processing/SV/12_02_SV/FhG/EXMARaLDA/12_11_07</relationTarget>
            <relationType>exmeralda</relationType>
        </mediaResourceRelation>
        <mediaResourceRelation>
            <id>2</id>
            <relationTarget>.../RDF/demo_review/S&amp;amp;V/TUSSEN_KUNST_-
AVR000080E2_Analysis.ttl</relationTarget>
            <relationType>analysisTTL</relationType>
        </mediaResourceRelation>
        <mediaResourceRelation>
            <id>3</id>
            <relationTarget>.../RDF/demo_review/S&amp;amp;V/TUSSEN_KUNST_-
AVR000080E2_subtitles_entities.ttl</relationTarget>
            <relationType>subtitleEntitiesTTL</relationType>
        </mediaResourceRelation>
    </mediaResourceRelations>
```

Create a MediaResource relation

Create a relation for a MediaResource.

Request:

```
Method URL:
  POST http://api.linkedtv.eu/mediaresource/{id}/mediaResourceRelations

Header:
   Authorization: Basic YWRtaW46bGlua2VkdHY=
   Content-Type: application/xml

Body:
    <mediaResourceRelation>
```

```
      <id>31</id>
      <relationTarget>.../sample.mpg</relationTarget>
      <relationType>LinkedTV Media Resource</relationType>
    </mediaResourceRelation>
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml

Body:
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <mediaResourceRelation>
      <id>31</id>
      <relationTarget>.../sample.mpg</relationTarget>
      <relationType>LinkedTV Media Resource</relationType>
    </mediaResourceRelation>
```

Get MediaResource relation

Get a single relation of a MediaResource.

Request:

```
Method URL:
  GET http://api.linkedtv.eu/mediaresource/1/mediaResourceRelations/{id}

Header:
   Authorization: Basic YWRtaW46bGlua2VkdHY=
   Content-Type: application/xml

Body:
    --- EMPTY ---
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml

Body:
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <mediaResourceRelation>
        <id>2</id>
        <relationTarget>.../RDF/demo_review/S&amp;amp;V/TUSSEN_KUNST_-
AVR000080E2_Analysis.ttl</relationTarget>
        <relationType>analysisTTL</relationType>
    </mediaResourceRelation>
```

Update metadata of a MediaResource relation

Update metatdata of a relation for a MediaResource.

Request:

```
Method URL:
  PUT http://api.linkedtv.eu/mediaresource/1/mediaResourceRelations/{id}

Header:
   Authorization: Basic YWRtaW46bGlua2VkdHY=
   Content-Type: application/xml

Body:
    <mediaResourceRelation>
        <id>2</id>
        <relationTarget>.../RDF/demo_review/S&amp;V/TUSSEN_KUNST_-
AVR000080E2_Analysis.ttl</relationTarget>
        <relationType>analysisTTLExtended</relationType>
    </mediaResourceRelation>
```

Response:

```
Header:
    Status Code: 200 OK
    Content-Type: application/xml

Body:
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <mediaResourceRelation>
        <id>2</id>
        <relationTarget>.../RDF/demo_review/S&amp;V/TUSSEN_KUNST_-
AVR000080E2_Analysis.ttl</relationTarget>
        <relationType>analysisTTLExtended</relationType>
    </mediaResourceRelation>
```

Delete a MediaResource relation

Use HTTP Method DELETE to delete a MediaResource relation.

Request:

```
Method URL:
DELETE http://api.linkedtv.eu/mediaresource/1/mediaResourceRelations/2

Header:
   Authorization: Basic YWRtaW46bGlua2VkdHY=

Body:
    --- EMPTY ---
```

Response:

```
Header:
    Status Code: 302 Moved Temporarily

Body:
    --- EMPTY ---
```

5.2 RDF REST API

The following table lists a preliminary collection of REST API calls to access the RDF Repository.

Operation	HTTP Method	URI	Response Body
Get all media fragments	GET	/mediafragment	XML of all media fragments
Get a media fragment	GET	/mediafragment/{id}	XML of specified media fragment
Get annotations of media fragment	GET	/mediafragment/{id}/annotation	XML of annotations from a media fragment
Get all media resources	GET	/mediaresource	XML of media resources
Get a media resource	GET	/mediaresource/{id}	XML of specified media resource
Get all annotations	GET	/annotation	XML of all annotations
Get spatial objects	GET	/spatialobject	XML of all spatial objects

The REST API for the RDF triple graph is implemented with Elda (open Linked Data api specification) http://www.epimorphics.com/web/tools/elda.html. Elda is a java implementation of the Linked DATA API http://code.google.com/p/linked-data-api/ http://linkeddata.org/. The Linked Data API provides a configurable way to access RDF data using simple RESTful URLs that are translated into queries to the SPARQL endpoint.

Get all fragments of a media resource

There are two ways to get media fragments of an media resource.

Request:

```
GET http://data.linkedtv.eu/mediaresource/{mediaresourceId}/mediafragment
or
GET http://data.linkedtv.eu/mediafragment?isFragmentOf={mediaresourceURI}
```

Response:

```
Header:
    Status Code: 200

Body:
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <result format="linked-data-api"
href="http://data.linkedtv.eu/API/mediafragment?isFragmentOf=http://data.linkedtv.eu/media/602
6703a-c02c-41bc-9ac3-9923b23ef8f5" version="0.2">
     <definition
href="http://data.linkedtv.eu/API/meta/mediafragment?isFragmentOf=http://data.linkedtv.eu/medi
a/6026703a-c02c-41bc-9ac3-9923b23ef8f5"/>
     <extendedMetadataVersion
href="http://data.linkedtv.eu/API/mediafragment?isFragmentOf=http://data.linkedtv.eu/media/602
6703a-c02c-41bc-9ac3-9923b23ef8f5&_metadata=all"/>
     <first
href="http://data.linkedtv.eu/API/mediafragment?isFragmentOf=http://data.linkedtv.eu/media/602
6703a-c02c-41bc-9ac3-9923b23ef8f5&_page=0"/>
     <hasPart
href="http://data.linkedtv.eu/API/mediafragment?isFragmentOf=http://data.linkedtv.eu/media/602
6703a-c02c-41bc-9ac3-9923b23ef8f5"/>
     <isPartOf
href="http://data.linkedtv.eu/API/mediafragment?isFragmentOf=http://data.linkedtv.eu/media/602
6703a-c02c-41bc-9ac3-9923b23ef8f5"/>
     <items>
       <item href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-
9923b23ef8f5#t=0.0,2.8">
         <isFragmentOf href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-
9923b23ef8f5"/>
         <temporalEnd datatype="xsd_float">2.799999952316284</temporalEnd>
         <temporalStart datatype="xsd_float">0</temporalStart>
         <temporalUnit>npt</temporalUnit>
       </item>
       ...
       <item href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-
9923b23ef8f5#t=20.92,22.76">
         <isFragmentOf href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-
9923b23ef8f5"/>
         <temporalEnd datatype="xsd_float">22.760000228881840</temporalEnd>
         <temporalStart datatype="xsd_float">20.920000076293950</temporalStart>
         <temporalUnit>npt</temporalUnit>
       </item>
     </items>
     <itemsPerPage datatype="long">10</itemsPerPage>
     <next
href="http://data.linkedtv.eu/API/mediafragment?isFragmentOf=http://data.linkedtv.eu/media/602
6703a-c02c-41bc-9ac3-9923b23ef8f5&_page=1"/>
```

```
    <page datatype="long">0</page>
    <startIndex datatype="long">1</startIndex>
    <type>
      <item href="http://purl.org/linked-data/api/vocab#ListEndpoint"/>
      <item href="http://purl.org/linked-data/api/vocab#Page"/>
    </type>
  </result>
```

Get all annotations of a media resource.

Request:

```
GET http://data.linkedtv.eu/mediaresource/{mediaresourceId}/annotation
```

Response:

```
Header:
    Status Code: 200


Body:
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <result format="linked-data-api" href="http://data.linkedtv.eu/API/mediaresource/6026703a-c02c-41bc-9ac3-9923b23ef8f5/annotation" version="0.2">
     <definition href="http://data.linkedtv.eu/API/meta/mediaresource/_identifier/annotation"/>
     <extendedMetadataVersion href="http://data.linkedtv.eu/API/mediaresource/6026703a-c02c-41bc-9ac3-9923b23ef8f5/annotation?_metadata=all"/>
     <first href="http://data.linkedtv.eu/API/mediaresource/6026703a-c02c-41bc-9ac3-9923b23ef8f5/annotation?_page=0"/>
     <hasPart href="http://data.linkedtv.eu/API/mediaresource/6026703a-c02c-41bc-9ac3-9923b23ef8f5/annotation"/>
     <isPartOf href="http://data.linkedtv.eu/API/mediaresource/6026703a-c02c-41bc-9ac3-9923b23ef8f5/annotation"/>
     <items>
       <item href="http://data.linkedtv.eu/annotation/1de6aa9b-6e17-4aab-8f9a-d0eb2d0b0c9e">
         <hasBody href="http://data.linkedtv.eu/entity/8c219677-47ea-4964-815d-add91320f234"/>
         <hasTarget>
           <item href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-9923b23ef8f5#t=70.04,74.6"/>
           <item href="http://data.linkedtv.eu/text/97bf1980-2cd7-46c7-9e47-be54f6b6f172#offset_135_141_Berlin"/>
         </hasTarget>
         <startedAtTime datatype="xsd_dateTime">2013-02-16T20:34:23.247Z</startedAtTime>
         <wasAttributedTo href="http://data.linkedtv.eu/organization/EURECOM"/>
         <wasDerivedFrom href="http://data.linkedtv.eu/text/97bf1980-2cd7-46c7-9e47-be54f6b6f172"/>
       </item>
       ...
       <item href="http://data.linkedtv.eu/annotation/9e2ab55b-639d-4f5d-8489-84185b242d4e">
         <hasBody href="http://data.linkedtv.eu/entity/cb0980fb-b536-4a08-a1f3-f8ae7fa8ed90"/>
         <hasTarget>
```

```
          <item href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-
9923b23ef8f5#t=204.92,208.92"/>
          <item href="http://data.linkedtv.eu/text/2a73ddfc-a769-415b-8981-
b2e062353845#offset_1087_1089_EU"/>
        </hasTarget>
        <startedAtTime datatype="xsd_dateTime">2013-02-16T20:34:23.253Z</startedAtTime>
        <wasAttributedTo href="http://data.linkedtv.eu/organization/EURECOM"/>
        <wasDerivedFrom href="http://data.linkedtv.eu/text/2a73ddfc-a769-415b-8981-
b2e062353845"/>
      </item>
    </items>
    <itemsPerPage datatype="long">10</itemsPerPage>
    <next href="http://data.linkedtv.eu/API/mediaresource/6026703a-c02c-41bc-9ac3-
9923b23ef8f5/annotation?_page=1"/>
    <page datatype="long">0</page>
    <startIndex datatype="long">1</startIndex>
    <type>
      <item href="http://purl.org/linked-data/api/vocab#ListEndpoint"/>
      <item href="http://purl.org/linked-data/api/vocab#Page"/>
    </type>
  </result>
```

Get all media fragments

Request:

```
Method/URL:
   GET/http://data.linkedtv.eu/mediafragment

Headers:
   Accept: application/xml

Body:
   --- EMPTY ---
```

Response:

```
Header:
   Status Code: 200 OK
   Content-Type: application/xml

Body:
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <result format="linked-data-api"
href="http://data.linkedtv.eu/API/mediafragment?_pageSize=3" version="0.2">
    <definition href="http://data.linkedtv.eu/API/meta/mediafragment"/>
    <extendedMetadataVersion
href="http://data.linkedtv.eu/API/mediafragment?_metadata=all&_pageSize=3"/>
    <first href="http://data.linkedtv.eu/API/mediafragment?_pageSize=3&_page=0"/>
    <hasPart href="http://data.linkedtv.eu/API/mediafragment">
      <definition href="http://data.linkedtv.eu/API/meta/mediafragment"/>
      <isPartOf href="http://data.linkedtv.eu/API/mediafragment?_pageSize=3"/>
```

```
      <type>
        <item href="http://purl.org/linked-data/api/vocab#ListEndpoint"/>
      </type>
    </hasPart>
    <items>
      <item href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-
9923b23ef8f5#t=0.0,2.8">
        <isFragmentOf href="http://data.linkedtv.eu/media/6026703a-c02c-41bc-9ac3-
9923b23ef8f5"/>
        <temporalEnd datatype="xsd_float">2.799999952316284</temporalEnd>
        <temporalStart datatype="xsd_float">0</temporalStart>
        <temporalUnit>npt</temporalUnit>
      </item>
      <item href="http://data.linkedtv.eu/media/e2899e7f-67c1-4a08-9146-
5a205f6de457#t=0.0,1.799">
        <isFragmentOf href="http://data.linkedtv.eu/media/e2899e7f-67c1-4a08-9146-
5a205f6de457"/>
        <temporalEnd datatype="xsd_float">1.799000024795532</temporalEnd>
        <temporalStart datatype="xsd_float">0</temporalStart>
        <temporalUnit>npt</temporalUnit>
      </item>
      <item href="http://data.linkedtv.eu/media/e2899e7f-67c1-4a08-9146-
5a205f6de457#t=0.0,1.96">
        <isFragmentOf href="http://data.linkedtv.eu/media/e2899e7f-67c1-4a08-9146-
5a205f6de457"/>
        <temporalEnd datatype="xsd_float">1.960000038146973</temporalEnd>
        <temporalStart datatype="xsd_float">0</temporalStart>
        <temporalUnit>npt</temporalUnit>
      </item>
    </items>
    <itemsPerPage datatype="long">3</itemsPerPage>
    <next href="http://data.linkedtv.eu/API/mediafragment?_pageSize=3&_page=1"/>
    <page datatype="long">0</page>
    <startIndex datatype="long">1</startIndex>
    <type>
      <item href="http://purl.org/linked-data/api/vocab#Page"/>
    </type>
  </result>
```


5.2.1 SPARQL Endpoint

The RDF Triplestore is based on OpenLINK Virtuoso Server Version 06.01.3127 on host *api.linkedtv.eu*. It is yet configured to allow full access for everyone, including *Upload*, *Update* and *Deletion* of Triple-Data (this is likely to be changed).

The SPARQL endpoint is available under http://data.linkedtv.eu/sparql.