Deliverable 5.6 Final LinkedTV End-to-End Platform

Jan Thomsen (CONDAT)

Version: 02.10.2014

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	*30th September 2014*
Actual date of delivery	*2nd October 2014*
Deliverable number	*D5.6*
Deliverable name	*Final End-to-End Platform*
File	*LinkedTV_D5.6.docx*
Nature	*Prototype*
Status & version	*V1.0*
Number of pages	*29*
WP contributing to the deliverable	*WP5*
Task responsible	*CONDAT*
Authors	*Jan Thomsen (Condat)*
Other contributors	*Ali Sarioglu (Condat), Daniel Oeckeloen, Pieter van Leeuwen (Noterik)*
Reviewer	*Jaroslav Kuchar (UEP)*
EC Project Officer	*Thomas Küpper*
Keywords	*Integration Platform, Architecture, ConnectedTV, Workflow, Services*
Abstract (for dissemination)	*This Deliverable describes the final LinkedTV End-to-End Platform, which integrates 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 *Final LinkedTV End-to-End Platform* as it is due M36 (September 2014). It focuses on the efforts, which have been undertaken for the finalization of the End-to-end Platform rather than a complete description of all parts and components. It builds on *D5.1 The LinkedTV Platform Architecture*, *D5.3 First LinkedTV End-to-End Platform* and *D5.4 Final LinkedTV Integration Platform,* which should be referred to for the whole picture.

This document does also not cover the technical details of the single components and services within the different workflow steps themselves (e.g. the different tools and services for video analysis or personalization). These again are subject of the respective Deliverables as results from Work Packages WP1 to WP 4. These components will only be described here as far as their interaction and integration with the platform is concerned.

With respect further development and to future releases it should be mentioned that the term *final end-to-end platform* refers to a state of the platform where all interfaces, services and supporting components are integrated and a complete end-to-end workflow is established. However, this does not mean that the platform or the integrated services will not be developed any further. Just in contrary, besides optimizations due to evaluation and experience with the productive environment, further functions will be implemented for support, monitoring, integration, administration etc. with the ultimate goal of providing a full business-ready LinkedTV PaaS (Platform as a Service), or, more specifically, Linked Media as a Service (LMaaS) Infrastructure.

1.1 Related Deliverables

D5.1	LinkedTV platform and architecture
D5.3	First LinkedTV end-to-end platform
D5.4	Final LinkedTV Integration platform

1.2 History of the document

Date	Version	Name	Comment
2014/06/02	0.1	Jan Thomsen	Document created and initial content
2014/09/19	0.2	Jan Thomsen	Most parts overworked
2014/09/25	0.3	Jan Thomsen	Final version for internal QA
2014/09/26	0.4	Jaroslav Kuchar	QA
2014/09/30	0.9	Jan Thomsen	Changes according to QA
2014/10/02	1.0	Jan Thomsen	Changes according to comments by Scientific Coordinator

Table 1: History of the document

2 The LinkedTV End-to-End Platform

2.1 The LinkedTV Ecosystem

The LinkedTV ecosystem overview (Figure 1) as has been introduced in *D5.4 Final Integration Platform* is a good means to depict the overall scope of the LinkedTV End-to-end Platform: each path going from one of these components to some other is a path supported by the End-to-Platform, the center of which is the LinkedTV Core Platform itself. At the outer boundaries this End-to-end Platform is connected to the Web (the light-colored bubbles).

The whole ecosystem is designed as an open system, which generally allows for the connection and additions of further nodes, or the replacement of existing ones. As also can be seen in the picture, the integration approach follows a cascading integration strategy, where certain nodes function as integration hubs for more specialized nodes.

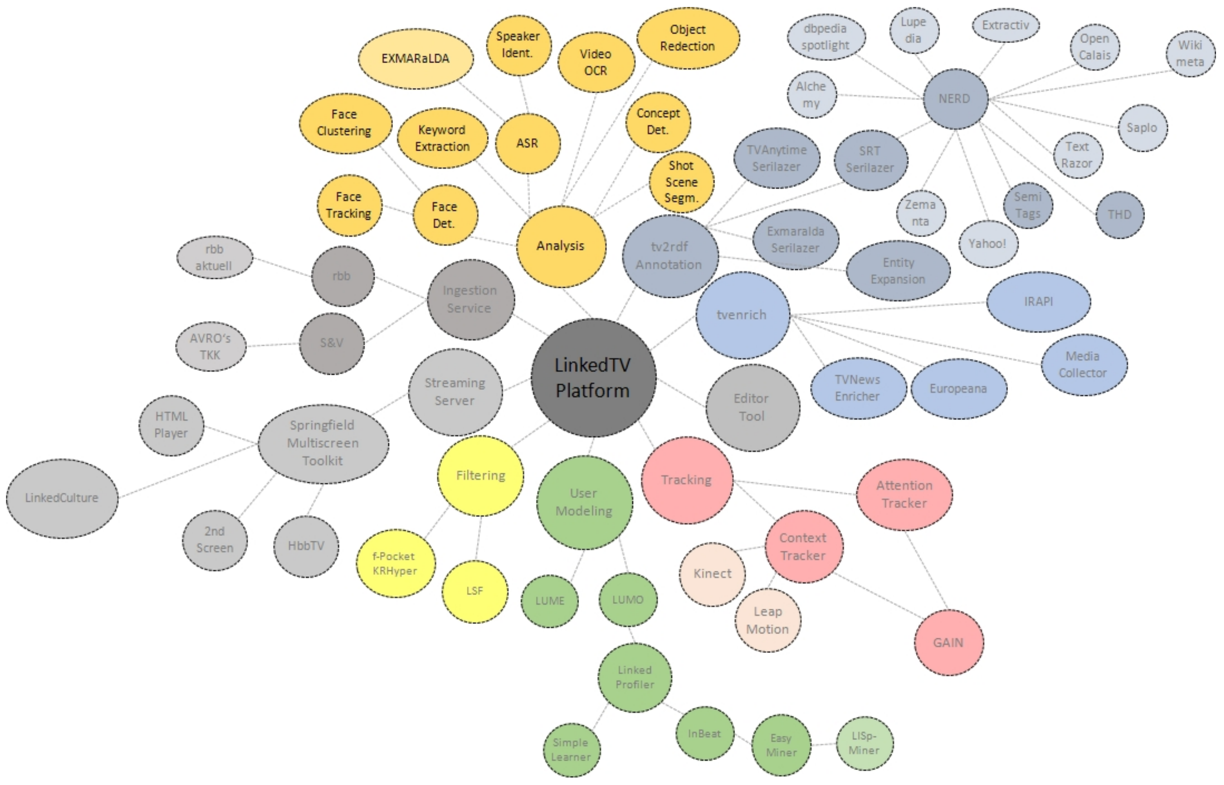


Figure 1: The LinkedTV Ecosystem

For the description of the overall LinkedTV Platform architecture, workflow and integration concepts refer to *D5.1 LinkedTV Platform Architecture* and *D5.4 Final LinkedTV Integration Platform*. In the remainder of this Deliverable we will particularly focus on the different aspects which have been developed during year 3. These aspects are:

- The LinkedTV Platform Service Layer
- The LinkedTV Platform Dashboard
- Integration of enrichment
- Integration of personalization and contextualization
- Extendibility by integration of third party components and services Guidelines for setting-up the LinkedTV Platform for customers
- Data Layer REST API extensions
- Technical aspects of deployment and production

These aspects will be covered in the next sections.

2.2 The LinkedTV End-to-end Platform Architecture

As a reminder Figure 2 shows the overall well-known LinkedTV workflow which has been described in-depth in Deliverables D5.1, D5.3 and D5.4 consisting of 3 main parts: 1) the ingestion of media resources together with available metadata files, 2) the analysis and annotation steps which creates annotated media fragments including an optional step of manual curation, and 3) the actual exploitation of these metadata in providing personalized and enriched end-user or viewer applications in broadcast and Web video.

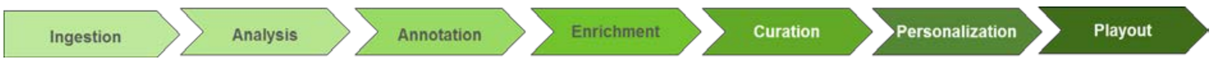


Figure 2: The LinkedTV Workflow

This subsection describes the final LinkedTV End-to-end Platform Architecture which supports and enables this workflow. Figure 3 shows the conceptual overview of the end-to-end platform: the Data Layer is part of the core platform developed and maintained by Condat and Noterik and consists of two main parts: the persistent and aggregated metadata (Condat) and the media resources played out via a media fragment streaming server. The Data exposes also a REST API under data.linkedtv.eu (see Section 2.7 for further details). The LinkedTV Service Layer contains all the distributed services for ingestion, analysis, serialization, annotation and personalization which are provided by the different partners. A dedicated interface to these services has been added which is described in detail in the next section. Above these services the actual different LinkedTV client applications are developed by consuming the services REST API for requesting actions and the data REST API for requesting data.

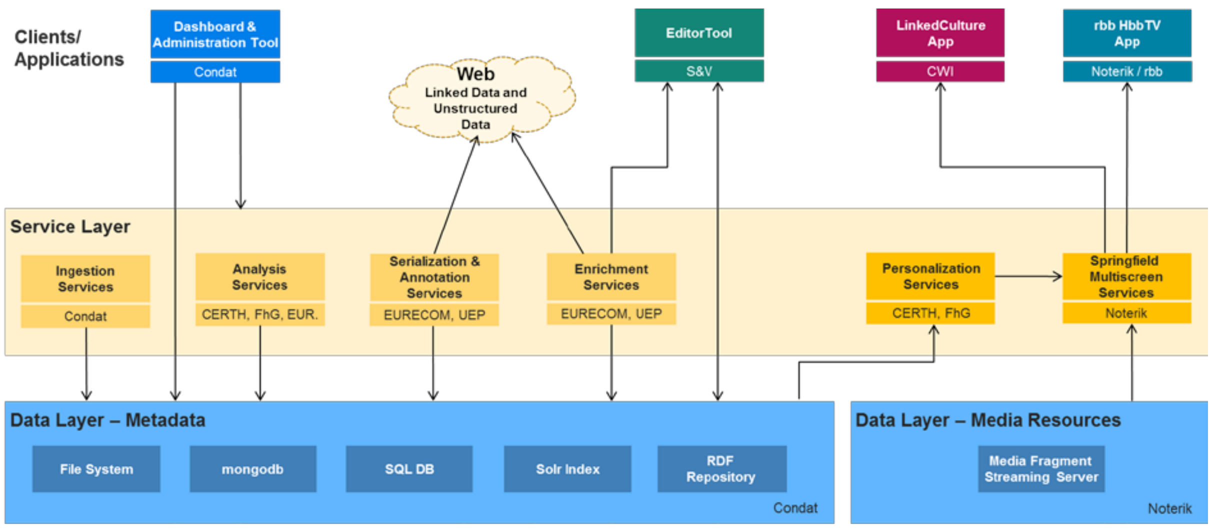


Figure 3: The LinkedTV End-to-end Platform Architectur

2.3 The LinkedTV Platform Service Layer

In order to provide a homogeneous interface to the different LinkedTV services like analysis, annotation, enrichment or personalization, the overall LinkedTV platform has been extended by an additional service layer, now resulting in a “sandwich architecture” [Figure 4].

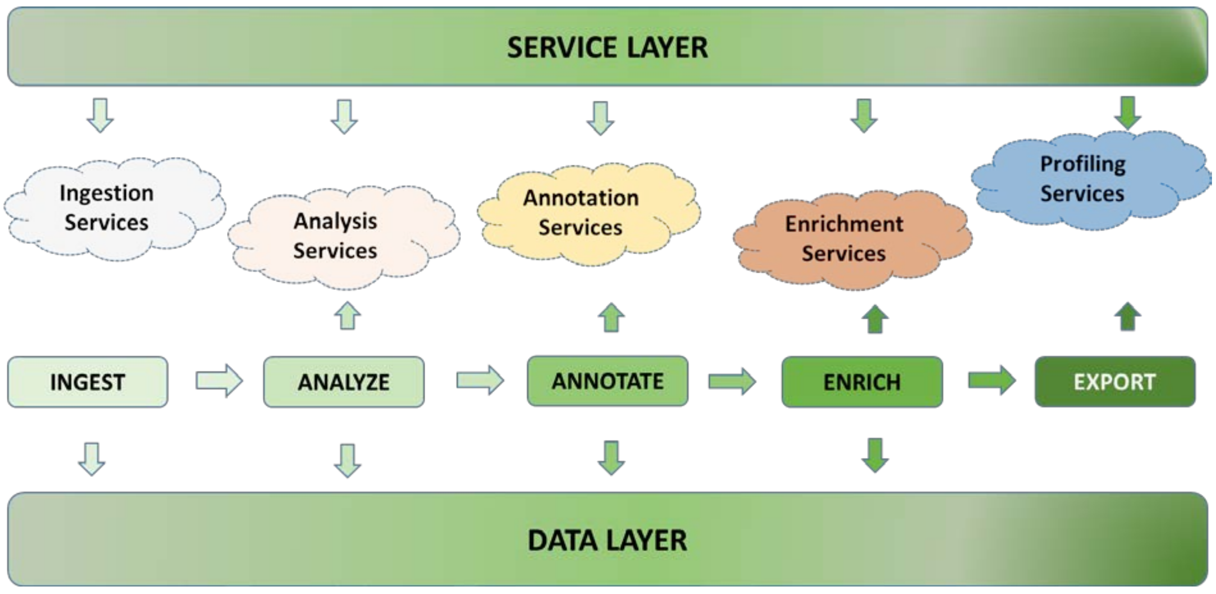


Figure 4: LinkedTV Platform Architecture with Service Layer

Additionally to the already existing Data Layer exposed under api.linkedtv.eu and data.linkedtv.eu the different LinkedTV services (e.g. Analysis Services, Annotation Services, are now being made accessible through a general proxy *services.linkedtv.eu,* depicted here as clouds because they are distributed all over different locations at the partner sites and accessible via standard REST APIs)[2]. In order to better understand the different APIs and the following overview describes their basic purposes and properties.

api.linkedtv.eu

Layer	Data Layer
Purpose	Provide access to the properties and workflow status of processed media resources, like title, related files, duration, language, etc.
General pattern	http://api.linkedtv.eu/mediaresource/{uuid}/

[2] For a detailed description of how these services are integrated into the platform architecture and workflow cf. *D5.4 Final LinkedTV Integration Platform*.

Used for	Getting and setting general properties and status updates of media resources
Used by	LinkedTV Platform Dashboard, Transfer service of Noterik, Internal Workflow Processing (Enterprise Service Bus)
Formats	HTML, JSON, XML

Table 2: api.linkedtv.eu properties

data.linkedtv.eu

Layer	Data Layer
Purpose	Provide access to the LinkedTV RDF Triple Store and thus providing the LinkedTV Linked Data Endpoint. This includes access to: - media fragments, chapters, scenes, shots and their annotations and enrichments - the LinkedTV ontologies - a SPARQL endpoint
General pattern	http://data.linkedtv.eu/{resource}/.. where {resource} can be *media resources, chapter, scenes*, etc.
Used for	Getting RDF data for media fragments and annotations
Used by	All services and clients which provide LinkedTV services and applications, e.g. Player, Enrichment, Personalization, Service Layer
Formats	HTML, JSON, TTL, RDF+XML

Table 3: data.linkedtv.eu properties

services.linkedtv.eu

Layer	Service Layer
Purpose	Provide a high-level and homogeneous access to the different LinkedTV Services from partners Prepare extension of further services and components for exploitation Facilitate monitoring, load-balancing, status checking of LinkedTV Services

General pattern	http://services.linkedtv.eu/{service}/.. where {service} can be *import, analyze, annotate, enrich, profile*
Used for	Configuring, triggering and controlling the calls to LinkedTV services
Used by	Player and LinkedCulture for personalisation, internal platform processing
Formats	JSON

Table 4: services.linkedtv.eu properties

The *service.linkedtv.eu* proxy provides a lot of advantages, in particular with respect to future exploitation and business models:

- **Export Service API**: LinkedTV thus can expose its services to develop all different kinds of LinkedTV applications instead of only calling them internally through its own workflow mechanism
- **Enable rich and flexible service contexts and transaction models**: each service invocation can be embedded into a whole context of further checks and actions, like error handling, authorization, customization, logging and monitoring, parameterization, notifications, or even fine-grained billing, licensing and service level negotiations. Some of these, like billing, are not part of the platform by now, but with this design it adding respective modules is already being prepared
- **Stable APIs**: client applications do not need to change their interfaces when underlying services change or are replaced by other services, or customers require their own services to be integrated
- **Open interfaces**: this provides also an open interface with clear LinkedTV endpoints which allows both the LinkedTV ecosystem to be integrated in various production environments as well as the easy development and integration of 3rd party components or services

The LinkedTV Service Layer exposes an interface via a REST API which is described in the following table. All calls can be parameterized with different values, which are omitted here for simplicity.

services.linkedtv.eu Endpoint	Description
`POST /import/mediaresource`	add a new media resource
`POST /import/mediaresource/:uuid/related`	add a new related resource for a media resource, like a subtitle file, a poster image or any other file

services.linkedtv.eu Endpoint	Description
`PUT /import/mediaresource/:uuid`	update an existing media resource
`GET /import/mediaresource/:uuid`	get all information about the media resource
`DELETE /import/mediaresource/:uuid`	delete a media resource
`DELETE /import/mediaresource/:uuid/related`	delete a media resource relation
`POST /analysis/mediaresource/:uuid`	request an analysis for a media resource
`DELETE /analysis/mediaresource/:uuid`	delete an analysis for a media resource
`PUT /analysis/mediaresource/:uuid`	update an analysis for a media resource
`GET /analysis/mediaresource/:uuid`	get the analysis result (Exmaralda file) for a media resource
`POST /annotation/mediaresource/:uuid`	request an annotation for a media resource based on some related file like a subtitle file, metadata file or analysis result file
`DELETE /annotation/mediaresource/:uuid`	delete annotations for a media resource
`POST /enrichment/mediaresource/:uuid`	request an enrichment for a media resource based on some related file like a subtitle file, metadata file or analysis result file
`DELETE /enrichment/mediaresource/:uuid`	delete enrichments for a media resource
`GET /profile/:userid/datapath?=:datapath`	request the profiling (personalization) of a given dataset with respect to a userid

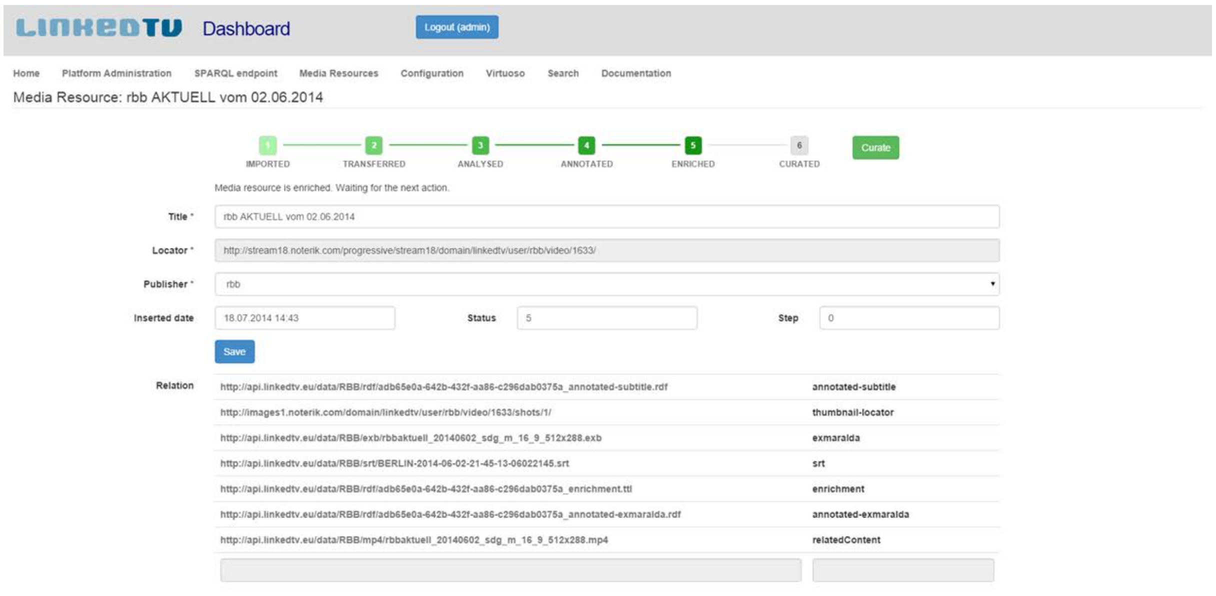
Table 5: Service Layer REST API

Technically, the LinkedTV Service Layer is being realized as a node.js[3] server application, a lightweight framework for highly scalable, data- and traffic intensive Web applications.

[3] http://nodejs.org/

2.4 The LinkedTV Platform Administration Tool and Dashboard

In D5.4 Final Integration Platform we already introduced the LinkedTV Platform Administration Tool. This Administration Tool has now been extended to an overall Dashboard which in particular allows for a manual uploading of media resources and their related files with a subsequent triggering of the different steps of the LinkedTV workflow.



Media Resource: rbb AKTUELL vom 02.06.2014

Media resource is enriched. Waiting for the next action.

Title *	rbb AKTUELL vom 02.06.2014
Locator *	http://stream18.noterik.com/progressive/stream18/domain/linkedtv/user/rbb/video/1633/
Publisher *	rbb
Inserted date	18.07.2014 14:43
Status	5
Step	0

Relation		
	http://api.linkedtv.eu/data/RBB/rdf/adb65e0a-642b-432f-aa86-c296dab0375a_annotated-subtitle.rdf	annotated-subtitle
	http://images1.noterik.com/domain/linkedtv/user/rbb/video/1633/shots/1/	thumbnail-locator
	http://api.linkedtv.eu/data/RBB/exb/rbbaktuell_20140602_sdg_m_16_9_512x288.exb	exmaralda
	http://api.linkedtv.eu/data/RBB/srt/BERLIN-2014-06-02-21-45-13-06022145.srt	srt
	http://api.linkedtv.eu/data/RBB/rdf/adb65e0a-642b-432f-aa86-c296dab0375a_enrichment.ttl	enrichment
	http://api.linkedtv.eu/data/RBB/rdf/adb65e0a-642b-432f-aa86-c296dab0375a_annotated-exmaralda.rdf	annotated-exmaralda
	http://api.linkedtv.eu/data/RBB/mp4/rbbaktuell_20140602_sdg_m_16_9_512x288.mp4	relatedContent

Figure 5: The LinkedTV Platform Dashboard

As can be seen in Figure 5 all information for a particular media resource such as *title*, *locator*, *related files* and especially the *status* of the workflow steps can be seen in one place. Some data, such as title can be changed by an editor, while other data, in particular the locator are fixed and could only be changed by integrated services. Whenever a certain status is reached the next step can be triggered manually by clicking the button. It has to be noted that some steps, especially the analysis step, can take quite long depending on which exact single methods are being executed. Also, executing a single step always means that different distributed services are involved in the background (cf. the LinkedTV Ecosystem graphics for an illustration).

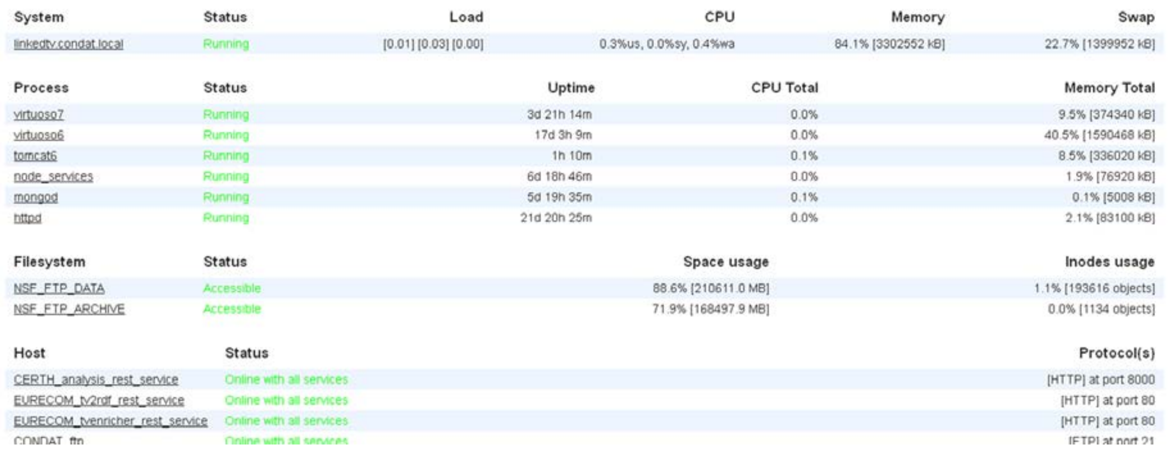
Although the Dashboard is now complete with respect to the end-to-end workflow by supporting the whole process from media resource ingestion up to enrichment and preparation for curation (Release 1.0 of the LinkedTV Dashboard) it will be developed further according to user evaluation results and demands or just for optimization. The planning for future releases includes:

- Giving the user control over which subprocesses will be executed. Currently for each type of program, such as "rbb aktuell" or "Tussen Kunst & Kitsch" fix templates have been predefined which specify which exact analysis methods have to be performed (e.g. shot segmentation, chapter segmentation, keyword extraction), some of which might be quite expensive in terms of time. Users should have to be able to change this default behavior by checking or unchecking single analysis methods.

- Giving users feedback about task completion. Currently users cannot see when a particular step is expected to be finished. Because each step at the first level includes calling different services distributed at different sites it is actually quite difficult to give real completion feedback as this would require completion feedback by each single subprocess. But a feedback based on heuristic estimations should be at least included.
- Giving the user the possibility to reset and redo different steps. Currently videos can be processed via the Dashboard only in a straight forward manner. However, it would useful to be able to discard e.g. analysis or enrichment results and redo them.
- Add statistics about how many videos have been processed, how many media fragments of which type have been detected, etc.
- Further improvements will surely come up during the Platform evaluation process from M37 to M42.

2.4.1 Monitoring Platform services

For the monitoring of all services which are directly or indirectly connected with the Platform the Administration Tool has been extended with a monitoring component monitoring service based on monit[4]



System	Status	Load	CPU	Memory	Swap
linkedtv.condat.local	Running	[0.01] [0.03] [0.00]	0.3%us, 0.0%sy, 0.4%wa	84.1% [3302552 kB]	22.7% [1399952 kB]

Process	Status	Uptime	CPU Total	Memory Total
virtuoso7	Running	3d 21h 14m	0.0%	9.5% [374340 kB]
virtuoso6	Running	17d 3h 9m	0.0%	40.5% [1590468 kB]
tomcat6	Running	1h 10m	0.1%	8.5% [336020 kB]
node_services	Running	6d 18h 46m	0.0%	1.9% [76920 kB]
mongod	Running	5d 19h 35m	0.1%	0.1% [5008 kB]
httpd	Running	21d 20h 25m	0.0%	2.1% [83100 kB]

Filesystem	Status	Space usage	Inodes usage
NSF_FTP_DATA	Accessible	88.6% [210611.0 MB]	1.1% [193616 objects]
NSF_FTP_ARCHIVE	Accessible	71.9% [168497.9 MB]	0.0% [1134 objects]

Host	Status	Protocol(s)
CERTH_analysis_rest_service	Online with all services	[HTTP] at port 8000
EURECOM_tv2rdf_rest_service	Online with all services	[HTTP] at port 80
EURECOM_tvenricher_rest_service	Online with all services	[HTTP] at port 80
CONDAT_ftp	Online with all services	[FTP] at port 21

Figure 6: Monitoring LinkedTV services and processes

Processes, which are running locally at the platform are also restarted automatically when the monitoring detects a failure. As services of the partners are all REST services they can be monitored through simple calls to an HTTP GET endpoint which returns a 200 status code when up and running or a 500 or 404 when not. Currently partners are only notified manually if problems exist, in future releases however an automatic forwarding to the respective partner contact emails could be implemented,

[4] http://mmonit.com/monit/

2.5 Integration of enrichment

The enrichment step within the LinkedTV workflow chain has been also fully integrated now.[5] The precondition for triggering the enrichment process is that the media resource is in state `ANNOTATED`, i.e. the results of the serialization and entity recognition process are stored as media fragment annotations within the RDF repository. Technically, also the enrichments are just further annotations, internally to be identified by the property `oa.motivatedBy=Linking`[6].

The enrichment process is triggered by calling the service named *TVEnricher*. The TVEnricher service is hosted by LinkedTV partner EURECOM under http://linkedtv.eurecom.fr/tvenricher. It serves as a single service end point but in itself covers different subservices which are not called by the Platform directly (following the cascading integration approach as described in *D5.4 Final Integration Platform*).

Although the necessary precondition for the enrichment process is the state `ANNOTATED` of the respective media resource, reaching that state is not the only event on which the enrichment process can be called. In general, we can distinguish the following three enrichment strategies:

1. `onAnnotated`: the enrichment is triggered directly when the media resource is in state `ANNOTATED`
2. `onScheduledUpdate`: the enrichment is triggered by scheduling it, e.g. on a daily or weekly basis; since enrichment results change very quickly over time, a scheduled update mechanism ensures that the preprocessed enrichments include the most recent ones. The *onScheduledUpdate* strategy should include two sub-strategies *add* and *replace*, where *add* just adds new enrichments and replace always discards previous ones and only stores the most recent ones.
3. `onDemand`: the enrichment process is triggered on demand by a LinkedTV client application directly at playout time, or shortly before playout time with caching and optional further processing, e.g. for personalization or categorization. This of course ensures that the enrichments consist of the most up-to-date results. Within the *onDemand* strategy it does of course not make sense to store the results in the repository.

Currently, the Platform itself supports the *onAnnotated* strategy and the *onScheduledUpdate_Add* strategy; the *onScheduledUpdate_Replace* strategy and the *onDemand* strategy is planned to be supported by the Platform in future releases.

Additionally, further enrichments can also be requested on an *onDemand* strategy by other clients directly from the TVEnricher service, without involvement of the Platform. This is most

[5] See *D2.6 Advanced concept labelling by complementary Web mining* for description of details of the enrichment process.

[6] For a thorough description of the nature of enrichments see Deliverable D2.4.

notably the case with the LinkedTV EditorTool[7], but also are client applications like the LinkedCulture application use this feature. In case of the EditorTool, enrichments are requested and then curated and stored back to the Platform Data Layer Repository when confirmed. These are then called *curated enrichments* and are distinguishable by a special provenance attribute `prov:wasAttributedTo <http://data.linkedtv.eu/organization/SV/EditorTool>`.

Integration into the LinkedTV Service Interface

The LinkedTV Service Interface for the enrichment service looks as follows

Property	**Description**
URI	`http://services.linkedtv.eu/enrichment/mediaresource/:uuid`

Checks (Preconditions)	`.../mediaresource/:uuid EXISTS`
	`.../mediaresource/:uuid/state >= ANNOTATED`

Calls	`http://linkedtv.eurecom.fr/tvenricher` (service specific parameters apply)

Parameters	`store=none|add|replace`
	`action=start|getResult|delete`

Actions	`store = add`: store enrichment results in the repository If `store = replace`: delete previous enrichments and add new ones If `store = none`: directly return results Add URL for editortool.linkedtv.eu/:publisher/:uuid Send notification to clients (in future releases)
Postcondition	`../mediaresource/:uuid/state = ENRICHED` (if CURATED this state remains)

Table 6: Properties of the integration of enrichment

[7] http://editortool.linkedtv.eu

Getting enrichment results: Integration into the LinkedTV Data Interface

To get the enrichment results the Data Layer REST API has been extended with the following end points:

data.linkedtv.eu Endpoint	**Description**
`GET /mediaresource/:uuid/enrichment`	get all enrichment annotations of the media resource
`GET /mediaresource/:uuid/enrichment/automatic`	get only automatically generated enrichments of the media resource
`GET /mediaresource/:uuid/enrichment/curated`	get only curated enrichments of the media resource
`GET /chapter/:uuid/enrichment`	get all enrichments of a chapter
`GET /chapter/:uuid/enrichment/automatic`	get automatically generated enrichments of a chapter
`GET /chapter/:uuid/enrichment`	get curated enrichments of a chapter

Table 7: Data REST API endpoints for enrichment results

Further endpoints will be provided if needed.

2.6 Integration of personalization and contextualization

The integration of the personalization and contextualization workflow was the final part which was missing for the end-to-end workflow which has now been integrated in the last three months. The personalization and contextualization workflow consists of three main parts:

1. Tracking user events with the UMONS AttentionTracker and the GAIN service from UEP[8]
2. Building a user model using this implicitly gained data as well as through explicit user profiling via FhG LUME[9]
3. Profiling of displayed content in the LinkedTV Player or any other end-user application via the Recommenders by FhG[10] and CERTH[11]

For an in-depth description of all involved personalization and contextualization components and services see *D4.6 Contextualisation solution and implementation*. In the remainder of this subsection the integration of these services into the LinkedTV Platform workflow is described as far as the Platform is concerned. The tracking of user events and storing the data in GAIN has been integrated into the LinkedTV Player – Springfield Multiscreen Toolkit.[12] The Player itself gets information about annotations to be displayed from the Platform and sends to GAIN all related information about these annotations, apart from this there is no Platform interaction involved for user tracking. Building a user model is happening completely outside the Platform, so there is no Platform interaction involved.

The main part of Platform interaction is occurring in the profiling part. The interaction workflow has been realized by means of the Service Layer API as described in Section 2.3 . It involves the 4 components PLAYER, SERVICES, DATA and PROFILER. The LinkedTV Service Layer provides the exact same interface

```
http://services.linkedtv.eu/profile/:userid/datapath?=:datapath
```

for personalized and non-personalized data. In case of non-personalized data the Player (or any other client) just needs to send a special userid :NOUSERID. Figure 7 depicts this profiling workflow without personalization. In this case the Profiler isn't called at all but the Service Layer Personalization Controller just calls the Data Layer API to get the requested media fragments and annotations and returns them directly. These requested annotations

[8] https://wa.vse.cz/gain/docs/

[9] http://linkedtv1.iais.fraunhofer.de:8888/personal-recommender/#/um/11

[10] http://linkedtv1.iais.fraunhofer.de:8888/personal-recommender/

[11] http://multimedia.iti.gr/api/

[12] See also D4.6 as well as D5.5 LinkedTV front-end: video player and MediaCanvas API v2

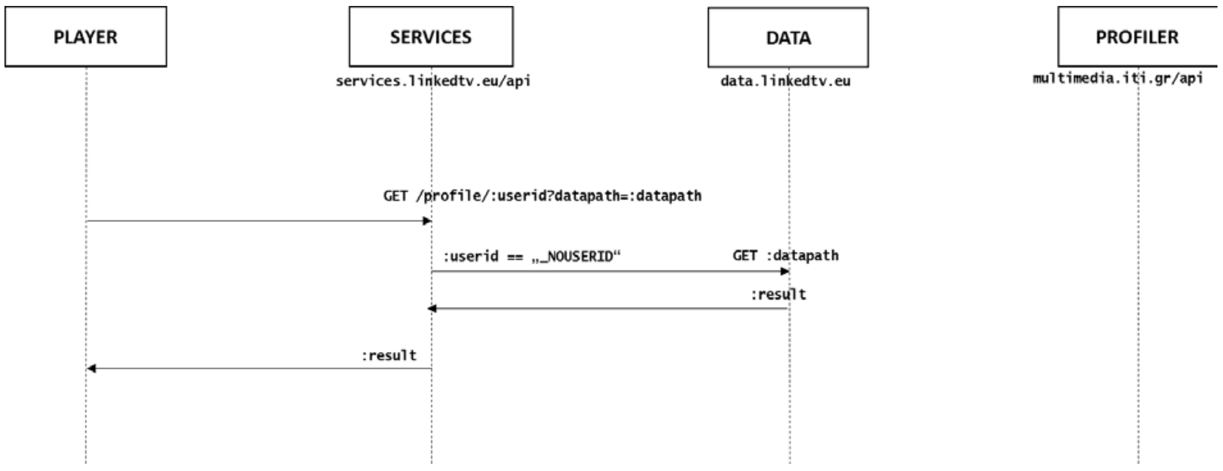


Figure 7: Profiling workflow without personalization

and media fragments are given in the :`datapath` parameter and can contain any call to theData Layer like `/mediaresource/:uuid/annotation/curated` etc.[13] The Service Layer REST API is agnostic about what actual information is requested to be personalized. It is completely up to the client application to decide *what* has to be personalized (e.g. chapters, curated enrichments, automatic annotations) and *when* it has to be personalized (i.e. during play time, in a pre-buffered mode directly before play time or in advance). This also of course heavily depends on how much information has to be personalized.

The extended profiling workflow *including* personalization is depicted in Figure 8. In this case the Service Layer Personalization Controller calls the Profiler directly and passes it the `datapath` parameter. The Profiler in turn now requests the data (called the “candidates”) from the Data Layer, applies it profiling algorithms (which also may include calls to other services) and returns the results to the Service Layer which in turn passes them again to the Player, or client application, respectively. The client application then decides how to display the results. Thus the Platform itself is in fact user agnostic: it doesn’t store any information about users, user events or user models but just routes user ids between different services, i.e. the Platform does not directly retrieve or use any user profile at any time in the LinkedTV workflow.

[13] The Data Layer REST API is described in Section 2.7

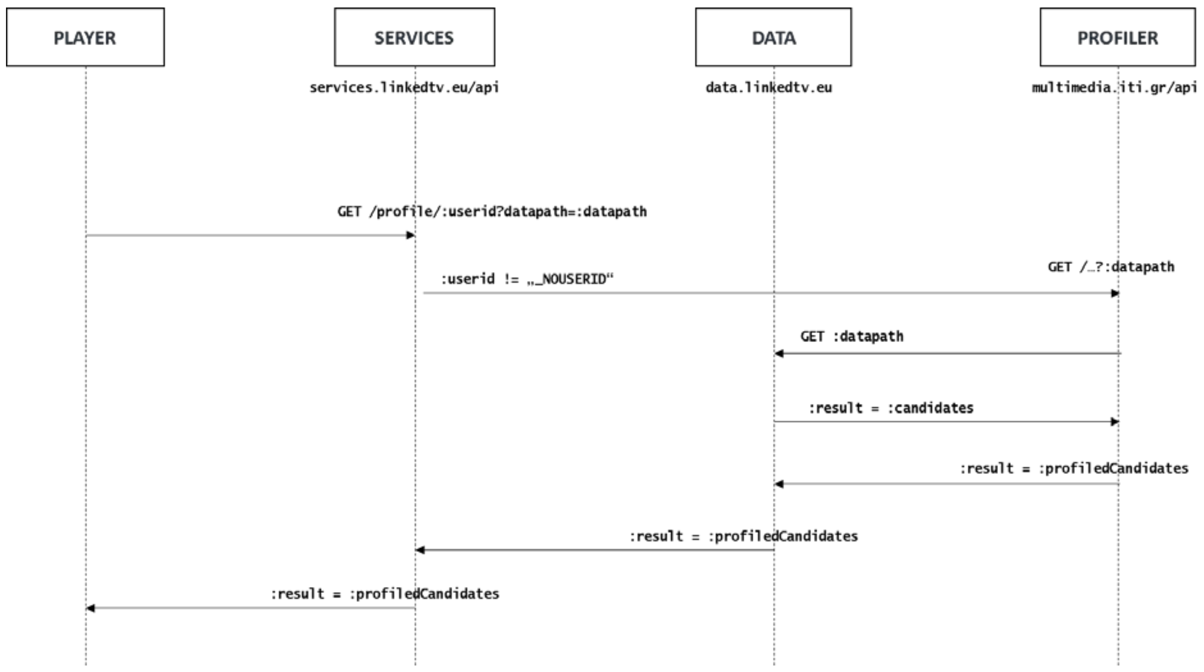


Figure 8: Profiling workflow with personalization

2.7 Data Layer REST API

The LinkedTV Data Layer includes all persistent LinkedTV data, most notably the RDF Repository, but as well an SQL Database for non-RDF data, a Solr-Index and a document-oriented NoSQL Database (MongoDB) currently only for internal usage. The Data Layer is exposed via the LinkedTV Data Interface under data.linkedtv.eu (RDF data) and api.linkedtv.eu (SQL) data. The Data Interface is a LDA[14] compliant REST API realized with the ELDA Framework.[15] Future releases will also ensure that the LinkedTV Data Layer Interface is fully compliant to the W3C Linked Data Platform 1.0 Standard[16] which is currently in a working draft status.

The Data Layer REST API has been extended with a lot more endpoint definitions to allow for easier direct access to data stored in the Data Layer. The following sheet gives an overview of the current state if of the Data Layer REST API. The meaning of the different calls should be self-explanatory.

data.linkedtv.eu Endpoint

`/mediaresource`
`/mediaresource/{id}`
`/mediaresource/{id}/mediafragment`
`/mediaresource/{id}/chapter`
`/mediaresource/{id}/shot`
`/mediaresource/{id}/annotation`
`/mediaresource/{id}/enrichment`
`/mediaresource/{id}/enrichment/automatic`
`/mediaresource/{id}/enrichment/curated`
`/mediaresource/{id}/entity`
`/mediafragment/{id}/enrichment/curated`
`/mediaresource/{id}/chapter/automatic`
`/mediaresource/{id}/chapter/curated`
`/mediaresource/{id}/scene`
`/chapter/{id}/spatialobject`
`/mediafragment`
`/mediafragment/{id}`
`/mediafragment/{id}/annotation`
`/mediafragment/{id}/enrichment`
`/mediafragment/{id}/enrichment/curated/chapter/{id}`
`/chapter/{id}/scene`
`/chapter/{id}/shot`
`/chapter/{id}/annotation`
`/chapter/{id}/enrichment`
`/chapter/{id}/entity`

[14] LDA: Linked Data API, https://code.google.com/p/linked-data-api/wiki/Specification

[15] https://github.com/epimorphics/elda

[16] http://www.w3.org/TR/ldp/

```
/shot
/shot/{id}
/shot/{id}/annotation
/shot/{id}/enrichment
/shot/{id}/entity
/annotation
/annotation/{id}
/entity
/entity/{id}
/entity/{id}/enrichment
/text
/keyword
```

Table 8: Data REST API endpoints

2.8 Extendibility by integration of third party components and services

The LinkedTV Platform is designed to be open for extensions by third parties which might be open source as well as commercial extensions. In the following we present an approach for a standardization of the third party LinkedTV extensions.

The idea is that custom components can be developed for each service which extends the respective general component class.

Class	Existing components	Examples for additional components
Importers	rbb.rbbAktuellImporter AVRO.TKKImporter SubtitleImporter TVAnytimeImporter ExmaraldaImporter	Importers for specific programmes arte.MetropolisImporter VimeoImporter
Analyzers	GermanASRAnalyzer EnglishASRAnalyzer KeywordExtractionAnalyzer SubtitleAnalyzer FaceDetectionAnalyzer ShotSegmentationAnalyzer SceneSegmentationAnalyzer ChapterSegmentationAnalyzer ConceptRedectionAnalyzer	CarDetectionAnalyzer AudioScriptAnalyzer FrenchASRAnalyzer
Serializers	SRTSerializer TVAnytimeSerializer ExmaraldaSerializer	WebVTTSerializer
Annotators	NERDAnnotator	special annotators which are not covered by NERD
Enrichers	MediaCollectorEnricher UnstructuredSearchEnricher	PersonalizedAdEnricher TwitterEnricher FacebookEnricher
AttentionTrackers	KinectAttentionTracker	LeapMotionAttentionTracker
Profilers	ImplicitProfiler ExplicitProfiler	
Exporters		WebVTTExporter

Table 9: LinkedTV Service Components

Note that these components are not components of the Core LinkedTV Platform itself but rather services of the middle layer as depicted in the «sandwich architecture» of Figure 4. They have to fulfill certain requirements to be able to be registered in a general LinkedTV Service Component Catalogue and integrated with the LinkedTV Platform Service Layer. Required attributes are:

- Provide service component properties like *URI*, *provenance information*, *release version*, *ondemand / persistent*, *contact*, *support of update/delete operations* , *3rd level services used* etc.
- If the component includes storage of RDF triples into the triple store, then also data.linkedtv.eu endpoints have to be included together with LDA specifications for GET, PUT, POST, DELETE operations
- The additional service components can then also be exposed in the Service Layer, e.g. like
  `services.linkedtv.eu`
  `/annotation/mediaresource/:uuid?annotator=personalizedAdAnnotator`
  `&{params}`
  or
  `/export/mediaresource/:uuid ?exporter=webVTTExporter&{params}`
- A specific testing process of new components or services has to ensure that they comply with the overall workflow and do not lead to security or performance issues on the Platform. Component providers must also sign an agreement on use of the data they get from the Platform which has also to be agreed with the content providers

3 Technical Aspects of Deployment and Production

Although the main goal of the LinkedTV Platform was not to provide a real business-ready product by end of the LinkedTV project but rather an integration and workflow platform for all research-oriented LinkedTV services, the LinkedTV Platform is of course intended to be exploited and further developed after the end of the project by Condat and the consortium partners. For a discussion of the market situation, business models and strategies for exploitation see *D8.6 Market and product survey for LinkedTV services and technology* and *D8.7 LinkedTV business models, v2*. Within this section we will discuss the technical and practical aspect of a possible exploitation of the LinkedTV Platform.

3.1 Making the LinkedTV Platform production-ready

The current state of the implementation and run-model of the LinkedTV Platform is that of a prototype and demonstration platform enabling an overall assessment of the validity of the LinkedTV concepts. This assessment will be done during the final 6 months of the project and the results will be reported in *D5.7 Validation of the LinkedTV architecture*. While the overall functional design, components, APIs, services etc. are expected to be the same (with changes according to the results of the evaluation and validation), there are some clear aspects which need to be addressed for a business and production-ready version of the LinkedTV Platform after the end of the project. These are mainly the following:

- *Scalability, performance and reliability*: in order to be able to handle large amounts of video data and user or client requests the Platform as well as the connected services have to be scaled up to be able to run on dozens of servers in parallel both for handling the number of requests in real-time as well as for load-balancing and fail-over strategies. This basically means to apply principles of Big Data processing to LinkedTV, addressing the classical Big Data characteristics of *volume*, *variety*, *velocity*, *variability* and *complexity*.[17] For the LinkedTV Platform this would mean to add the possibility to create clusters within the Service Layer and the Data Layer, including a clustered triple store installation, and to add a dedicated management layer for distributed coordination e.g. based on Apache Zookeeper.[18]
- *Adding business and customization functionalities:* in order to be able to serve different customers other features have to be included, e.g.
    - simple possibility to adapt the look and feel to the respective CI guide lines
    - reliable usage statistics for each service of the core platform as well as the connected services with a component for supporting licensing, billing and royalty fee models

[17] http://en.wikipedia.org/wiki/Big_data

[18] http://zookeeper.apache.org/

- configuration models to handle service level agreements; different customers normally don't need all services, e.g. they don't might need video analysis or personalization; or they need only specific video analysis techniques but e.g. not automatic chapter segmentation because this can be achieved via related TV-Anytime or other metadata; by this also different service levels should be supported, e.g. depending on the available time frame between the availability of the video and the broadcast airing time. The LinkedTV Service Layer is designed to already support the integration of such components into the handling of service calls.
- integration into local production workflows: an integration into a local production workflow means to provide custom connectors to directly import and export metadata from and to local systems like production planning systems, programme planning systems, EPG systems, archives, CMS or other. This does also effect the connected services, as e.g. additional metadata files (e.g. audio scripts) could be processed to further improve results. See Section 2.8 for a discussion of the extendibility of the platform with custom components and services.

3.2 Deployment and Hosting Models

Currently the LinkedTV Platform is hosted locally at Condat and running in a demonstration mode. For an actual high-traffic production mode hosting at Condat would not be feasible as Condat is not acting as a hoster per se. Three deployment or hosting models are possible:

1. Local installation and integration at the customer. This model is in particular suitable when an integration and adaption into the local production workflow is required. In this case the Platform administration would be done by the IT department of the customer itself.
2. Installation at and hosting by a specific classical hoster like Host Europe, Strato, etc.. In this case the Platform administration could be done by either the customer or Condat. This model is more suitable if the LinkedTV Platform is more used as an external service with single service endpoints. In this case for each customer a single and separate installation would be maintained, which can be distributed over as much virtual server instances as needed.
3. Hosting in a cloud service like Amazon Web Services or the Google Cloud Platform. In this case the LinkedTV Platform would be provided as a cloud service, or *Platform as a Service* (PaaS). Also in principle this could be the same model as in (2) here the idea is that the LinkedTV Platform is running on a per-service basis, handling different user requests simultaneously and on a highly scalable basis.

These deployment and hosting models are only valid for the LinkedTV Core Platform; for the connected services which are hosted by the different partners different models may apply, which is covered in *D8.7 LinkedTV business models, v2.*

3.3 Setting up the LinkedTV Platform for customers

In this section we describe how the LinkedTV would have to be set up and customized for specific customers such as broadcasters or archives.

In general, the LinkedTV Platform will not be an out-of-the-box or a set-up-and-run-in-minutes product because there are so many different services involved and so many different parameters need to be tailored to the specific usage scenarios. So the set-up procedure for the LinkedTV Platform will in general follow a *Plan-Install-Configure-Integrate-Test-Run* process, which is described in the following:

Plan

During the planning process all specific requirements have to be clarified. This includes the answering of questions like: how fast have the annotation results to be provided? what are the white lists for this programme? which metadata files are provided? which analysis techniques are needed, like ASR, chapter segmentation, etc,? which annotations and enrichments are needed? do the annotations have to be curated? is personalization needed? are there any custom components which have to be developed or integrated? Which hosting model is appropriate? which triple store will be used?

The planning process will be normally done during one or more workshops.

Install/Set-Up

If hosting model (1) or hosting model (2) have been selected, after the planning the installation and initial setting up of the LinkedTV Platform instance will be done. In case of hosting model (3) the next step will directly be *configure/build*. The installation will be performed as much as possible through installation scripts, but may also include the installation or even compilation of individual components like that of a Virtuoso Triple Store or other RDF repository.

Configure/Build

The configure/build step includes activities like: parameterization of the calls to the LinkedTV services in the Service Layer, setting up file system and directories, setting up process and service monitoring, configuring the workflow chain endpoints for getting or connecting video sources and metadata files, etc. In case new connectors or components have to be implemented this will also been done during this step, which then also might involve an adaptation of connected analysis or annotation services.

Integrate / Test

The integrate step includes the actual integration of the core platform services with all local services as well as the connected LinkedTV services. This also includes LinkedTV clients which might be a specific player, a Content Management System, an EPG system, a video portal or any other client. The different components and services need then to be tested individually as well as within the whole end-to-end workflow and also with respect to performance, scalability and reliability.

Run

The specific run or production mode depends on which of the above deployment or hosting models has been selected. It might be maintained by the customer itself in case of a local installation or by Condat in case of a hosted or cloud-based instance.