Deliverable 6.1.1

Project Title	Next-Generation Hybrid Broadcast Broadband
Project Acronym	HBB-NEXT
Call Identifier	FP7-ICT-2011-7
Starting Date	01.10.2011
End Date	31.03.2014
Contract no.	287848
Deliverable no.	6.1.1
Deliverable Name	Initial Version of the HBB-NEXT System Architecture
Work package	6
Nature	Report
Dissemination	Public
Author	Eugen Mikoczy (ST), Michael Probst (IRT)
Contributors	Stefan Heller (TARA), Sachin Agarwal, Felix Gomez Marmol, Gines Dolera (NEC), Ray van Brandenburg (TNO), Gregor Rozinaj, Pavol Podhradsky, Ivan Kotuliak (STUBA), Sebastian Schumann (ST), Janina Renz, Mark Gülbahar (IRT)
Due Date	31.06.12
Actual Delivery Date	11.07.12

Table of Contents

1. Executive Summary

This document presents the initial version of the HBB-NEXT system architecture as well as the architecture of the service enablers modules delivered from WP3, WP4 and WP5.

Section 2 explains the background and the relationship to other HBB-NEXT activities and deliverables and also demonstrates the role of the system architecture. Furthermore, it highlights the technical needs and benefits of a standard based on open technologies that HBB-NEXT plans to follow.

Section 3 provides guidance how the architectural elements are documented and which methodologies and tools were used.

In section 4 the first set of technical requirements is listed which have been derived from the functional requirements defined in D2.2 [5]. These minimal mandatory requirements are split into several groups such as system, service, terminal and security.

The high level architecture is presented in section 5 providing an overview about the services and features of an HBB-NEXT system as well as the domain model and scenarios of service delivery. Additionally, a summary of the high-level view on service enablers/modules and their interaction and high level procedures is presented.

In section 6 the detailed logical and physical architecture is described as well as the design of selected modules.

This document is concluded by section 7 which draws a conclusion from this version of the document and describes the plans for the next version for next milestones defined for the series of the follow up deliverables in WP6.

2. Introduction

Within this document the HBB-NEXT system architecture is described in several phases of designing of system architecture and HBB-NEXT framework development. The initial version of document D6.1.1 focuses on a first version of the architecture where a high level architecture and description of service enablers/module for initial prototypes is provided (which some of them will be part of the initial system setup). In next versions (D6.1.2 and D 6.1.3) of this document we will provide more information on how to integrate the different modules as well as how to develop application on top of framework application.

As mentioned in the project proposal HBB-NEXT intends to develop an application framework that can provide defined service enablers over modular distributed and open architecture.

The need for defining well-described interfaces to achieve interoperability has been identified. Interfaces can be divided into external and internal ones. Especially external one interface towards application developer and second one towards end terminal are essential to design open system. Such a well specified and open interfaces then enable to external parties like application developer, 3rd party service providers to easier develop HBB applications that any device with standardized browser can access them.

Several architectures and technologies used these days with hybrid broadcast broadband applications have been analysed. In the past, there have been several activities that try to focus only on standardising the end device and the browser environment like Multimedia Home Platform (MHP) [32] and OpenTV [33]. Last few years since introduction of HbbTV 1.0 [1] showed the importance of an open development environment and well specified rules how application can be developed, as this could better help to industry and developers to accept and adapt these HBB standards.

It is therefore the goal of HBB-NEXT to work towards a standard based open solution. Though in the industry proprietary solutions are still used in parallel especially by over the top players for specific services to achieve global interoperability standardization is important.

Using open and standardized technologies can extend potential interoperability (terminal to platform, application to platform/terminal) making it more relevant and attractive for the market. With a standardized solution it is also expected to have on one hand a more cost effective implementation as development cost can be shared by multiple deployment and on the other hand be more attractive for developers because of a wider potential market (more terminals/users).

The HbbTV 1.0 standard is based on CE-HTML [2] but there is a trend to introduce HTML5 [3] in the near future, most probably also for the next release of HbbTV [1]. Generally, standard web technologies are also used in connected devices and multi-screen platforms.

There are several potential deployment models of HBB services, some of them described in D.2.4 [4], depending if hybrid services are delivered by over the top players purely via internet, Telco operators delivered all services over managed networks, or HBB providers that combine broadcast with broadband delivery (see Figure 1).

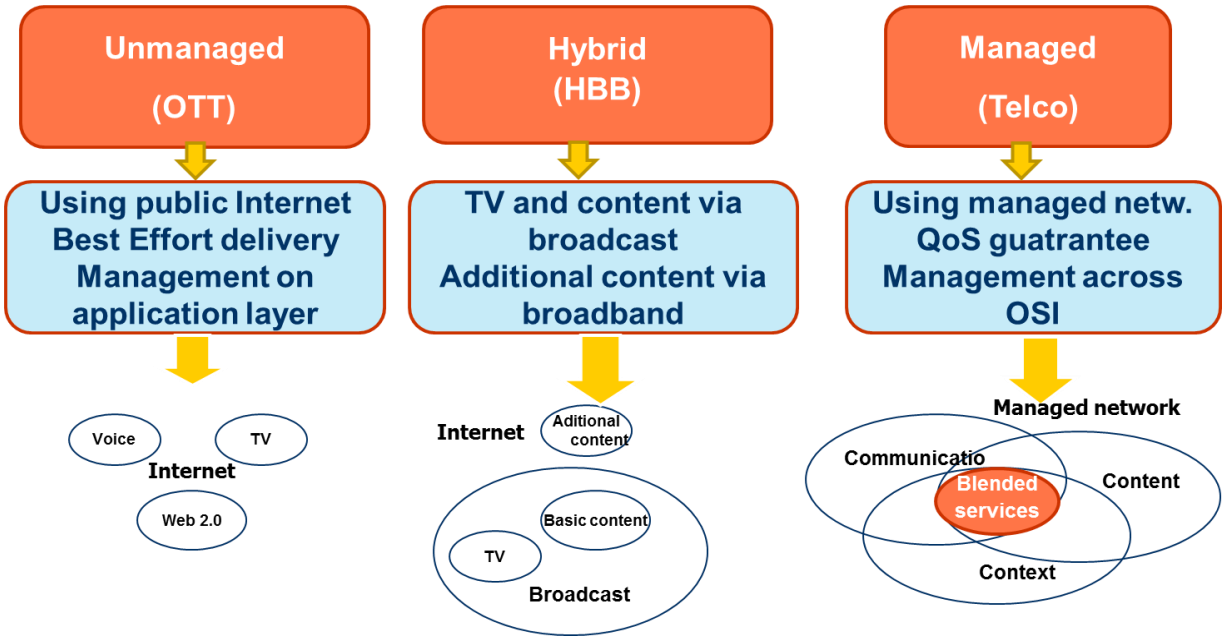

Figure 1: Differences in deployment models

The HBB-NEXT architecture will be designed independently of the deployment models so that HBB-NEXT platforms and service enablers can be used as a framework or selected modules individually, depending on the specific needs of application and service providers. Therefore, HBB-NEXT plans to provide well-defined APIs to application developers or 3rd party service providers.

The architecture design follows the service oriented architecture (SOA) principles with interoperable service enabler modules that can work independently from other modules. This approach gives great flexibility for deployment of just those modules that are necessary for particular applications or HBB-NEXT provider (see Figure 2).

The term “HBB-NEXT provider” includes broadcaster, content, service and application provider as well. More details about the domain model and deployment scenarios are described in section 0.

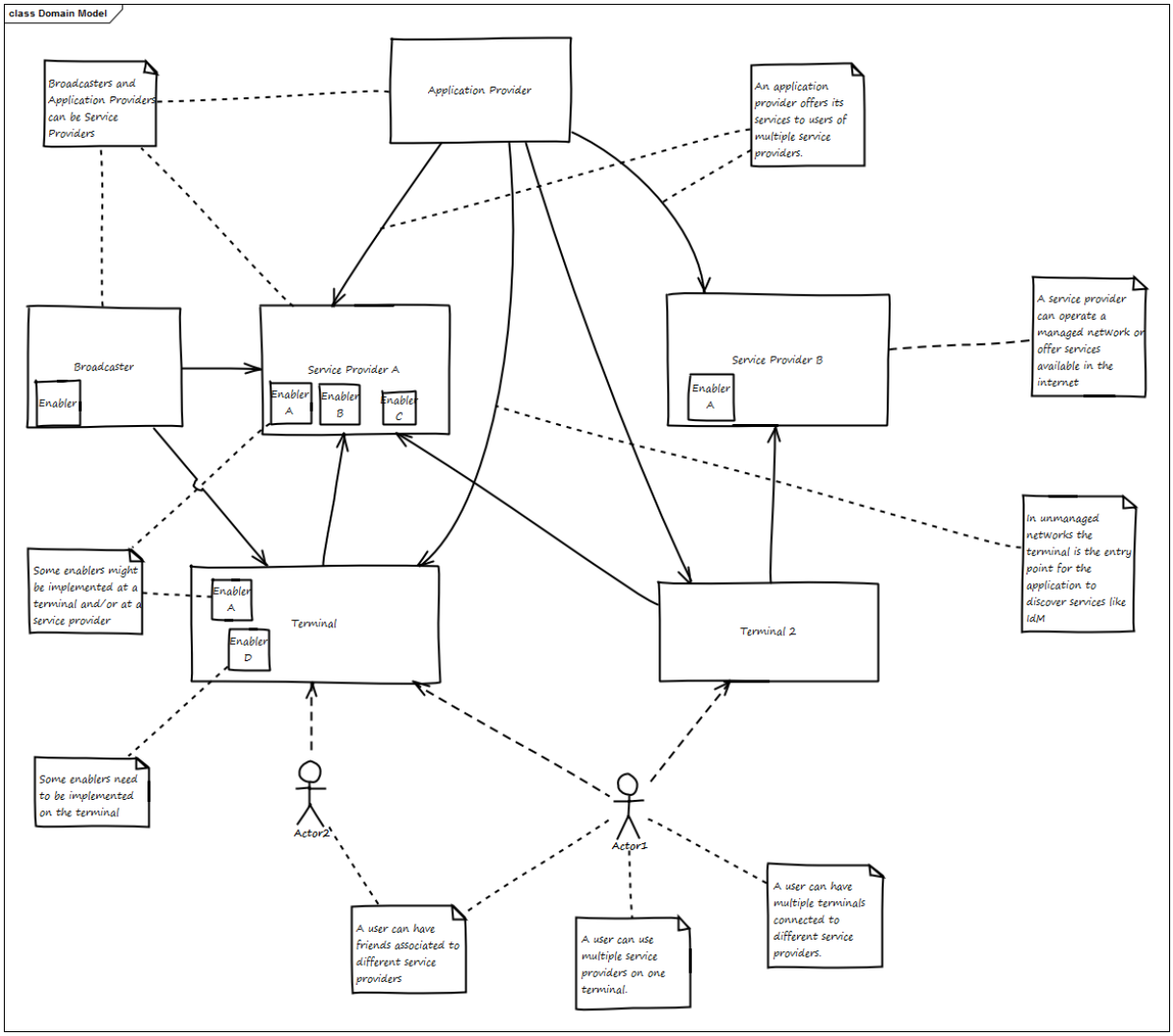

Figure 2: Flexible deployment model based on HBB-NEXT framework

This layered and open architecture for HBB applications can be understood as an enabling platform (see Figure 3) for next generation HBB applications and new application scenarios.

Instead of focusing on internal interfaces HBB-NEXT makes use of standardized APIs for application developers. Terminal interoperability can be achieved by using same standard based technologies like those specified in the upcoming HTML5 [3] specification, a future HbbTV 2.0 specification and the ETSI MCD Converged Multi-screen Service specification.

HBB-NEXT will contribute APIs and requirements needed additionally to standardization bodies within the HBB-NEXT WP7. Further information about planned standardization contributions can be found in HBB-NEXT D7.2 [6].

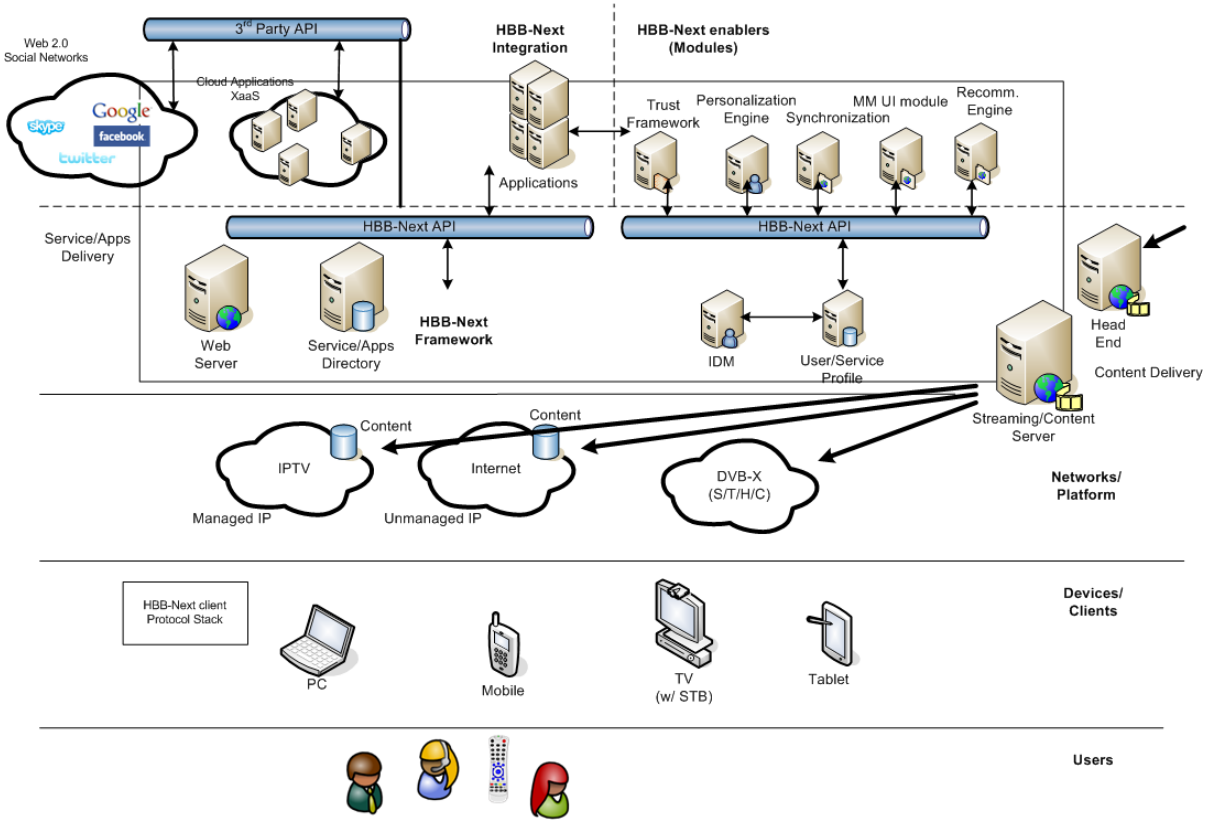

Figure 3: Principle architecture and role of APIs

3. Architecture Principles and Methodology

In this chapter the approach of defining the high level and detailed architecture for this document is described. The second part of this chapter introduces the methodology used for generating the documentation.

3.1. Architecture Design Principles

The architecture work of WP6 is based on prior work in WP2 on scenarios, use cases and functional requirements and first input on the detailed architecture for enablers fromWP3, WP4 and WP5.

WP6 asked WP3, WP4, and WP5 to provide for each of their enablers a set of technical requirements and a set of (software) modules including external interfaces. The technical requirements shall complement the functional requirements defined in deliverable D2.2 [5] and can be found in chapter 4.

Based on the input from the other WPs an integrated architecture of the HBB-NEXT framework has been developed. This includes high level software components and their external interfaces, i.e. APIs and protocols. Chapter 5 provides a high level view of the architecture while chapter 6.3 describes the components in more details.

Chapter 5.1 introduces a domain model showing how the enablers of the HBB-NEXT framework can be distributed to different service provider domains thus enabling the business models described in HBB-NEXT deliverable D2.4 [4].

3.2. Documentation Methodology and Tools

3.2.1. UML and Enterprise Architect

For documenting the system architecture HBB-NEXT has chosen the Unified Modelling Language (UML). UML [23] provides a set of tools (i.e. diagram types) for presenting different views on a system. The documentation of the HBB-NEXT system architecture uses UML diagrams where it is appropriate.

Component or class diagrams are used to illustrate the relation between the software modules and the provided and used external interfaces. Sequence diagrams show how the interfaces of modules are used and how the modules interact with others in a timely manner, i.e. the diagram provides a sequence of API calls an application is usually doing. HBB-NEXT decided to use a common tool (Enterprise Architect) for defining the UML models of the system architecture. The tool allows the sharing of all models on a central repository. The diagrams shown in this document are exported from there.

3.2.2. RESTful API Definition and Documentation

HBB-NEXT agreed to define APIs of web services which are used by applications as so-called RESTful APIs. REST [22] is a concept without providing a unified documentation tool like SOAP XML. Therefore this chapter provides a set of guidelines and conventions.

Note: When a web browser is used to run applications using a RESTful API as proposed by this chapter, it requires that the browser and the web server implement cross origin resource sharing (CORS) [24], that the browser implements HTML 5 web messaging [25], HTML 5 web sockets [26], or a proxy is installed at the application server to allow API calls though the same origin policy.

Usage of HTTP Methods

HTTP Method	Usage
GET	To retrieve information of an object or a resource provided by the web service. The use of this method should not change the state of any resource of the web service.
POST	To create a new object or resource available through the web service.
PUT	To update an object or resource of the web service.
DELETE	To delete a resource.

Resources

An HTTP resource is a hierarchical structure defined by a path name, i.e. levels separated by the slash character, for example /level1/level2. In principle the construction of the resource name depends on the module though following restrictions should apply:

- The first level should identify the module/component, e.g. "/RecEng".
- The second level may identify the version of the API of the module, e.g. "/RecEng/2.0". The web service should provide the latest version of the API with a level called "latest".

Content Format

A web service should offer responses in the JSON (java script object notation) format.

4. Technical Requirements

This chapter presents the first set of technical requirements for specifying the HBB-NEXT prototype. At this early stage they are referring to the planned prototype applications for each enabler. Those applications will be independent of each other and adaptation might be necessary when building the one integrated prototype at a later stage. Therefore, it is very likely that technical requirements have to be revised and extended in upcoming versions of this document.

A structured approach has been carried out for generating this first set of technical requirements:

- At first, WP3, WP4 and WP5 were asked to provide first technical requirements which are relevant for their planned prototype applications.
- After collecting the requirements they were grouped and consolidated in one spread sheet.
- A first cross-check between the technical requirements and the mandatory functional/system/user requirements of D2.2 [5] was carried out for identifying missing technical requirements, particularly for non-enabler related ones.
- One review/feedback loop was carried out.
- The mandatory (shall) requirements for the current stage were copied from the spread sheet into this document.

The technical requirements are grouped by the main building blocks of the current HBB-NEXT system architecture. These major architectural categories are:

- 01: System and Network
- 02: Services and Applications
- 03: Terminal Devices

The description of a requirement must contain one of the following terms to define the prioritization of the requirement: “shall”, “should”, “may”. The definition of these terms has been adopted from IETF RFC 2119 [7].

The formatting of the technical requirements basically follows the same principles as used for the functional/system/user requirements of D2.2 [5]. In this case each requirement is of type TR (= Technical Requirement) and attributed to one of the architectural categories.

The following numbering scheme will be used: TR.[CATEGORY].[REQ NUMBER]

In the next version of this document it is planned to assign a unique number to each technical requirement.

4.1. System and Network

Description	Related Enabler	WP
The server executing the engines (e.g recommendation engine and Group Context Server) shall be able to run Java / REST applications.	Not enabler related	-
A service SHALL be available in the network for managing user profiles.	IDM	3, 5

4.2. Services and Applications

Description	Related Enabler	WP
An HBB-NEXT application's user interface SHALL use clear language, comprehensible to a broad audience, e.g. not using “AES”, or “256-bit”.	Not enabler related	-
The app-store SHALL have enough computation resources to execute the trust and reputation model in use within a reasonable time-frame.	Application Trust	3
The app-store SHALL have enough resources to store and provide HBB applications to the end-users.	Not enabler related	-
The app-store SHALL be accessible from the public Internet, which means that it should be connected via IP to the public Internet.	Application Trust	3
The cloud SHALL provide a method to decode an incoming MPEG transport stream into raw video (at least SDTV resolution) and raw audio.	Cloud offloading	4
The cloud SHALL provide a method to reduce the resolution of a higher resolution video to a lower resolution video (at least SDTV resolution) and reduce the audio stream bitrate if needed.	Cloud offloading	4

Description	Related Enabler	WP
The cloud SHALL provide a method to encode raw video into WebM-encoded video (at least SDTV resolution) and encode raw audio into Ogg-encoded audio.	Cloud offloading	4
The cloud SHALL provide a method to multiplex multiple elementary streams into at least one standard container format e.g. WebM (at least 1 video and 1 audio stream).	Cloud offloading	4
The cloud SHALL provide a method to mix two raw video streams in order to achieve a picture-in-picture overlay effect.	Cloud offloading	4
The cloud SHALL provide a method to start more worker machines for media processing, based on processing demand.	Cloud offloading	4
The cloud SHALL provide a method to stop one or more worker machines for media processing based on processing demand.	Cloud offloading	4
The cloud SHALL provide an interface to receive a MPEG-TS stream (at least via HTTP interface).	Cloud offloading	4
The cloud SHALL provide an interface to transmit a HTTP media stream suitable for playback in a web browser (at least on a PC and a mobile device/tablet).	Cloud offloading	4
The cloud shall provide a programming interface to provision more worker machines when required. The decision of when to start/stop worker machines will be automatic.	Cloud offloading	4
The cloud shall provide the ability to detect client web browser capabilities (e.g native webm support) of JavaScript-enabled browsers.	Cloud offloading	4
The cloud SHALL provide the ability to store cloud-offloading applications (At least gStreamer pipelines) in a database in the cloud.	Cloud offloading	4
The cloud SHALL provide the ability to use browser capability information to make decisions of how to assist the HBB-Next application via processing/storage offloading capabilities in cloud(at least 1 application example).	Cloud offloading	4
The cloud SHALL provide the ability to overlay an image (at least in SVG format) over a raw video supported in the cloud.	Cloud offloading	4
The cloud SHALL provide the ability to monitor the applications running in the cloud and report resource usage via a web interface.	Cloud offloading	4
The cloud SHALL be able to ascertain the resource requirement of a media process up to a confidence level based on its description.	Cloud offloading	4

Description	Related Enabler	WP
The cloud SHALL be able to start media processes (At least gStreamer pipelines) on the underlying worker machine infrastructure on demand.	Cloud offloading	4
The cloud shall be able to stop media processes (At least gStreamer pipelines) on the underlying worker machine infrastructure.	Cloud offloading	4
The cloud SHALL be able to expose media process output via standard interfaces.	Cloud offloading	4
The cloud SHALL be able to import media from the processes via standard interfaces.	Cloud offloading	4
The security manager SHALL have an interface for policy enforcement settings (interface for administration).	Security Management	3
The security manager SHALL have persistent relation to Identity Management.	Security Management	3
The security manager SHALL be resistant to DoS attack.	Security Management	3
The security manager SHALL be able to enforce different roles of a user towards a service.	Security Management	3
The profile manager SHALL have a database for user profiles.	IDM	3
Identity Management SHALL have database for user data, user devices and relations.	IDM	3
The identity management SHALL be able to subscribe to changes in a context via a notification framework.	IDM	3
The identity management SHALL provide an interface to list active users on a device.	IDM	3
The identity management SHALL provide an interface to modify users in a certain context.	IDM	3
The identity management SHALL provide an interface that provides a subset of users that are relevant for multi-modal interface interaction, i.e., a subset of user identifiers (e.g. faces) that are usually connecting in the context.	IDM	3
The identity management SHALL provide an interface to retrieve all users, who are active in a certain context, i.e., who are relevant for context-based operations (e.g. multi-modal recognition, personalization).	IDM	3
The identity management SHALL provide an interface to set users as 'active' in a certain context.	IDM	3

Description	Related Enabler	WP
The identity management SHALL provide an interface to retrieve full or partial information that is relevant for identifying the user using the multi-modal interface (e.g. multi-modal vectors).	IDM	3
The profile management SHALL provide an interface by which the user's profile can be managed (retrieved, modified, extended and deleted).	IDM	3
The profile management SHALL provide an interface by which the user's profile parameters can be retrieved or set.	IDM	3
The identity management SHALL provide a method that returns a parameter from the identity profile.	IDM	3
The identity management SHALL provide an interface to manage users and the devices they use to access the framework or applications.	IDM	3
The identity management SHALL provide an interface to manage different roles of a user towards a service.	IDM	3
The identity management SHALL provide an interface to manage relations between users, between devices, and between users and devices.	IDM	3
The identity management SHALL provide an interface to manage the relation between a user and a service.	IDM	3
The personalization engine SHALL provide an interface to read, write, and manage the profile of a group.	Personalization Engine	5

4.3. Terminal Devices

Description	Related Enabler	WP
A terminal device SHALL be compliant to HbbTV V1.0 [1]	Not enabler related	-
It SHALL be possible to query the capabilities of a terminal device via the network.	Not enabler related	-
The terminal device SHALL provide a user interface for controlling the connection to other devices.	Not enabler related	-
A terminal device SHALL register itself at an 'identity server' and frequently refresh the status showing its availability in the network.		6
The terminal SHALL have a microphone array.	Multi-Modal User Ident., Multi-User Pers. Engine	3, 5

Description	Related Enabler	WP
The terminal SHALL have a camera.	Multi-Modal User Ident., Multi-User Pers. Engine	3, 5
The terminal SHALL have sufficient power for image and voice processing within multimodal interface.	Multi-Modal User Ident., Multi-User Pers. Engine	3, 5
The client device (STB/PC) shall have a way to extract PCR values from an MPEG-TS stream.	A/V Sync	4, 6
The client device (STB/PC) shall have a method to extract per-frame PTS/DTS values from an MPEG-TS stream.	A/V Sync	4, 6
The client device (STB/PC) shall be able to access private data sections in an MPEG-TS stream.	A/V Sync	4, 6
The client-device (STB/PC) shall be able to support decoding and playout of at least 2 simultaneous video streams.	A/V Sync	4, 6
The client-device (STB/PC) shall be able to receive, demultiplex and decode a MPEG-TS stream received over the broadcast interface.	A/V Sync	4, 6
The client-device (STB/PC) shall be able to receive and/or request an MPEG DASH or HTTP Live Streaming (HLS) stream.	A/V Sync	4, 6
The client-device (STB/PC) shall be able to communicate with an external server via HTTP.	A/V Sync	4, 6
The client-device (STB/PC) shall be able to control playout timing accurately (frame-accurate).	A/V Sync	4, 6
The second-screen device (tablet/smartphone) shall be able to receive and/or request and MPEG DASH or HTTP Live Streaming (HLS) stream.	A/V Sync	4, 6
The second-screen device (tablet/smartphone) shall be able to control playout timing accurately (frame-accurate).	A/V Sync	4, 6
The second-screen device (tablet/smartphone) shall be able to communicate with an external server via HTTP.	A/V Sync	4, 6
The client-device (STB/PC) shall contain a (configurable) NTP client.	A/V Sync	4, 6
The client-device (STB/PC) shall be able to synchronously playout media streams obtained through different technologies (e.g. a DVB broadcast stream and an MPEG DASH broadband stream)	A/V Sync	4, 6
The client-device (STB/PC) and second-screen device (tablet/smartphone) shall be able to synchronize their playout.	A/V Sync	4, 6

5. HBB-NEXT High Level Architecture

This chapter explains the HBB-NEXT system architecture from a high level view. The next section describes how the service scenarios and use cases are mapped to the enablers of the HBB-NEXT framework. In section 0 a generic domain model is introduced for showing the relation of HBB-NEXT business models to the system architecture.

Figure 4 gives an overview of the HBB-NEXT high level system architecture. The core of the architecture is the set of enablers which build the HBB-NEXT Framework. The HBB-NEXT applications make use of the framework to provide new type of services to the consumer over broadcast and broadband networks. The range of consumer devices consists of traditional TV sets and popular personal devices like Tablet PCs and Smart Phones.

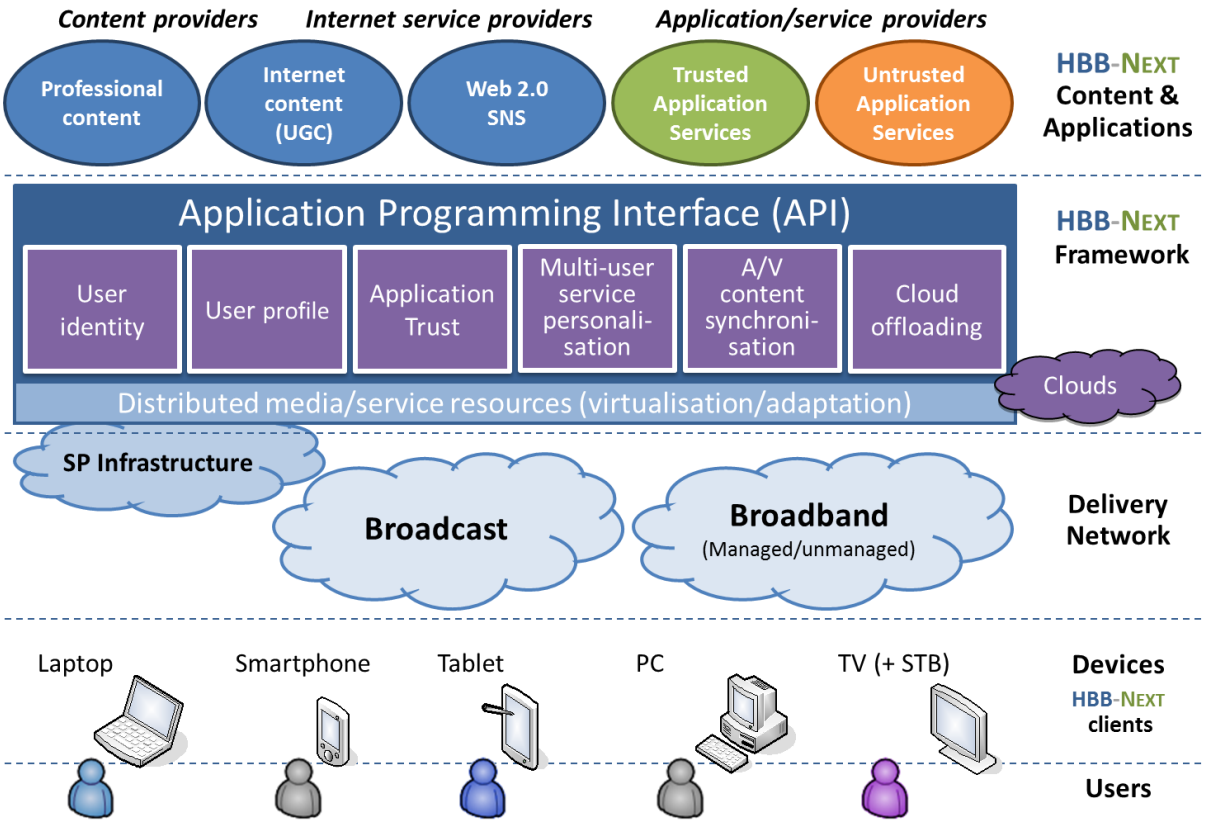

Figure 4: HBB-NEXT high level architecture

5.1. Domain Model

The domain model in this chapter is a generalization from the business models described in D2.4 [4]. They show the main business roles and the services which they offer and consume. The domain models shall help to understand the impact of the different business models on the HBB-NEXT architecture.

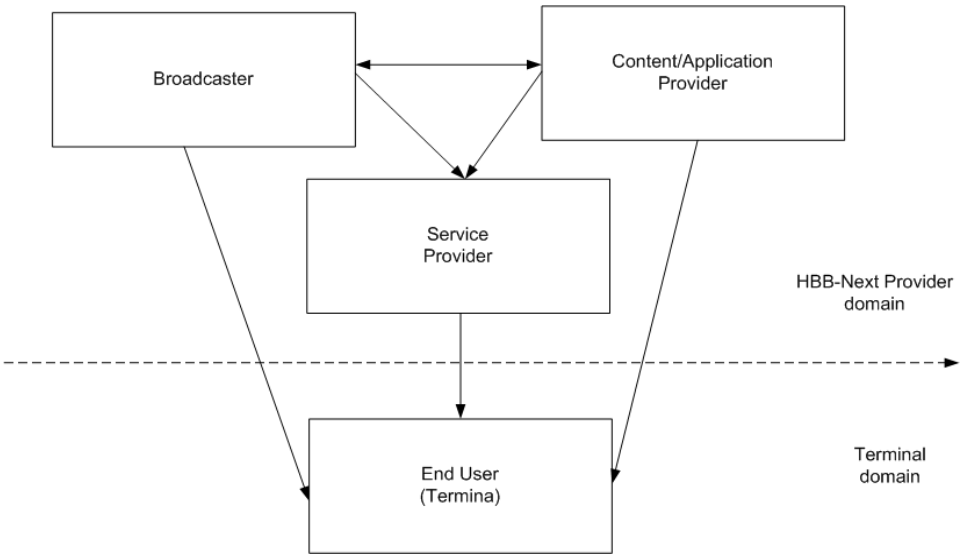

Figure 5: Generic HBB-NEXT domain model

Business Role	Description
Broadcaster	Offers live TV services to the consumer over a broadcast network. Offering hybrid broadcast broadband services to the consumer, would combine the broadcaster role with the application-provider role.
HBB-NEXT Application Provider	The application provider offers content-related applications to the consumer over a broadband or broadcast network. He makes use of the services of the HBB-NEXT service providers. Examples of HBB-NEXT services offered by an application provider are recommendation services and sign language services. Depending on the business models the application provider might be a broadcaster, a 3rd party enriching broadcast services, etc.

Business Role	Description
HBB-NEXT Service Provider	Provides services of the HBB-NEXT framework to the consumer, broadcaster and/or application provider. HBB-NEXT has identified the following service providers in D2.4: ▪ Media Cloud Service Provider ▪ Identity Provider ▪ Application Trust Provider ▪ Rich Media Provider ▪ Multimodal I/F Hardware Vendor ▪ Group Recommendation Provider A single service provider can take either one or all multiple of these roles.
Consumer	The consumer domain consists of the user(s) who consumes the content and all of his devices. The consumer role has one or more HBB-NEXT enabled terminals that include some of the enablers of the HBB-NEXT framework, those services can be used by the application provider: A/V content synchronization Identity of its user (for horizontal markets)

Examples how multiple HBB-NEXT service providers can cooperate are given in D2.4 [4].

6. HBB-NEXT Detailed Architecture

This chapter documents the detailed HBB-NEXT system architecture by identifying and describing the software modules which are realized in the HBB-NEXT framework.

Chapter 6.1 gives an overview of the modules and their relations. In chapter **Fehler! Verweisquelle konnte nicht gefunden werden.** the physical distribution of the components is shown.

In chapter 6.3 a description of each module is included, giving first a functional overview and providing the detailed information like APIs, software and hardware requirements.

6.1. Logical Architecture

Figure 6 shows the components (i.e. software modules) which construct the HBB-NEXT framework. Each module is placed in one out of three domains: Application provider, service provider and terminal. If a module is located in the terminal domain, it means that it is installed on or part of the user's terminal device. Modules in the service provider domain are located at the service provider side including broadcasters, and be accessed either via a broadcast or broadband network. Applications run on the terminal devices, e.g. as HTML/JavaScript pages, using services provided in the network from the application provider (backend) and other service providers.

The diagram is a UML component model including interfaces between the components using the assembly connector. The connector bridges between a component which provides an interface – the full circle – and another component which requires or uses that interface – half of a circle. The colour of a component shows which work package of HBB-NEXT is responsible for developing the software for it. Please refer to the legend in the diagram. The subsequent table describes each component and gives a direct link to the chapter of the detailed component description.

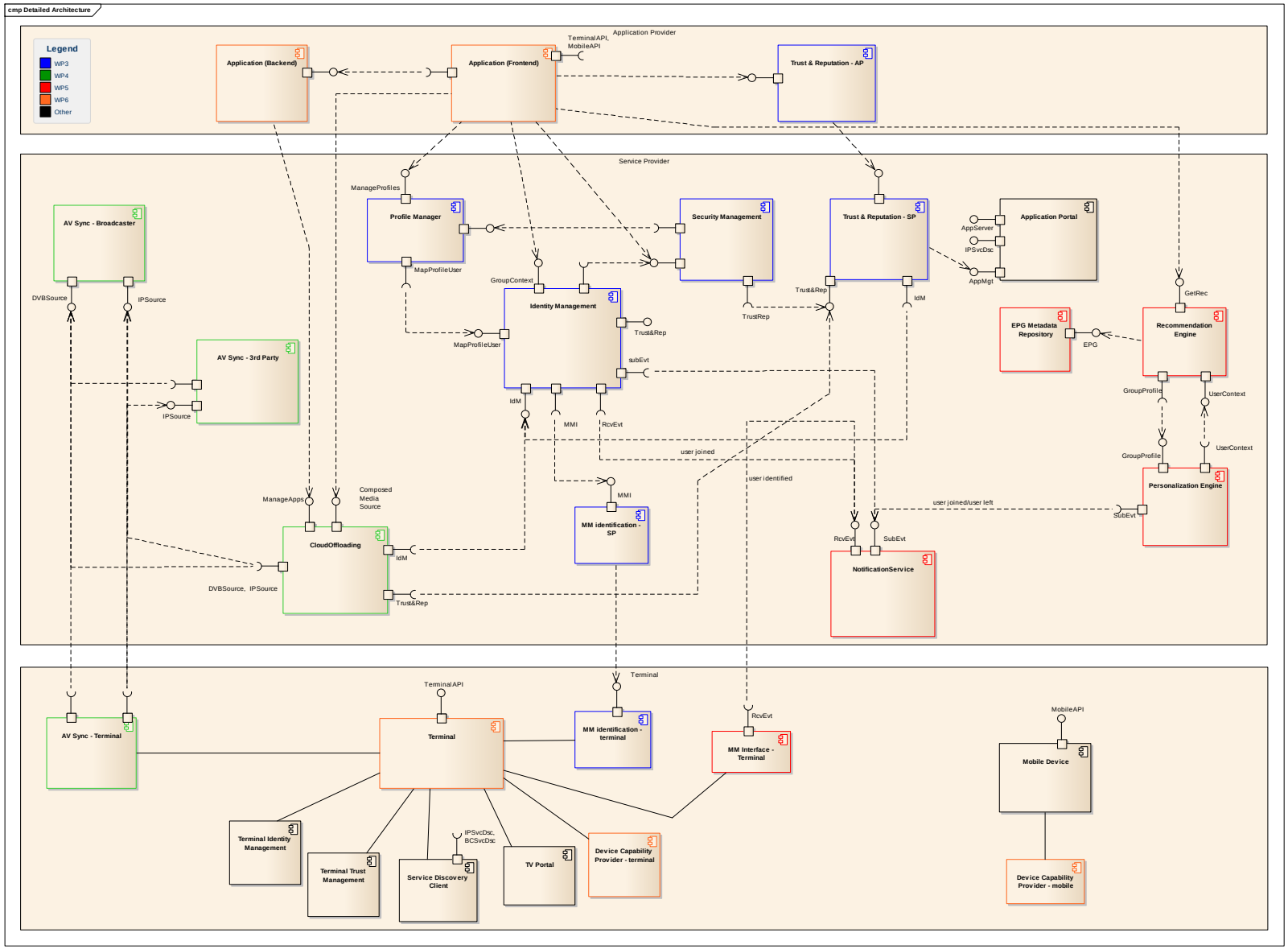

Figure 6: Components of the HBB-NEXT architecture

WP3 - Module	Description	Chapter
Profile Manager	Stores the user profile including user-service relations and information of services and applications.	6.3.5
Identity Management	Provides identity information of users and their devices. It stores user/device and user/user relations.	6.3.3
Security Manager	Enables application and service authorization. Allows authentication of users. Provides user identity information.	6.3.15
Multi-model Identification	Enables single- and multi-user identification by using cameras, microphones, etc.	6.3.6
Trust & Reputation	Collects feedback from users about applications. Computes the reputation of an application based on the feedback. Provides reputation information to application portals, etc..	6.3.8

WP4 - Module	Description	Chapter
AV Sync – Broadcaster	Delivers content on DVB and IP networks Delivers single service components of DVB services over IP Signalling for IP content to DVB delivery	6.3.7
A/V Sync – 3rd party	Provides additional content to linear broadcast content, e.g. a sign language service Synchronizes additional content to broadcast stream	6.3.7
A/V Sync – Terminal	Synchronizes and renders content received from multiple networks Support for a single content provider Support for 3rd party content provider	6.3.7
Cloud Offloading	Offers functionalities which cannot run directly on devices, e.g. due to performance or cost reasons, on a network server.	6.3.13

WP5 - Module	Description	Chapter
Recommendation Engine	A module enabling context-aware personalised content recommendations for single users and groups of users It supports two filtering types (metadata-based filtering and collaborative filtering) Content recommendations can be given for both source-domains, the broadcast and the broadband domain	6.3.2
Preference Profile Service	Part of the recommendation engine	6.3.2
Personalization Engine	Aggregates single user profiles Applies logic to calculate group profile Combines presence information and content that is being consumed and turns this into ratings / preferences	6.3.1
Notification Service	A module enabling subscription to events, and providing notification of events to all subscribers Events in the HBB-NEXT context are to be understood as: a channel change, a person entering/leaving the room, a 2nd screen device being in sync with a TV/STB, etc	6.3.3
Multi Modal Interface		6.3.6
EPG metadata repository	Provides program and content information for broadcast and broadband networks.	6.3.11

WP6 - Module	Description	Chapter
Terminal	HBB-NEXT enabled television set or STB	6.3.14
HBB-NEXT Application	HBB-NEXT applications will be described in the upcoming versions of D6.3	D6.3.x
Device Capability Provider	Provides capabilities of end user devices, like supported A/V codecs, resolutions number of video codecs and renderers	6.3.10

Other	Description	Chapter
Mobile Device	Personal devices like tablet PCs, smart phones, etc.	6.3.14
Terminal Trust Management	Manages access to sensitive terminal API by applications	6.3.12
Terminal Identity Management	Terminal based user and second device management	6.3.12
Service Discovery	Broadcast Services Discovery IP Services Discovery	6.3.9

6.2. Physical Distribution

This chapter will describe the physical architecture of the HBB-NEXT framework in the next version of this document. A first draft can be seen in Figure 3.

The physical realisation of individual components is contained in the detailed module description in the next chapters.

6.3. Detailed Module Descriptions

The following chapters will focus particularly on the functionality of all identified HBB-NEXT modules will be described, including mostly a description of their architecture and interfaces.

6.3.1. Personalization Engine

The Personalization Engine (PE) component is responsible for providing personalized group profiles on the base of single user profiles. The component shall retrieve single user profiles and use the parameters of single users to provide group parameters. In its simplest form (and on the implementation roadmap) the PE will retrieve the ages of all users that are active in a context and provide the average age and standard deviation (or its square empirical variance). The PE should be capable of being extended in any form with more complex algorithms (e.g. movie categories abstraction to higher category).

6.3.1.1. Interfaces, Protocolsand API

The interface of the PE module uses a REST API to provide the information to other components. The REST interface shall provide the following functions (summarized from tech. requirements):

- Retrieve and modify single user profile in the system (via PM)
- Get and set group profile for the active context

The REST API concept will be described in the following. It shall be noted that it is subject to changes due to the parallel work in WP5.

Root URL: https://pe.example.org/api/v1/

- Transport: https (enforced), http for debugging purposes
- Base URL: pe.domain.tld (subdomain pm. assumed to point to PM module)
- Level /api/: As a front-end may be available on the main URL, the level 'api' is added
- Level /v1/: The version is added to the URL. The latest version can be available under the URL without version string (link)

The API itself provides a list of resources under the /resources link. The planned resource is

- /context

Resources (contexts) have unique IDs that can be retrieved from the IdM, e.g., /context/AxfG42. The context can be provided based on a user, a set of users, or based on a device, or set of devices respectively.

Selective profile parameter values can be queried using the attribute name in the form /user/a1b2c3?q=age.

Unless specified otherwise, the default content is used in the body. The requested resource might specify the content type in case multiple content types are offered (e.g. /user.json or /user.xml).

The method GET is used to retrieve an object.

The method PUT is used to modify (update) an object.

The PE is a stateless component that uses data from both the IdM and PE and provides an answer “on the fly”. For more complex operations, it may be extended with a database and stateful logic to pre-calculate context data.

6.3.2. Recommendation Engine

The recommendation engine module provides functionality to Applications and Services to receive context-aware personalized multi-user and multi-device-based content recommendations. For optimizing recommendations, context information from users in a group and the collection of user profile information will be used for content filtering, as well as collaborative content filtering algorithms, all of them serving as input for the service personalization processes. The recommendation engine will enable content recommendations coming from both the broadcast and broadband domain by building on algorithms for merging the HBB metadata. For providing this functionality a metadata merging function will be implemented.

6.3.2.1. Module Internal Structure and Interfaces

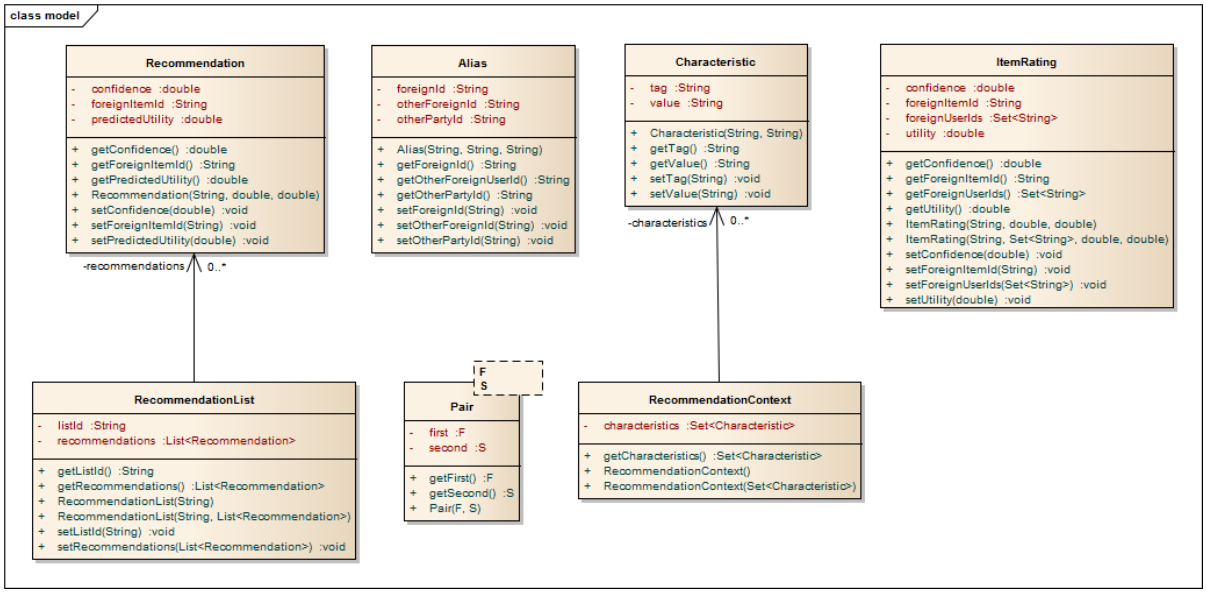

Figure 7: Internal classes used by the Recommendation Engine

The recommendation module consists of several sub-modules, which all of them are used by the Recommendation Engine implementation to calculate and process content recommendations – also taking into account context and characteristics (see Figure 7).

6.3.2.2. Module External Interfaces and APIs

The interfaces providing recommendations based on the hybrid implementation of content-based-filtering and collaborative-filtering algorithms are externally visible, as depicted in Figure 8. They can be accessed by Applications and Services through the *Recommendation Engine* and the *Preference Service*.

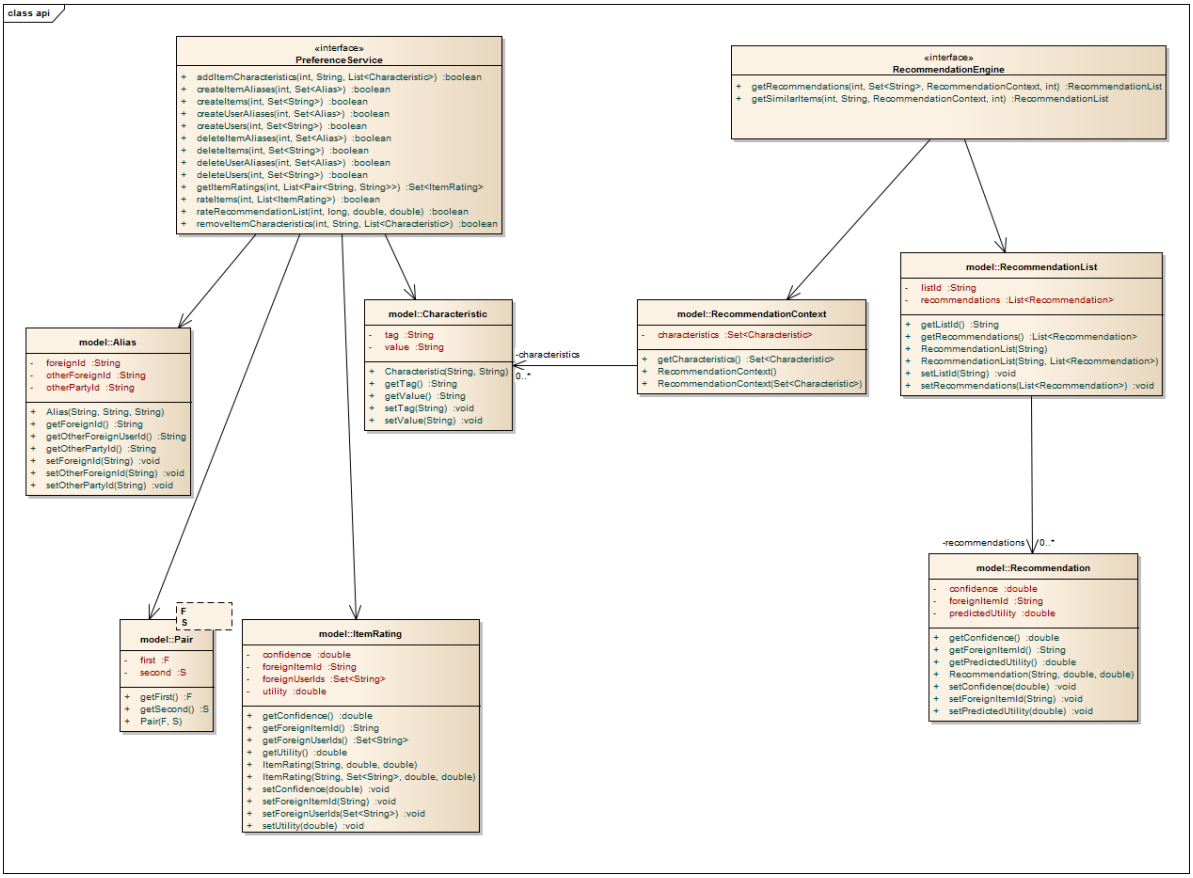

Figure 8: Interfaces provided by the Recommendation Engine

Recommendation Engine:

This interface defines the Recommendation Engine API. A Recommendation Engine recommends items based on individual and/or group preferences as well as the group context. It offers the following methods:

```
RecommendationList getRecommendations(int partyId,
                                      Set<String> foreignUserIds,
                                      RecommendationContext context,
                                      int maxSize)

Get recommendations for a set of users (could be one) based on their (combined)
preferences. The recommendation context contains additional constraints that
must be taken into account while calculating the recommendations.
```

```
RecommendationList getSimilarItems(int partyId,
                                   String foreignItemId,
                                   RecommendationContext context,
                                   int maxSize)

Get items that are in some way similar to the given item.
```

Preference Service:

This interface defines the Preference Service API. The Preference Service enables the storage and retrieval of individual and group preferences (storage in both ways, explicit by a user and/or implicit by the system). It offers the following methods:

```
Boolean addItemCharacteristics(int partyId, String foreignItemId,
List<Characteristic> characteristics)
Add Characteristics to the item that is identified by the given foreignId.

Boolean createItemAliases(int partyId, Set<Alias> itemAliases)
Create item aliases that refer to items that already exists at another party.

Boolean createItems(int partyId, Set<String> foreignItemIds)
Create new items that are identified by the party with the given foreignIds.

Boolean createUserAliases(int partyId, Set<Alias> userAliases)
Create user aliases that refer to users that already exists at another party.

Boolean createUsers(int partyId, Set<String> foreignUserIds)
Create new users that are identified by the party with the given foreignIds.

Boolean deleteItemAliases(int partyId, Set<Alias> itemAliases)
Delete item aliases.

Boolean deleteItems(int partyId, Set<String> foreignItemIds)
Delete the items that are identified by the party with the given foreignIds.

Boolean deleteUserAliases(int partyId, Set<Alias> userAliases)
Delete user aliases.

Boolean deleteUsers(int partyId, Set<String> foreignUserIds)
Delete the users that are identified by the party with the given foreignIds.

Set<ItemRating> getItemRatings(int partyId,
                          List<Pair<String,String>> userItemPairs)
Get the ratings provided by the given users for the specified items.

Boolean rateItems(int partyIds, List<ItemRating> ratings)
Rates content items with the given parameters.

Boolean rateRecommendationList(int partyId, long listId,
                          double utility, double confidence)
Rate a list of recommended items to i.e learn preferred list properties.

Boolean removeItemCharacteristics(int partyId, String foreignItemId,
                    List<Characteristic> characteristics)
Delete Characteristics from the item that is identified by the given foreignId.
```

6.3.2.3. Hardware and Software Components

Hardware components:

Since the implementation of the Recommendation Engine will be purely Java-based, the Recommendation Engine as such will be deployed on a standard off-the-shelf PC.

Software components:

The Java-based implementation will be installed on a PC and accessed through a Web Service Interface by an application running on a demo-STB.

6.3.2.4. Prototype Description

The Recommendation Engine prototype will realize the following scene from one of the Usage Scenarios, which is:

"One person is browsing through her personalized EPG. As a second person from the same household enters the room and takes seat on the couch, the system detects the person, and the EPG updates itself according to the ad-hoc group profile built by the mix of the two users' profiles being present."

6.3.2.5. Interaction and Integration with other Modules

Basic features of the User Identification module (WP3) will be used by this prototype in order to realize the automatic update of the EPG / the recommendations given to the two users.

6.3.3. Notification Service

The notification service will be based on a message queue server to send and subscribe for events of the HBB-NEXT framework. The component will be further described in the next version of this document.

6.3.4. Identity Management

The Identity Management (IdM) component is responsible for managing the information about users, devices and their respective relation. This includes, but is not limited to, user identifiers, links to user profiles, authorization and authentication data and device IDs.

The component shall manage all user identities with related information.

The component shall manage all device identities with related information.

The component shall manage relations between users, between devices, between users and their devices, and between devices and their active users.

It stores data mainly for the multi-modal interface (identification vectors, information about users on the device, etc.) and the security manager (authentication data for identity provider functionality). Identification vectors contain the parameterized information of modalities. They will be exchanged with the multi-modal interface[1]. The API provides means to store the vectors like for any other parameter.

The identity manager is tightly coupled with the profile management, which contains the user profile and service related information (see section 6.3.5).

6.3.4.1. Interfaces, Protocols and API

The interface of the IdM module uses a REST API to provide the information to other components.

The REST interface shall provide the following functions (summarized from tech. requirements):

- Retrieve, add, modify, delete user in the system
- Add, modify, delete user from a context
- Retrieve users active in a context (at the present moment)
- Retrieve users relevant for a context (usually active there)
- Retrieve devices active in a context (at the present moment)
- Retrieve devices relevant for a context (usually active there)
- Provide multi-modal vectors for users relevant for a context (single modes, all modes)
- Fulfil requirements of security management (e.g. credentials)

[1] Note: The user interaction interfaces and development (front-end, dialogs, capturing, etc.) is not provided by the identity management module.

- Retrieve, add, modify, delete device in the system
- Add, modify, delete device from a context
- Attach, detach user to device
- Retrieve devices of a user
- Retrieve, add, modify, delete roles of a user
- Retrieve, add, modify, delete relation between a user (relation, value)

The REST API concept will be described in the following. It shall be noted that it is subject to changes due to the parallel work in WP3.

Root URL: https://idm.example.org/api/v1/

- Transport: https (enforced), http for debugging purposes
- Base URL: idm.domain.tld (subdomain idm. assumed to point to IdM module)
- Level /api/: As a front-end may be available on the main URL, the level 'api' is added
- Level /v1/: The version is added to the URL. The latest version can be available under the URL without version string (link)

The API itself is provides a list of resources under the /resources link.

The planned resources are

- /user/
- /device/
- /context/

Resources (users, devices, contexts) have unique IDs that can be queried, e.g., /users/a1b2c3 or /device/314159. The response body will contain the complete data set in this case.

Selective data parameter values can be queried using the attribute name in the form /users/a1b2c3?q=email or /users/a1b2c3?q=firstName+lastName or /device/?user=a1b2c3.

Data can be searched using the search string for the value in the form /users/?q=name&s=Jon.

Unless specified otherwise, the default content is used in the body. The requested resource might specify the content type in case multiple content types are offered (e.g. /user.json or /user.xml).

The method GET is used to retrieve an object.

The method POST is used to create an object.

The method PUT is used to modify (update) an object.

The method DELETE is used to delete an object.

The planned internal architecture of the module is shown in Figure 9.

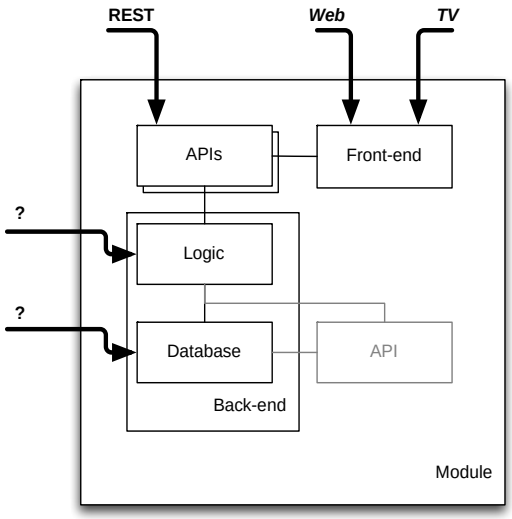

Figure 9: Internal module architecture

The API will communicate with other modules via REST. The same API can be used by front-end components (e.g. web site, menu and dialogs on the TV screen) to provide the module functionality to the user. Front-end components are not part of the planned implementation.

The API will access the back-end database; programming logic will connect database and API. The database may be queried directly or through an API it provides.

6.3.5. Profile Management

The Profile Management (PM) component is responsible for managing user and service profiles. Profiles contain services and their respective preferences.

The component shall manage all user profiles.

The component should manage service profiles.

The profile management contains the user profile and service related information.

6.3.5.1. Interfaces, Protocols and API

The interface of the PM module uses a REST API to provide the information to other components.

The REST interface shall provide the following functions (summarized from tech. requirements):

- Retrieve, add, modify, delete profile in the system
- Fulfil requirements of security management (e.g. privacy)

The REST API concept will be described in the following. It shall be noted that it is subject to changes due to the parallel work in WP3.

Root URL: https://pm.example.org/api/v1/

- Transport: https (enforced), http for debugging purposes
- Base URL: pm.domain.tld (subdomain pm. assumed to point to PM module)
- Level /api/: As a front-end may be available on the main URL, the level 'api' is added
- Level /v1/: The version is added to the URL. The latest version can be available under the URL without version string (link)

The API itself is provides a list of resources under the /resources link.

The planned resource is:

- /user/

Resources (user profiles) have unique IDs that can be queried, e.g., /user/a1b2c3. The response body will contain the complete data set in this case.

Selective data parameter values can be queried using the attribute name in the form /user/a1b2c3?q=services.

Data can be searched using the search string for the value in the form /user/?q=serviceid&s=98765.

Unless specified otherwise, the default content is used in the body. The requested resource might specify the content type in case multiple content types are offered (e.g. /user.json or /user.xml).

The method GET is used to retrieve an object.

The method POST is used to create an object.

The method PUT is used to modify (update) an object.

The method DELETE is used to delete an object.

The architectural concept of the PM module is the same as for the IdM module.

6.3.6. Multi-Modal User Interface

The multimodal interface is responsible for seamless user recognition and authentication using modalities (voice, face detection, etc.). Beside this the multimodal interface serves for commands using gestures or voice to control the STB. The modalities used in the system are coming from proposed scenarios and functional requirements.

6.3.6.1. Module Internal Structure

The architecture is depicted in following Figure 10:

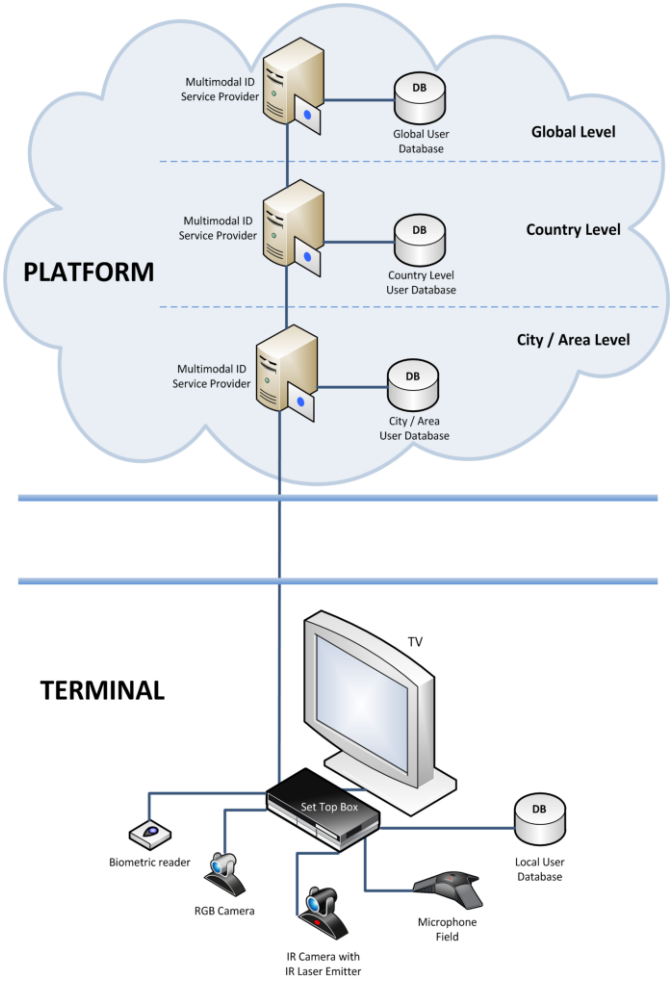

Figure 10: Architecture for Multimodal Interface

The architecture should be layered as there are following requirements:

The STB should be able to identify local users (users who are known to the system, e.g. at home). Therefore, the STB should be able to act without any internet connectivity.

The recognition in local area should be possible with current means, but worldwide seamless user identification is not possible now. Therefore, interconnecting the STB to local (City/Area) level for seamless identification has been suggested. Any other identification type would be done using some additional information for reliable user identification.

6.3.6.2. Module External Interfaces and APIs

The APIs are based on requirements and are optimized for bandwidth requirements. Detailed interface are:

`Boolean MMIGetVoiceActivity (intDetectionTimePeriod)`
Return the result of voice activity detection in the defined interval.

`Boolean MMIInitVoiceActivityDetector (ListOfParameters)`
Initiate the routine for voice activity detection with starting values of parameters. False if initialization errors.

`MMIStopVoiceActivityDetector ()`
Stop the routine for voice activity detection.

`Boolean MMIInitVoiceUserIdentification (ListOfParameters)`
Initiate the routine for speaker identification with starting values of parameters. False if initialization errors.

`MMIStopVoiceUserIdentification ()`
Stop the routine for speaker identification.

`Boolean MMIGetVoiceUserIdentity (intDetectionTimePeriod, intUserIndex, float confidence)`
Return index of the last speaker with the estimated confidence. False if no identification within DetectionTimePeriod.

`Boolean MMIGetAllVoiceUserIdentity (intDetectionTimePeriod, int list UserIndex, float list confidence)`
Return the list of indexes of detected speakers within the detection interval with the estimated confidences. False if no identification within DetectionTimePeriod.

`Boolean MMIRecordNewVoiceUser (intMaxRecordTimePeriod, intNewUserIndex)`
Records and stores a new user within the MaxRecordTimePeriod interval. False if no voice activity within MaxRecordTimePeriod was detected.

`MMIRemoveVoiceUser (intUserIndex)`
Remove user defined by UserIndex out of user list.

`Boolean MMIVoiceUserPresent (intDetectionTimePeriod, intUserIndex, float confidence)`
Return true if a user defined by UserIndex was active within the detection interval and its confidence. False if not.

`Boolean MMIUnknownVoiceUserPresent(intDetectionTimePeriod, float confidence)`
Return true if an unknown user was active within the detection interval and its confidence. False if not.

`Boolean MMIGetFacePresence (intDetectionTimePeriod)`
Return the result of face presence detection in the defined interval.

`Boolean MMIInitFaceActivityDetector (ListOfParameters)`
Initiate the routine for face presence detection with starting values of parameters. False if initialization errors.

`MMIStopFacePresenceDetector ()`
Stop the routine for face presence detection.

`Boolean MMIInitFaceUserIdentification (ListOfParameters)`
Initiate the routine for user identification with starting values of parameters. False if initialization errors.

`MMIStopFaceUserIdentification ()`
Stop the routine for user identification.

`Boolean MMIGetFaceUserIdentity (intDetectionTimePeriod, intUserIndex, float confidence)`
Return index of the last user with the estimated confidence. False if no identification within DetectionTimePeriod.

`Boolean MMIGetAllFaceUserIdentity (intDetectionTimePeriod, int list UserIndex, float list confidence)`
Return the list of indexes of detected users within the detection interval with the estimated confidences. False if no identification within DetectionTimePeriod.

`Boolean MMIRecordNewFaceUser (intMaxRecordTimePeriod, intNewUserIndex)`
Records and stores a new user within the MaxRecordTimePeriod interval. False if no face activity within MaxRecordTimePeriod was detected.

`MMIRemoveFaceUser (intUserIndex)`
Remove user defined by UserIndex out of user list.

`Boolean MMIFaceUserPresent (intDetectionTimePeriod, intUserIndex, float confidence)`
Return true if a user defined by UserIndex was active within the detection interval and its confidence. False if not.

`Boolean MMIUnknownFaceUserPresent(intDetectionTimePeriod, float confidence)`
Return true if an unknown user was active within the detection interval and its confidence. False if not.

6.3.6.3. Hardware and Software Components

The following hardware components are supported by the multi modal interface:

- microphone array
- video camera
- optionally infrared camera
- optionally infrared laser

6.3.7. A/V Content Synchronisation

The synchronisation module as a whole provides the functionality to synchronise two streams coming from IP and a DVB network. The streams either originate from one broadcaster, where one broadcast service has been split up into one DVB source and one IP source or a third party provides additional IP content to a broadcast service. In this case the IP content is being aligned to the DVB content by the third party.

This module also provides the DVB and IP play-out server which can be used for normal DVB and IP playout.

6.3.7.1. Module Internal Structure

The synchronisation module consists of three sub-modules: The broadcaster, the 3rd party and the synchronisation module in the terminal.

The broadcaster and the synchronisation module each provide two interfaces: One for providing/receiving DVB and one for IP sources.

The 3rd party module provides additional content to the DVB source. To align this it receives the DVB source and provides the aligned IP source.

The IP and DVB source enter the synchronisation module in the terminal where the two receivers feed them into the synchroniser which will then feed/control the renderer.

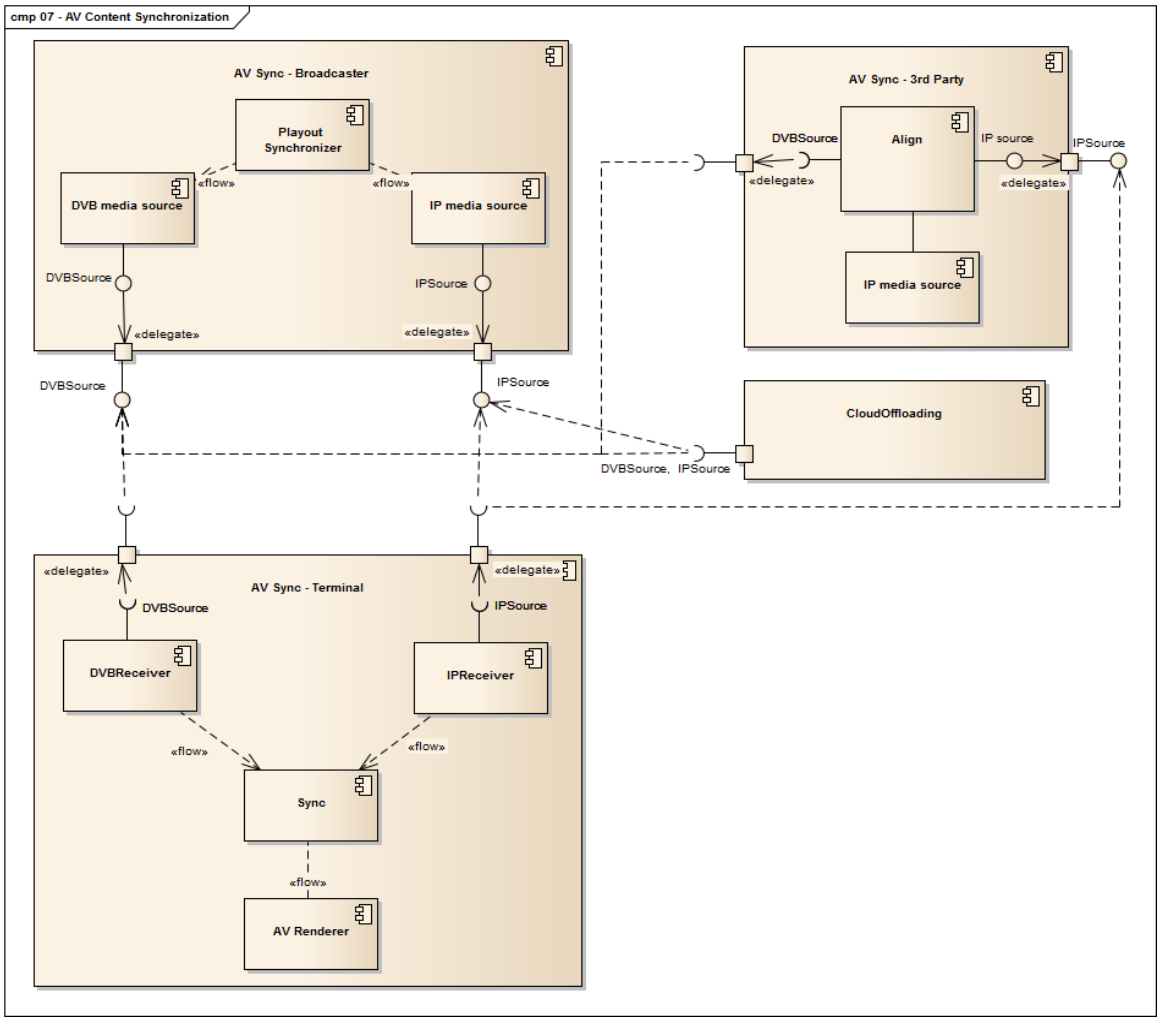

Figure 11: Internal architecture of the A/V content synchronization

6.3.7.2. Module External Interfaces and APIs

The interfaces providing and receiving DVB and IP sources are externally visible. They can be accessed when offloaded to a cloud.

The DVB source interface delivers a DVB transport stream either modulated for DVB-S, C, or T, or streamed over IP. HBB-NEXT may define additional signalling required for synchronizing the DVB source with a corresponding IP source. This will be implemented by the DVB source interface. The IP source interface delivers a media stream using an adaptive streaming protocol like MPEG-DASH.

6.3.7.3. Hardware and Software Components

The module will consist of 3 sub modules, located at different places. There is a broadcaster part which consists of the DVB and IP playout servers described in the subsequent chapters.

The receiver side will be a software component, which will be ported to the prototype STB used in HBB-NEXT (see chapter 6.3.14).

The architectural design of the 3rd party content provider module will be added in the next version of this document.

6.3.7.4. DVB Playout Server

The DVB playout server will provide the following features for the project:

- broadcasting linear TV channels to the STB/TV terminal
- source for IP delivered services or service components
- signalling required for synchronization of broadcast services with IP delivered services or service components at the HBB-NEXT terminals
- application signalling and data transmission over broadcast

The broadcast server of IRT provides the required functionality, except for the signalling of synchronized IP content. It is planned to implement missing features for the HBB-NEXT prototype.

The HBB-NEXT framework will include broadcast-related applications, i.e. applications which are provided by the broadcaster as an extension to the linear TV service. Signalling and transport of broadcast-related applications are defined by HbbTV 1.0 [1].

6.3.7.5. IP Streaming Server

For the IP video server, the project will mainly target HTTP Adaptive Streaming technologies, specifically MPEG DASH [8] and Apple's HTTP Live Streaming [9].

Over the past few years, with the growing popularity of mobile devices such as smartphones and tablets, this type of streaming technology has taken a large share of the market, primarily due to the fact that it is, compared with traditional streaming technologies such as RTP [27], simple, scalable and able to use standard HTTP servers.

One of the main concepts behind the various adaptive streaming protocols is that they split the original content up in numerous fixed length chunks, each of which is independently decodable. By sequentially requesting and receiving chunks, a client can recreate and play out the content. One of the primary advantages of this mechanism is that it allows a server to make each chunk available in multiple qualities, allowing a client to request the quality which best matches his available bandwidth and seamlessly switch between different encodings when the available bandwidth changes.

There are currently a number of competing HTTP Adaptive Streaming protocols in the market, such as Microsoft Smooth Streaming [30], Adobe HTTP Dynamic Streaming (HDS) [31], Apple HTTP Live Streaming (HLS) [9], 3GPP AHS [28][29] and MPEG DASH [8]. In this project, MPEG DASH and Apple HLS have mainly been targeted. The most important reason for choosing MPEG DASH is that it is part of the HbbTV 1.5 [10] specification. The motivation for also looking at Apple HLS is that it is currently the most widely supported and implemented HTTP Adaptive Streaming protocol and the only supported video streaming protocol on a large share of mobile devices.

The following figure gives an overview of the IP Playout server architecture which will be used in the HBB-NEXT project.

For the encoder any of a number of available open-source tools might be used, the most obvious of which are FFMPEG [11] or GStreamer [12].

Figure 12: IP Playout Server architecure

The used Segmenter tool depends on the used HTTP Adaptive Streaming protocol. In case MPEG DASH [8] is being used, there are currently two main open-source alternatives: A set of tools developed by the University of Klagenfurt [13] or the MP4Box tool developed by ParisTech [13].

For hosting the segments created by the Segmenter, any HTTP Server might be used, such as Apache HTTP Server [15] or LigHTTPd [16].

6.3.7.6. Interaction and Integration with other Modules

In case of the terminal not being able to synchronise the two sources this functionality can be offloaded to the cloud by using the cloud offloading module (see chapter 6.3.13).

6.3.8. Application trust

The Application Trust component is responsible for collecting behavioural information about the entities[2] in the system and for computing a reputation value according to it. This could be performed with the aim of determining how trustworthy an entity is in order to carry out a transaction[3] with it.

6.3.8.1. Representing Recommendations

Behavioural information comes from direct past experiences or experiences that other sources have had with a given entity. This information is modelled as recommendations. A recommendation represents the opinion that an entity has about another entity or service based on past interactions between them. In other words, a recommendation determines how an entity rates another entity or a service.

Recommendations will be modelled as a value within a continuous interval [0, 1] in order to ease the computation and the integration in probabilistic calculus. In this way, a recommendation will be converted to this format even though it could be given from different ways by the users, such as a binary value (like, dislike), an integer within a range (from 0 stars to 5 stars), etc.

[2] Within the context of HBB-Next, the term 'entity' for the Application trust module might refer to: a user, an application offered to the users through the app-store, an application provider/developer, another app-store, etc.

[3] Within the context of HBB-Next, the term 'transaction' for the Application trust module might refer to the delivery and installation of an application offered to the users through an app-store.

6.3.8.2. Aggregating Recommendations

From the gathered recommendations, the Application Trust module aggregates them in order to obtain a global reputation value. The internal module in charge of performing this aggregation is named Reputation Computation Engine [17].

If some of the recommendations do not come from direct experiences, the confidence of these recommendations should be taken into account [20]. That is, if a recommendation has been given by other entity, this recommendation is weighted according to the reliability placed in such an entity.

The Reputation Computation Engine should consider the possible incorrect feedbacks provided by either malicious users or simply users that by mistake provide wrong rating values to the relying parties [19]. In this sense, recommendations about the same entity could be compared between them detecting atypical recommendations.

Since there are multiple ways of aggregating recommendations, the module deploys different Reputation Computation Engines. Each of these Reputation Computation Engines implements a different way of aggregating recommendations. At each time, it is selected just one of the Reputation Computation Engine to obtain the global reputation value of a service or entity [18]. The selected one depends on the system conditions or other parameter defined by the system administrator.

One of the ways of aggregating recommendations takes into account the user preferences [21]. In this way, a customized reputation value will be calculated based on certain parameters of the users. For this purpose a Preferences Engine module is defined, which provide to the Reputation Computation Engine the required information about the user preferences.

6.3.8.3. Providing Feedback

After the service has been provided and the user has an opinion about the service, she is able to evaluate it to allow the reputation of that service to be updated. A friendly web-based form will be shown to the users so that she can provide her feedback. This feedback is considered as a recommendation by the system, which in turn will be taken into account for computing future reputation values.

The recommendations are therefore stored in a database to be accessible for the following aggregation process.

6.3.8.4. Proposed Architecture

The components of the Application Trust component are described in the following:

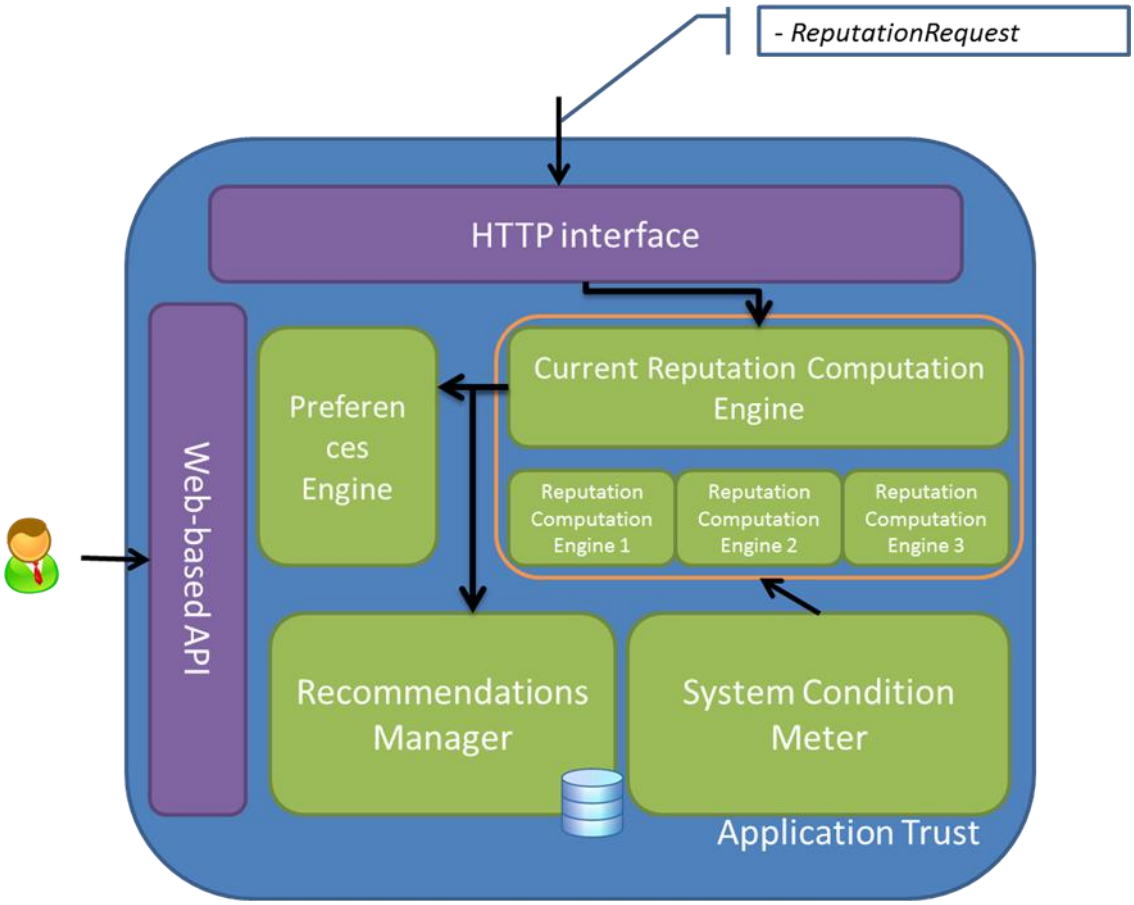

Figure 13: Application Trust, General Architecture Overview

- Current Reputation Computation Engine: This module is in charge of aggregating recommendations in order to get a global reputation value. It requests the Recommendations Manager module for getting the recommendations together with their weights. This module is the one active at the moment.
- Reputation Computation Engine n: Each of these modules represents an inactive Reputation Computation Engine. It becomes the Current Reputation Computation Engine when certain conditions happen in the system, directed by the System Condition Meter.

- Preferences Engine: This module provides to the Reputation Computation Engine the required information to compute a customized recommendation. It is worth mentioning that not all of the Reputation Computation Engines support user preferences.
- Recommendation Manager: Stores and manages the recommendations which the Reputation Computation Engine uses in order to obtain a reputation value.
- System Condition Meter: This module measures the system conditions and other parameters in order to select the most appropriate Reputation Computation Engine at each moment.
- HTTP API: This module provides a service for other modules to request reputation values associated to a service or to another entity.
- Web-based API: This API renders and shows reputation information to the users in a friendly way. It is also in charge of collecting user feedback and making use of the Recommendation Manager to store them.

6.3.8.5. Functionality

Interfaces

The module presents two main interfaces allowing the rest of entities and users to have access to its functionality. These two interfaces, as shown next, are: HTTP interface and User web-based interface.

HTTP Interface

This module presents an internal HTTP interface component acting as a service for the rest of the modules of the architecture and allowing performing reputation request. To this end, the component receives `<ReputationRequest>` messages over HTTP specifying information about the entity or service whose reputation information is desired. This message optionally includes the preferences of the user who wants to use the service in order to provide a customized reputation as previously commented.

A <ReputationRequest> message is sent as an answer to the previous message indicating the reputation once aggregated by the Reputation Computation Engine. Likewise it is done with the recommendations, the value of the reputation is specified as a value within a continuous interval [0, 1].

User Web-Based Interface

A web Interface module is defined to render the information which needs to be shown to the users after a reputation value is calculated. This module is also in charge of showing the web form to the user in order for her to provide the feedback.

This module presents the information in a friendly way to enable accessibility capabilities into the system.

6.3.9. Application, TV Apps Management, Service Discovery

This chapter will describe modules for application hosting, TV apps management and service discovery. However, this version of the system architecture is preliminary and focuses on developments for milestone 6. Therefore this chapter will not be included until the next version of this document.

6.3.10. Multi-Screen Support, Device Monitoring, Device Capabilities

This chapter will describe modules for providing required information of devices for enabling multiscreen features. However, this version of the system architecture is preliminary and focuses on developments for milestone 6. Therefore this chapter will not be included until the next version of this document.

6.3.11. EPG metadata repository

The metadata repository component is a service that provides EPG information for broadcast and broadband services ("hybrid metadata"). The module will receive input in different formats, standard and proprietary formats.

The details of this component are not yet defined. The main “consumer“ of the repository will be the recommendation engine. A potential candidate for the format is TV-Anytime, as it supports both broadcast and broadband A/V content.

For milestone 6 a predefined static set of metadata will be used which contains broadcast and YouTube [34] content.

6.3.11.1. Interaction and Integration with other Modules

The metadata will provide EPG data to the recommendation engine.

6.3.12. Terminal Identity and Trust Management

The terminal services for user-, trust- and device (second screen)-management provide the necessary terminal side functionalities for personalization and communication with companion devices.

The terminal user management module allows the user to login/-out by using the identity of a network identity provider. The function of the identity provider in HBB-NEXT is handled by the security manager. Applications can request the active users of the terminal by a terminal API.

The device management module organises the link connections of personal user devices and is also closely related to the identity management module.

In difference to the identity management module described in chapter 6.3.3, which is service provider/operator based, this module is a terminal service targeting HBB markets where the user does not have a contract with a single service provider (network operator).

The terminal trust management module is contrary to the application trust module responsible for broadcast related application trust. As broadcast related applications are automatically sent when changing the channel this differs from IP and downloadable content and applications. The terminal trust module is used to enable the user to control which applications may access sensitive data like his identity

For the next development cycle of HBB-NEXT it is planned to properly align these modules with the identity management module.

6.3.13. Cloud Offloading and Media Adaptation

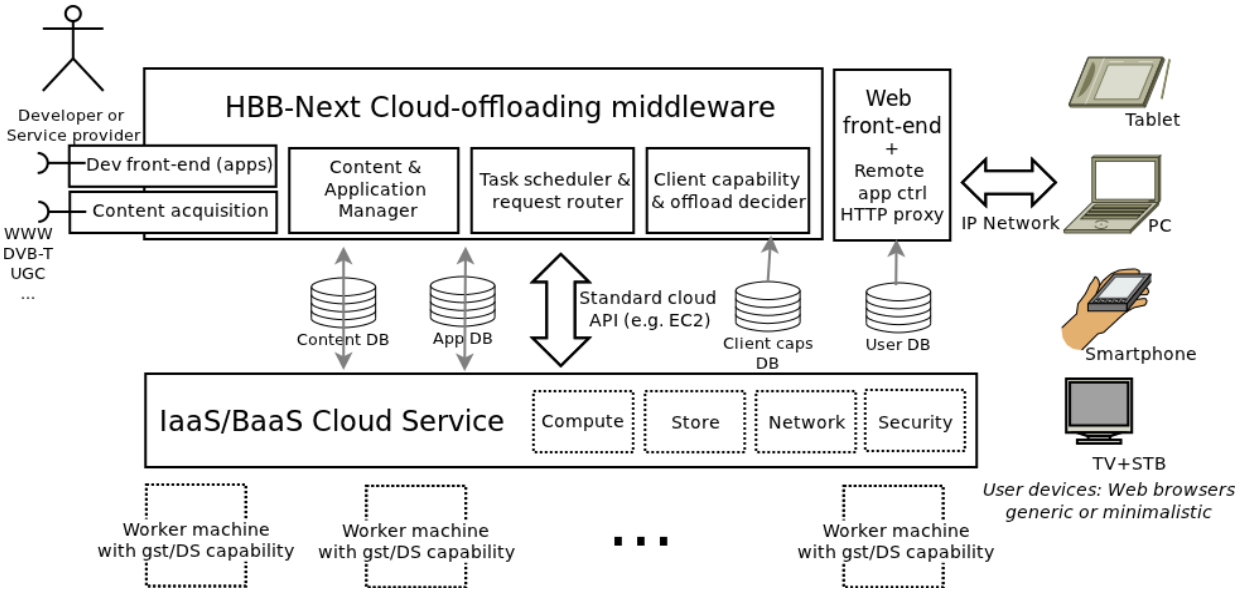

Figure 14: High level Architecture of the HBB-NEXT Cloud Offloading service

The high level architecture for cloud-offloading is shown in Figure 14. The architecture builds on top of an IaaS cloud to provide the HBB-NEXT cloud offloading service.

In this section the underlying considerations are discussed while designing a cloud-based offloading architecture for the HBB-NEXT project. The architecture to implement a scalable cloud-offloading system that can support multiple client devices and diverse media processing applications will furthermore be presented. For the implementation GStreamer [12] for media processing and Openstack IaaS [35] (Infrastructure as a Service) as our infrastructure cloud platform are being considered. Openstack can be swapped for other IaaS cloud services such as Amazon's AWS [36]; this interoperability is ensured by choosing the well-known EC2 [37] cloud API. The suitability of virtual machines for highly processor intensive tasks may be questionable, but in fact the underlying cloud can simply be swapped for MaaS (Metal servers as a service) or Amazon AWS high performance cluster computing services controlled using the standard EC2 IaaS cloud control API.

Compared to a typical web application, which might only entail occasional delivery of HTML documents and database queries per user, media processing applications are much more processor intensive on a sustained basis.

Delays introduced while waiting for cloud worker machines to boot up would affect many more users on average (since each worker machine can only serve a few users and many more have to be spawned on short notice as compared to a typical web application). Moreover, a marginally overloaded web server may not immediately be noticed by users in contrast to an overloaded media server that is unable to serve video at the required frame-rate. Clearly, architecting scalable media applications in the cloud has different performance requirements as compared to a web service.

The above discussion also brings forth an important economic factor in system design, namely, that a commodity cloud server cannot process real-time media streams for too many users. Clearly, assigning a dedicated server resource (e.g. a GStreamer media processing pipeline) to one user is not cost-effective, and the web application assumption of ``one user, one isolated session'' cannot be directly applied in cloud media processing applications.

Fortunately, most users tend to gravitate toward consuming similar content at any given point of time, as observed in web video popularity studies, which report strong adherence to the Pareto principle (e.g. the 10% videos attract 90% user requests). The key scalability driver in the proposed architecture is the detection of duplicate content processing requests across users and devices which occur due to most users requesting the same content. The system then groups such client requests to be served from a single media processing pipeline in order to eliminate duplicate media processing in the cloud.

6.3.13.1. Proposed Architecture

Figure 14 shows the high level architecture of the HBB-NEXT cloud offloading subsystem. The system is built over an IaaS or MaaS cloud through which worker machines can be started or stopped on demand by the HBB-NEXT cloud-offloading middleware. The worker machines are customized with media-processing-specific software such as GStreamer making them capable of executing any application media processing pipeline specified by a developer. Further customizations allow fine grained control and monitoring of these worker machines.

The HBB-NEXT cloud offloading middle-ware comprises of an application and content head-end, which is the portal for developers and content providers to create, store, and schedule content-rich applications on the system. For example, a developer may want two video streams to be mixed for a picture-in-picture functionality. The application to describe this processing may be described as a GStreamer media processing pipeline and stored in the application database (app DB) while the content database can be used to store the associated elementary video streams whose picture-in-picture mixing is desired. This component is also linked to the trust and reputation framework. The content and application manager communicates to the web front-end to present the user with application and content information.

The key controlling component of the system is the task scheduler and request router. Several optimizations, such as fine-grained process-level computational requirements, are taken into account while making scheduling and resource allocation decisions in this component. This component is responsible for routing application processing requests to appropriate worker machines, detecting duplicate processing requests by clients and patching these clients to already-running processing tasks, automatically scaling up (or down) the underlying IaaS/BaaS cloud and monitoring the cloud health to maintain quality of service.

The client capability and offloading decision component is responsible for identifying connecting-client capabilities and accordingly offloading tasks to the cloud from the client device. For example, this component uses the user-agent string from the client's web browser to ascertain the screen size (hence, video resolution) to target while transcoding video for the client. Different device profiles and characteristics are continuously stored and updated in the client caps database to assist with this decision making.

Finally, the web front-end acts as the entry portal to the HBB-NEXT cloud offloading subsystem and controls aspects such as user authentication and integration with other WWW services such as social networks. Moreover, this HTTP server might offer HTTP-proxy functionality for content stream delivery to certain classes of devices. For example, Apple's HTTP live streaming protocol is implemented over the HTTP protocol and the corresponding server may be part of the web front-end.

Another function of the web front-end is to enable application control via second screen devices (e.g. a tablet computer to control a content application being played back on a TV) using HTTP protocols or their real-time derivatives such as web-sockets.

6.3.13.2. Media Adaptation

Media adaptation is needed to deal with heterogeneity across clients and devices, content, networks and protocols. A key success indicator of the HBB-NEXT cloud offloading subsystem will be its ability to support end-device heterogeneity. For example, transcoding media in the cloud can convert media from one format to another, making it possible for devices with limited media decoders to receive videos originally offered in an incompatible video format and transcoded in the cloud. The challenge lies in including the multiple open standard interfaces that different client devices may employ today. For example, some mobile devices are only able to decode video streams presented in specific codecs: Apple devices such as iPads and iPhones do not decode WebM [38] encoded video. A cloud-based media transcoder that converts WebM video into H.264 video [40] will solve this problem. Moreover, while some users' networks may be capable of delivering high resolution media other users' may not have access to such high quality networks at all times. This problem can be addressed by using the cloud to transcode media into a series of resolutions and then offer these various resolutions to clients. The system's client capability manager will include intelligence to make the decisions about how such heterogeneous devices can be served appropriately adapted media from the cloud.

The HBB-NEXT cloud offloading service will include GStreamer-based content detection capabilities and the automatic setup of media processing pipelines to deal with such heterogeneous content. Moreover, it will include a HTTP proxy that converts media into browser-friendly formats (such as WebM for Google Chrome browser [39]).

6.3.14. HBB-NEXT Terminal Devices

The HBB-Next terminal device is a major component of the HBB-NEXT system architecture as shown in Figure 6 (see chapter 6.1) and acts as the interface between the HBB-NEXT network services/applications and the end-user. Therefore, the HBB-NEXT terminal must be capable of presenting various types of media data, of running interactive and downloadable applications and of receiving input from the end-user. The detailed requirements for such a device are defined in chapter 4.3 of this document.

In HBB-NEXT, two general types of terminal device are defined:

- HBB-NEXT terminal (acting as the primary device)
- Second screen device

The primary HBB-NEXT terminal is the one which is connected to the 'big' TV screen. Typical HBB-NEXT terminal devices are TV sets with integrated digital TV decoders, Set-top-boxes and Personal Computers. They have to be compliant to the HbbTV standard [1] offering a basic framework for hybrid broadcast and broadband services. Today, such devices are publicly available and implement state-of-the-art technology. HBB-NEXT is extending this framework.

Some modules like the AV content synchronization need to be implemented on the HBB-NEXT terminal. Figure 15 shows the principle software architecture of the terminal, based on the prototype box from TARA Systems.

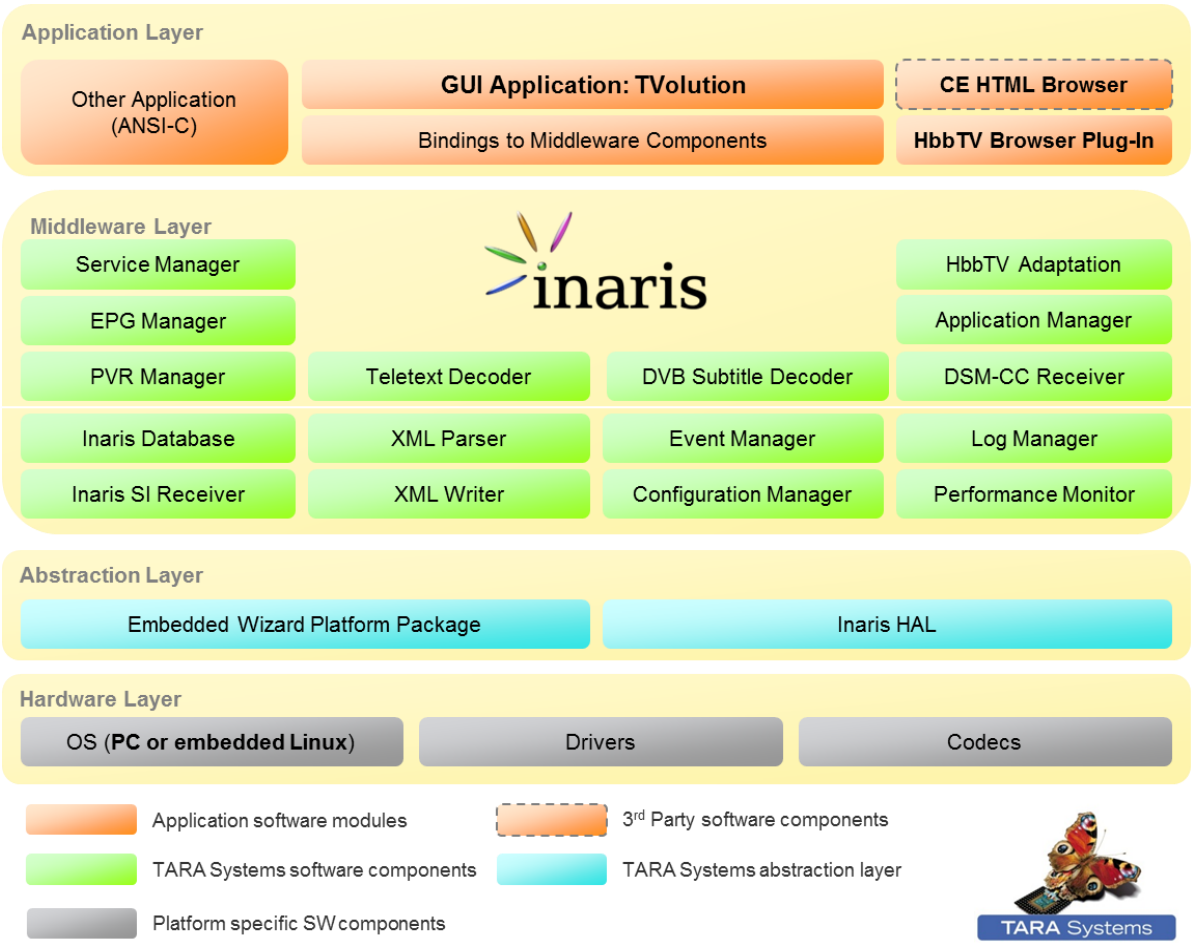

Figure 15: STB software architecture

The second screen device is typically a mobile device such as a smartphone or a pad compute but it can be also any other device which implements the required features. It is connected via the home network or the Internet to a primary device. The content shown on the primary device can be shown on the secondary screen as well, but possibly with modified attributes. For example, the second screen device can be used to present the content shown on the primary device with a different audio language or with additional subtitle information. So, it is easier for impaired people to join the viewing experience in a group and become part of the collective viewing experience. Moreover, a primary device can stream content to one or more second screen devices.

6.3.15. Security Manager

The Security Manager (SM) component is responsible to manage multi-factor authentication, authorization, and policy enforcement for "multifactor levels" and for profile data access control.

The component shall handle authentication, i.e. it acts as an identity provider towards the STB. The IdM shall act as back-end for the authorization process.

It shall provide means to manage and verify tokens.

The component shall handle policies. It shall provide means to manage and enforce policies.

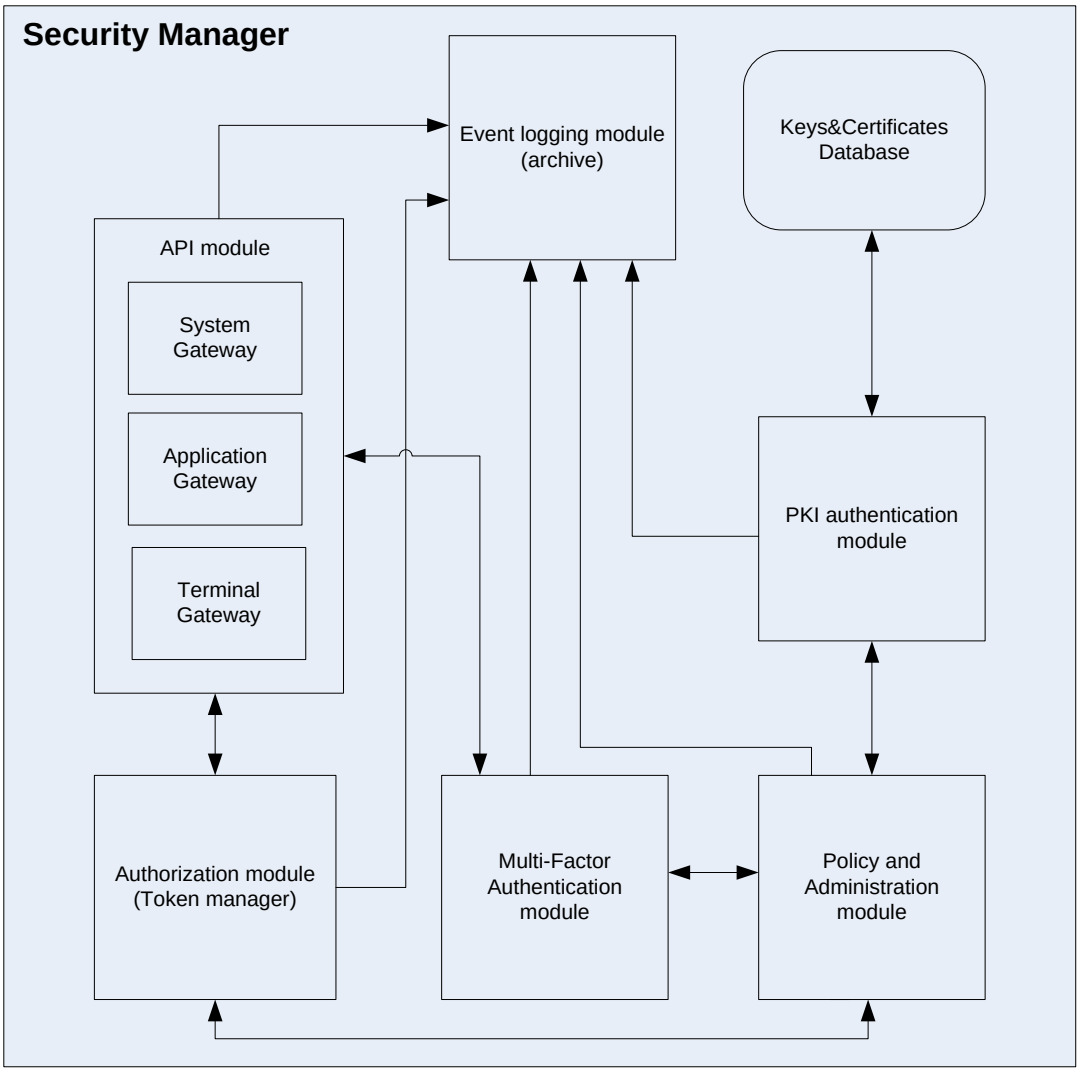

Figure 16: Internal architecture of the security manager

The Security Manager will consist of several modules:

Authorization module (Token manager) – will evaluate if the terminal has access to the HBB Network as role-based access controller. This module will either grant or deny access to the HBB Network.

Multi-Factor Authentication (MFA) module – will be responsible for managing multi-factor authentication processes, depending on required level of security during User/Group is accessing the application/s.

MFA will perform a user identification process to create a list of "best matches" and then perform a series of verification processes to determine a conclusive match. Number of necessary verification processes depends on the required authentication level for accessing the service/application/data. HBB-Next project supposes to have several required levels of authentication to verify users before accessing the service with required “level of security”. (personal EPG – low level is required; instant messaging – middle level is required, e-banking – high level is required)

Policy and Administration module – policy enforcer for MFA module and point of security rules mapping and configuration.

PKI authentication module – will be intended to manage Public Key Infrastructure (PKI) tasks within HBB-Next domain, acting as certificate management system within HBB-Next domain.

Event logging module (archive) – event logger for all activities performed by SM.

Keys&Certificates Database – will hold sensitive cryptographic key information and single public key certificates.

API module – secure interface for communication among HBB-NEXT entities consisting of three different sub-modules, the so called gateways.

- *System Gateway* – an interface towards HBB-Next core modules (Identity Management, Profile Manager, Trust&Reputation)

- *Application Gateway* – an interface towards HBB-Next application, which will be used only for communicating with Application Servers and not to applications hosted on the terminal. *Note – necessity of this interface will be a part of further detailed design analysis within WP3.*

- Terminal Gateway –an interface towards Terminal/ End-user device. This Gateway is needed because of different rules for external devices/terminals might be used than to all servers within Core and Application layers.

7. Conclusion and Outlook

This document describes the initial system architecture of HBB-NEXT. The modules which form the architecture have been identified and their principal interfaces. For some modules it is still under discussion how they fit in the architecture depending on the scenarios and business models behind, e.g. for user identity management. The documentation of the components is preliminary and reflects the state of development, i.e. some have already a defined API.

During the project two major updates of this document are planned, namely D6.1.2 and D6.1.3. They will be used to document the development of the HBB-NEXT framework, completing the module description with APIs, describing the interaction of interconnected modules, and a more detailed set of technical requirements.

One of the key objectives of HBB-NEXT is to contribute to standardization bodies, currently planned are the ETSI MCD Converged Multi-screen Service specification and the next release of the HbbTV specification. The intention is to use the next revision of this document as a basis for contributions on technical requirements, protocols and APIs towards the targeted standardization bodies.

8. References

[1] Hybrid Broadcast Broadband TV (HbbTV): ETSI TS 102 796 V1.1.1 (2010-06)

[2] Web-based Protocol and Framework for Remote User Interface on UPnP™ Networks and the Internet (CE-HTML): CEA-2014-A (2007-07). http://www.ce.org/Standards/

[3] HTML5. http://dev.w3.org/html5/spec/

[4] HBB-NEXT deliverable 2.4: "Description of selected business models" http://www.hbb-next.eu/documents/HBB-NEXT_D2%204.pdf

[5] HBB-NEXT deliverable 2.2: "System, Service and User Requirements" http://www.hbb-next.eu/documents/HBB-NEXT_D2.2.pdf

[6] HBB-NEXT deliverable 7.2: "Dissemination and Standardization Strategy" http://www.hbb-next.eu/documents/HBB-NEXT_D7%202.pdf

[7] Key words for use in RFCs to Indicate Requirement Levels: IETF RFC 2119

[8] Dynamic adaptive streaming over HTTP (DASH): ISO/IEC DIS 23009-1:2012

[9] Apple's HTTP Live Streaming: https://developer.apple.com/resources/http-streaming/

[10] HbbTV 1.5: http://www.hbbtv.org/pages/about_hbbtv/HbbTV-specification-1-5.pdf

[11] FFMPEG: http://ffmpeg.org/

[12] GStreamer: http://gstreamer.freedesktop.org/

[13] University of Klagenfurt: http://www-itec.uni-klu.ac.at/dash/

[14] MP4box tools: http://gpac.wp.mines-telecom.fr/

[15] Apache HTTP Server: http://httpd.apache.org/

[16] LigHTTPd: http://www.lighttpd.net/

[17] A. Josang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision. Decis. Support Syst., 2007.

[18] Félix Gómez Mármol, Marcus Q. Kuhnen, and Gregorio Martínez Pérez. Enhancing OpenID through a Reputation Framework. In Autonomic and Trusted Computing, LNCS 6906, pages 1–18. 8th International Conference, ATC 2011, Springer, sep 2011.

[19] F. Gómez Mármol, G. Martínez Pérez, Security threats scenarios in trust and reputation models for distributed systems, Elsevier Computers & Security 28 (7) (2009) 545–556.

[20] S. Songsiri, MTrust: a reputation-based trust model for a mobile agent system, Autonomic and Trusted Computing. No. 4158 in LNCS. Third International Conference, ATC 2006, Springer, Wuhan, China, Sep. 2006, pp. 374–385

[21] Y. Wang and J. Vassileva, "A Review on Trust and Reputation for Web Service Selection," Proc. 1st Int'l. Wksp. Trust and Reputation Management in Massively Distributed Computing Systems, Toronto, Canada, June 2007.

[22] Roy Thomas Fielding: "Architectural Styles and the Design of Network-based Software Architectures", Dissertation at University of California, Irvine, 2000.

[23] UML Resource Page: http://www.uml.org/

[24] Cross-Origin Resource Sharing: http://www.w3.org/TR/cors/

[25] HTML 5 Web Messaging: http://www.w3.org/TR/webmessaging/

[26] HTML 5 Web Sockets: http://www.w3.org/TR/websockets/

[27] RTP: A Transport Protocol for Real-Time Applications: http://www.ietf.org/rfc/rfc3550.txt

[28] 3GPP TS 26.234 V9.3.0 (2010-06), Transparent end-to-end Packet-switched Streaming Service (PSS) Protocols and codecs (Release 9)

[29] 3GPP TS 26.244 V9.2.0 (2010-06), Transparent end-to-end packet switched streaming service (PSS), 3GPP file format (3GP) (Release 9)

[30] Microsoft Smooth Streaming: http://www.iis.net/download/SmoothStreaming

[31] Adobe HTTP Dynamic Streaming: http://www.adobe.com/products/hds-dynamic-streaming.html

[32] "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.3": ETSI ES 201 812 V1.1.1 (12 2003)

[33] http://opentv.com/

[34] http://www.youtube.com

[35] http://www.openstack.org/

[36] http://aws.amazon.com/

[37] http://aws.amazon.com/ec2

[38] http://www.webmproject.org/

[39] http://www.google.de/chrome/

[40] ITU-T Recommendation H.264 / ISO/IEC 14496-10:2005: “Information technology – Coding of audio-visual objects- Part 10: Advanced Video Coding”.

9. Abbrevations

9.1. General abbrevations

3GPP	3rd Generation Partnership Project
AHS	3GPP Adaptive HTTP Streaming
API	Application Programming Interface
AWS	Amazon Web Services
BaaS	Business as a Service
CE-HTML	HTML for Consumer Equipment, formally known as CEA-2014A [2]
CORS	Cross Origin Resource Sharing
DASH	MPEG Dynamic Adaptive Streaming over HTTP [8]
DVB	The Digital Video Broadcasting Project
DTS	Decoding Time Stamp
EC2	Amazon Elastic Compute Cloud
EPG	Electronic Programme Guide
ETSI	European Telecommunications Standards Institute
IaaS	Infrastructure as a Service
IETF	Internet Engineering Task Force
IP	Internet Protocol
HBB	Hybrid Broadcast Broadband (generic term)
HbbTV	“Hybrid Broadcast Broadband Television” specification [1]
HDS	Adobe HTTP Dynamic Streaming
HLS	Apple HTTP Live Streaming
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
MaaS	Management as a Service
MCD	ETSI Media Content Distribution
MHP	DVB Multimedia Home Platform
MPEG	Motion Picture Expert Group

MSS	Microsoft Smooth Streaming
PCR	Program Clock Reference
PTS	Presentation Time Stamp
REST	Representational State Transfer
RTP	Realtime Transport Protocol
SaaS	Software as a Service
SOAP	Simple Object Access Protocol
STB	Set top box
SVG	Scalable Vector Graphic
UML	Unified Modelling Language
URL	Uniform Resource Locator
WebM	audio video codec

9.2. HBB-NEXT abbreviations

IdM	Identity Management
PE	Personalization Engine
PM	Profile Management
SM	Security Manager
WP	Work Package