SW Architecture Specification

Project: ANDROID4TV

iWedia

Filename	iWedia_Android4TV_SW_Architecture_v1.5
Version	v1.5
Classification	PUBLIC
Status	Approved
Date	May 20, 2014
Author	iWedia

SUMMARY

PREFACE

AUDIENCE

This document is intended for the people who are involved in the process of evaluation, implementation, integration or testing of the Android4TV components. It provides a high-level overview of the Android4TV solution and should help the user to better understand the capabilities of the individual modules or the solution in its entirety.

RELATED DOCUMENTS

#	Title	Description	Version/Date
[1]	ANDROID4TV_Framework_API_Specification_v1.0.doc	Android4TV Java API	V1.0 May 20, 2014
[2]	http://developer.android.com/sdk/index.html	Android SDK	

Table 1 : Related documents

DEFINITIONS/ACRONYMS/ABBREVIATIONS

Definition/Acronym/Abbreviation	Description
PVR	Personal Video Recorder
EPG	Electronic Program Guide
MHEG	Multimedia And Hypermedia Experts Group
HBBTV	Hybrid Broadcast Broadband Television
CA	Conditional Access
CI/CI+	Common Interface/Common Interface Plus
POSIX	Portable Operating System Interface
SDK	Software Development Kit
GUI	Graphic User Interface
API	Application Programming Interface
NPAPI	Netscape Plugin Application Programming Interface
HAL	Hardware Abstraction Layer
DTV	Digital Television
DVB	Digital Video Broadcasting
DVB-S, DVB-C, DVB-T	Digital Video Broadcasting – Satellite, Cable, Terrestrial
AIT	Application Information Table
JNI	Java Native Interface
DLNA	Digital Living Network Alliance
DTCP	Digital Transmission Content Protection
DMP, DMR, DMS, DMC	Digital Media Player, Renderer, Server, Controller
UPnP	Universal Plug and Play
DTG	Digital TV Group
HD	High Definition
RTOS	Real Time Operating System

Table 2 : Definitions/acronyms/abbreviations

1 INTRODUCTION

We are witnessing a transformation of TV set from a device with pre-defined functions, as we have known it from the moment of origination in the 30s of the last century, into a programmable device with functions that could easily be changed during exploitation. Technology drivers for these changes are increasing processing power of SoCs intended for TV, and development of infrastructure for application distribution in the form of various application stores, and the significance of networked devices and content in the homes.

On the other hand we have witnessed the success of Android on the mobile market which beside other things, provides the solution for stratified market and have an enormous impact on a way of how the applications being distribututed on different devices, and across multiple platforms and geographies. These days, the power of the Android is recognized by the TV/STB manufacturers as well. However, the biggest challenge for the manufacturers and the whole Android Community is ability to set the standard in the field of TV. iWedia has industry proven solution which certainly responds to the challenge.

In order to extend the Android capabilities with DTV related functions, iWedia offers the integration of DTV middleware (MW) with the accompanying Java based framework into the standard Android software stack as a part of the Android4TV software solution. The Android4TV extension does not affect the application framework, which comes with Android by default, while offering a standardized model for implementation of TV-centric applications. Consequently, the developers are not aware of this middleware integration.

Android4TV based applications are implemented in the same way as any other Android application, in Java programming language, using the standard Android SDK. In this way, the applications based on ANDROID4TV solution can be used along with the large number of already available Android applications.

Android4TV solution aims to be platform agnostic. This is supported by the fact that the software was initially ported to Gingerbread version and then maintained to support all upcoming versions as well as various SoC architectures.

1.1 FEATURES

Android4TV main goal is to extend the Android application framework in order to provide following features:

- DTV features like channel list, channel setting, now-next info, EPG, Teletext, Subtitles.
- Screen and control sharing between widgets and TV broadcast application
- Remote Control Navigation (TV can be remotely controlled by another Android mobile phone or tablet).
- Remote configuration and monitoring using TR-069 protocol.
- Other TV applications downloadable through Android market or other OEM branded application store.

Main use cases for Android4TV software solution in Android OS are as following:

- Access installed Android applications while streaming Live TV in background
- Data retrieval using active Internet connection
- Support for both, web and broadcast based services (HbbTV, MHEG, IPTV, SAT>IP)
- Support for asynchronous events from other Android applications

1.2 HIGH LEVEL SOFTWARE ARCHITECTURE

Android4TV software solution is divided into several software layers and each belongs to specific part of the Android OS. Each layer provides some specialized service for others and provides modular structure where specific blocks can be potentially replaced by others with same functionality (Figure 1).

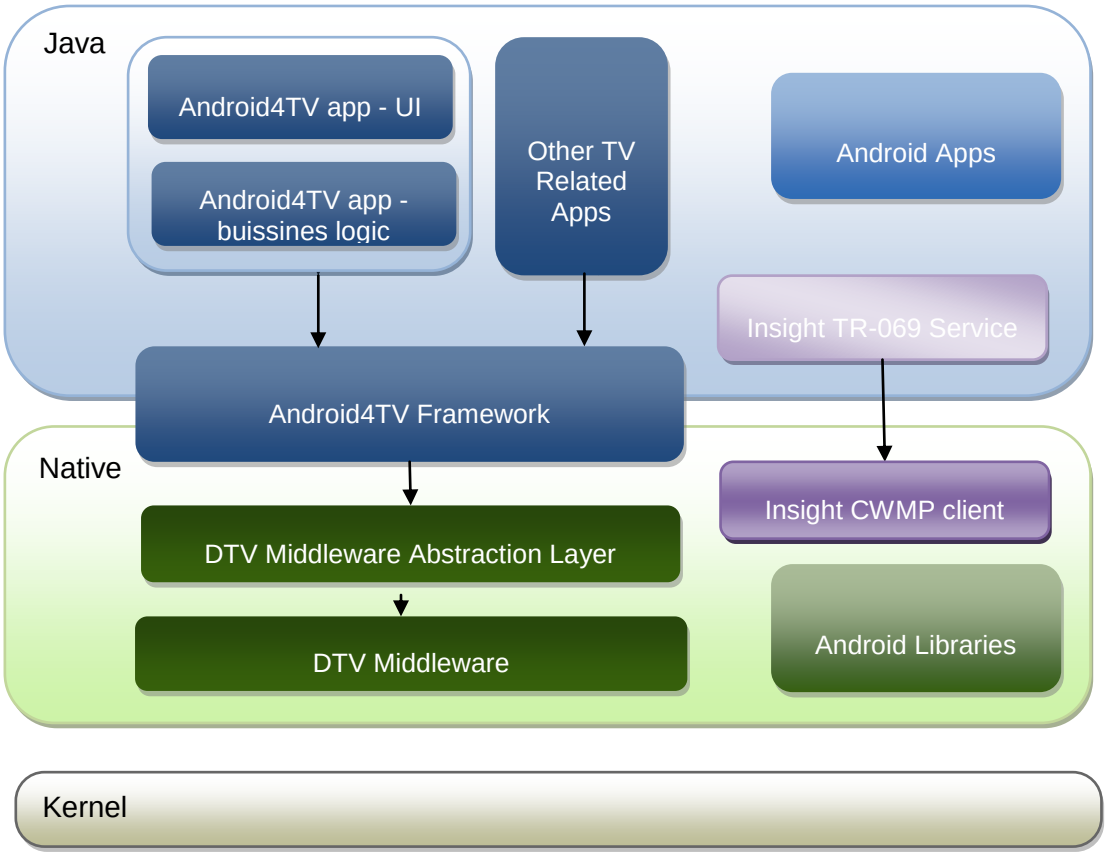

Figure 1: High level Android4TV software architecture

2 ANDROID4TV ARCHITECTURE OVERVIEW

Android4TV is iWedia's approach for the integration of TV within the Android framework. It consists of services, as higher level modules, that provide the ability to build a complete TV system. All services are modular – these are designed to co-exist with any OEM developed services.

Figure 2 shows Android4TV building blocks integrated in standard Android OS. Right side of the picture represents standard Android building blocks (with standard Android applications, managers, native libraries), and left side represents Android4TV building blocks with respect to their position in standard Android OS architecture.

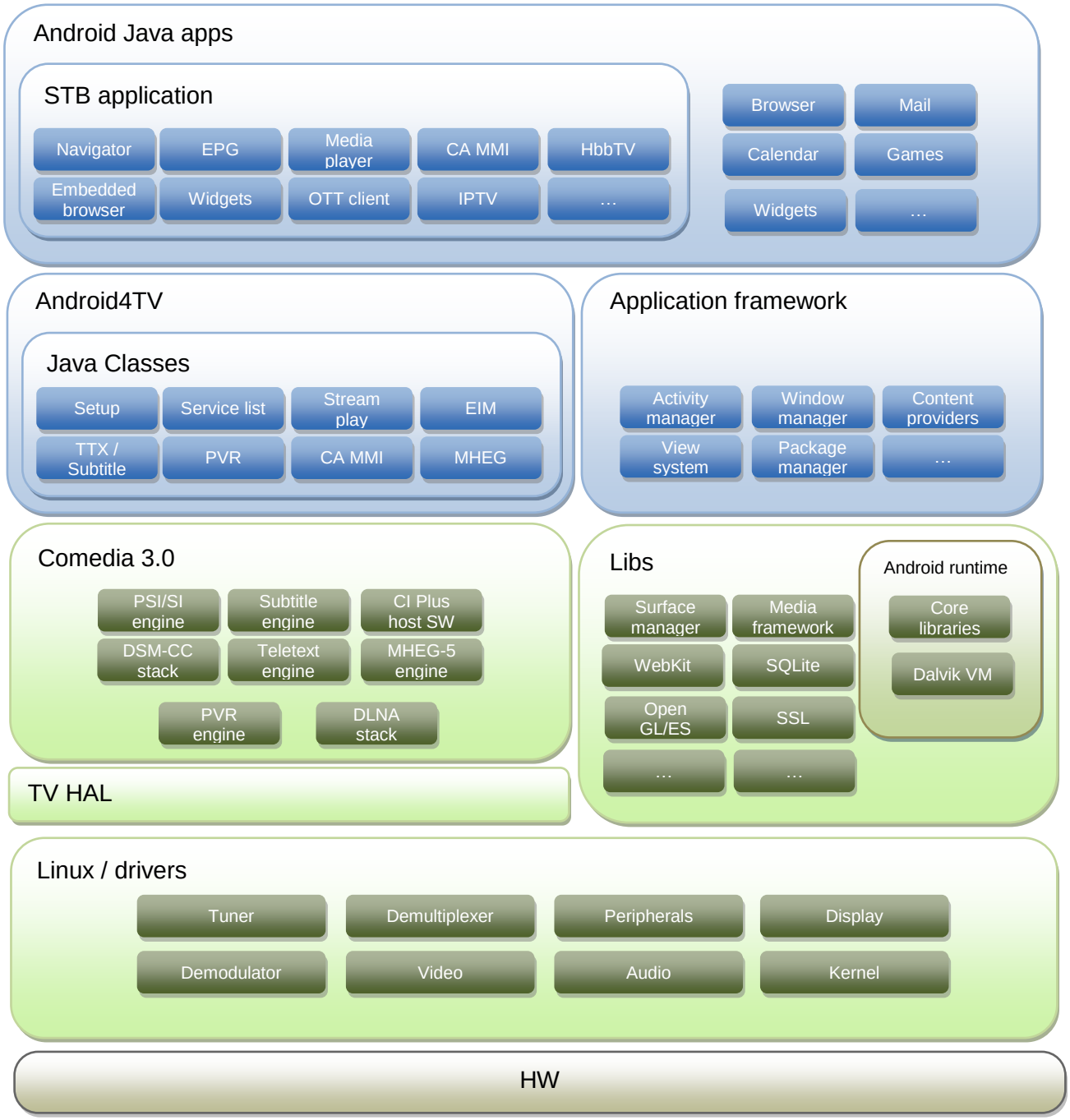

Figure 2: Android4TV block diagram

From top to bottom, following layers can be identified:

- Application layer
- Android4TV Framework layer
- Native layer

Most important modules that compose ANDROID4TV building blocks are also shown on Figure 2. Detailed description of building blocks and relations between modules from which modules are consisted of can be found in Chapter **Error! Reference source not found.**.

2.1 APPLICATION LAYER

Application layer contains Java applications which are written as all other Android applications by all means, using the standard Java Android SDK and Android4TV application framework extension.

Android4TV application is consisted of two processes that communicate between each other and follow client-server model.

Android4TV application that has server role is called business logic. Bussiness logic is Android Java service that is running continuously in the background, even when there is no client application. Most important roles of business logic are:

- Providing asynchronous events
- Maintaining and tracking the life cycle of the client applications

Android4TV application that takes client role is simple UI with only graphic blocks (dialogs, menus etc.) without any unnecessary logic inside. It is a single Android activity. Graphic blocks are enchanced graphic elements from Android application framework in the way that they can be easily customized.

All DTV features are integrated within single DTV application aimed to be used as a tool for demonstration of the framework capabilities. Together with basic DTV features, DTV application contains other features like local/network multimedia playback, network settings etc.

2.2 ANDROID4TV FRAMEWORK

Android4TV framework layer encapsulates DTV features into Java classes, interfaces and enumerations. The main role of this layer is to provide the interface for creating Android applications with DTV features.

Android4TV framework is using underlying native services to provide basic DTV and multimedia playback features. For IPC calls in an Android environment Binder mechanism is used. In this case, native services are acting as “server” for Binder IPC calls from Java.

Android4TV Framework API comprises following set of Java packages:

- com.iwedia.dtv.audio
  - *provides volume control, audio track manipulation and audio decoding events*
- com.iwedia.dtv.ci
  - *provides conditional access routines required for encrypted DTV service playback*
- com.iwedia.dtv.display
  - *provides display layers manipulation, changing z-order, size/position etc.*
- com.iwedia.dtv.dtvmanager
  - *provides the DTV Service implemented as DTVManager, that is available via Binder service*

- com.iwedia.dtv.epg
  - *provides EIT data collection and parsing*
- com.iwedia.dtv.framework.service
  - *provides interface definitions for asynchronous events*
- com.iwedia.dtv.hbbtv
  - *provides HbbTV control routines such as show/hide, load URL, set keys*
- com.iwedia.dtv.mheg
  - *provides MHEG control routines such as show/hide, send key*
- com.iwedia.dtv.parental
  - *provides parental rating setup and control utilities*
- com.iwedia.dtv.picture
  - *provides video post-processing routines (de-interlacing, noise reduction, ..)*
- com.iwedia.dtv.pvr
  - *provides DTV broadcast recording, playback of recorded content and timeshifted playback*
- com.iwedia.dtv.reminder
  - *provides reminder control functionality*
- com.iwedia.dtv.route
  - *provides route manipulation routines, creating route and obtaining device descriptors*
- com.iwedia.dtv.scan
  - *provides service scan setup and control routines for different signal types*
- com.iwedia.dtv.service
  - *provides service and service list control routines*
- com.iwedia.dtv.setup
  - *provides country and language settings*
- com.iwedia.dtv.sound
  - *provides audio post-processing routines (equalizer, audio presets ..)*
- com.iwedia.dtv.streamcomponent
  - *provides broadcast stream components manipulation routines*
- com.iwedia.dtv.subtitle
  - *provides subtitle control functionality*
- com.iwedia.dtv.swupdate
  - *provides system software update control functionality*
- com.iwedia.dtv.systemmanager
  - *provides power management utilities and setting wake up sources*
- com.iwedia.dtv.teletext
  - *provides teletext manipulation utilities*
- com.iwedia.dtv.types
  - *provides basic type definitions used in module*
- com.iwedia.dtv.video
  - *provides video manipulation utilities and video decoding events*

Android4TV Framework provides an environment for different vendors to implement TV-centric applications with features that best suit their needs. There is no need for TV application developers to be aware of the middleware implementation within the lower levels of the software stack.

2.3 NATIVE LAYER

Traditionally, DTV software stack is implemented in native “C” code. Due to the fact that most field-proven software stacks are already implemented in “C” (migration from various embedded systems) and to achieve significant performance gain, an important portion of DTV software stack is implemented in native.

DTV software stack in native consists of following software layers:

- **DTV Middleware Abstraction Layer**

  DTV Middleware Abstraction Layer exposes middleware features to the upper layers. It abstracts the middleware services and provides simpler middleware usage. When integrated in Android environment, this layer needs to be adapted to suit Android Binder IPC.

- **DTV Middleware Layer**

  DTV middleware layer, written in native (C/C++) code, uses Linux drivers for managing DTV hardware. Implementation of this layer is vendor-specific. The main task of middleware module is to hide the complexity of underlying driver layer logic in order to provide simple API which can be used by the upper layer software modules. Beside that, core middleware module provides essential features required for DTV, which are:

  - **PSI/SI engine** – provides simple interface for service scan implementation
  - **PVR engine** – provides broadcast stream recording, recorded content playback and time-shifted playback
  - **Service database** – provides storage (in non-volatile memory) of services found during service scan
  - **Settings database** – provides storage (in non-volatile memory) of various settings such as sound and picture settings, country and language settings and other
  - **OAD** – provides Over-The-Air software update
  - **DSMCC** – provides support for Digital Storage Media Command and Control toolkit

  Beside afore mentioned features provided by core middleware module, most important features provided by middleware module are:

  - Service scan
  - Service playback
  - Teletext
  - Subtitles
  - EPG
  - SSU

- **DTV Hardware Abstraction Layer**

  DTV Hardware Abstraction Layer (HAL) defines the way how DTV middleware interacts with Android and low level code. It defines the interface that middleware layer requires Linux drivers to implement. This layer enables porting of middleware layer (without changes) on any platform that complies with defined interface. DTV HAL interface is expected to provide:

  - Power handling
  - User input capture
  - Display and graphic layout control
  - Storage devices support
  - Transport stream handling
  - Security module
  - DTV interfaces control
  - Content playback control

In Android4TV software solution the DTV application is responsible for the presentation of the video and graphic content, therefore it provides:

- Surface for drawing OSD (Graphics)
- View for live video (Display)

The board support package (BSP) low level drivers will be used to:

- Handle the decoding flow
- Tuner/demodulator control
- DTV inputs/interfaces
- A/V post-processing

Android4TV DTV HAL coexists without interference with existing Android Media Framework as there is no overlapping functionality with one exception to Media Player and AV HAL module as they share the same resources.

3 ANDROID4TV MODULES

Main idea of Android4TV framework is to extend standard Android application framework with DTV capabilities. Android4TV framework consists of basic DTV and extended software modules. Other modules can be integrated in the same way as extended modules.

Android4TV framework consists of software modules that cover basic DTV features:

- Live Broadcast
- IPTV
- EPG
- PVR
- Teletext
- Subtitle

Android4TV extended DTV software modules are:

- Common Interface (CI/CI+)
- HbbTV
- MHEG
- Local media module for multimedia content browsing and reproduction
- DLNA module implements basic DLNA devices: DMP, DMR, and DMS
- Insight client

Implementation of each module is spread through software layers mentioned above. Layers and modules are designed in such way that any modules part from one layer can be replaced without changing of modules parts from other layers.

3.1 ANDROID4TV BASIC DTV FEATURES

3.1.1 Live broadcast

Live broadcast module implements basic TV functionality: audio and video playback for Live TV streams, service scanning, service zapping, service list management, audio and video settings.

Depending on the type of device front-end system, application is able to provide support for DVB-S/S2, DVB-T/T2 or DVB-C/C2 broadcast transmission standards.

Besides support for physicaly built-in tuners the Android4TV solution provides support for reception of satellite delivered content via IP interface (SAT>IP).

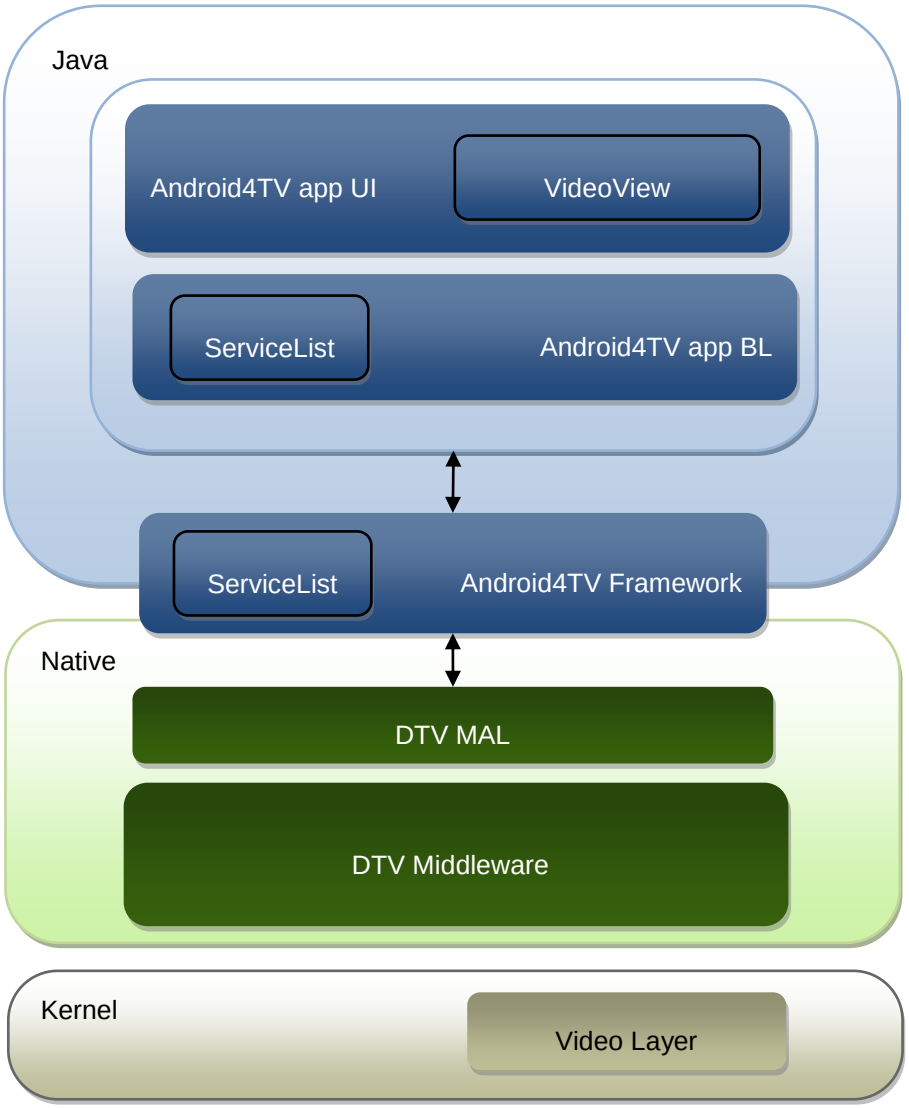

Figure 3: Live broadcast module architecture

The Live broadcast behavior can be controlled by two types of devices:

- Using remote control
- Using other Android device such as mobile phone or tablet

3.1.2 IPTV

Android4TV solution has integrated IPTV support. IPTV support is implemented through virtual IP tuner inside HAL layer and is considered as a separate service type equal to existing live broadcast service types. This implementation simplifies handling of IPTV services in upper layers of middleware and in application layers as they can treat it like any other live broadcast service type. This also means that the existing teletext, subtitle and EPG engines can be used for IP delivered content identically as they are used for other (broadcasted) services.

Virtual tuner is implemented as a set of IP data acquisition protocols that need to conform to a generic protocol API. Protocol core can be implemented as an external (user space) library or as a kernel module. Moreover, virtual tuner can additonaly support technologies like FCC (Fast Channel Change) clients or the modules for decryption of DRM protected content on top of the available protocols.

As a result of such IP tuner architecture, an integration of a new acquisition protocols or DRM decryption module is considerably simplified.

As it was previously mentioned, integration of the IPTV virtual tuner has been done in no way different than the integration of the other, physical ones. Consequently, the way of how IPTV tuner is used by the upper layer is the same except in respect to the scanning and zapping process.

Scanning of the IPTV services can be done within the upper (application) layers using the service list file which contains relevant service information like: service name, service URL, audio and video PIDs. Accordingly, the zapp process can be initiated straight from the application layer by providing the service URL to be zapped in.

The list of the currently supported IP protocols and technologies is provided bellow.

Acqusition protocols:

- RTP, UDP for multicast
- HTTP, RTP, UDP for unicast

Technologies that are pre-integrated with the client SDKs of major Video over IP Content Delivery Networks include:

- VQE for Cisco Videoscape
- FCC/RET – Fast Channel Change/transmission RETry – for Alcatel-Lucent Multiscreen Video Platform
- Pre-integrated with the client SDKs of major Digital Right Management (DRM) systems and Conditional Access Systems (CAS):
- Verimatrix
- Microsoft PlayReady

Technologies that are pre-integrated with the client stacks of major adaptive streaming protocols:

- Microsoft Smooth Streaming
- HLS
- MPEG DASH

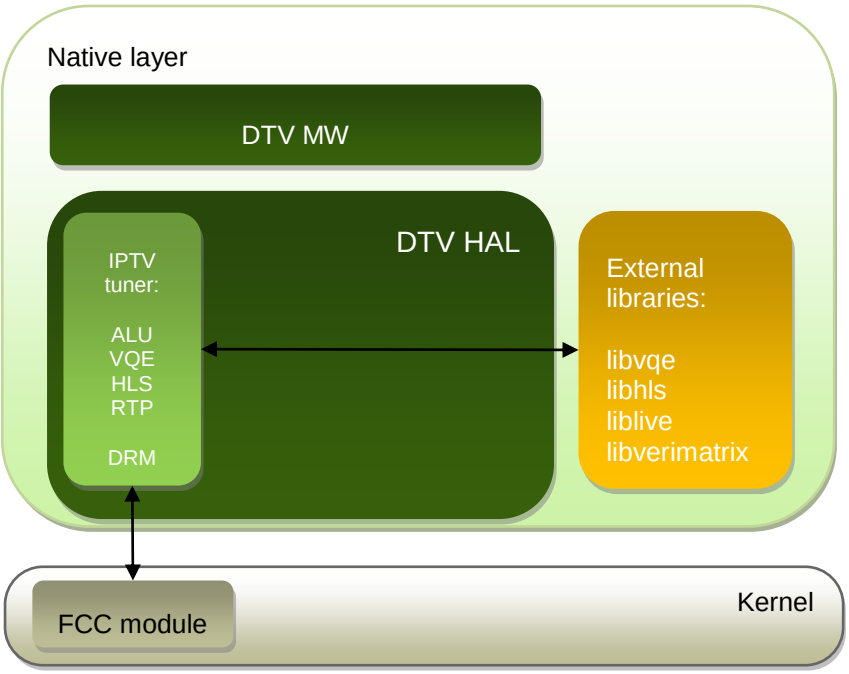

Figure 4 : IPTV tuner module architecture

One of the modules that strongly relies on IPTV support is SAT>IP module. The SAT>IP client engine is implemented within the Android4TV SAT>IP client library named libsipcl. As it is shown on a figure 5, from middleware perspective there will be no differences between the built-in satellite tuner and the SAT>IP library acting as a remote tuner. Also SAT>IP library is providing unique API for all supported IP protocol types, as well as set of helper functions that are reducing effort needed for porting on embedded platforms.

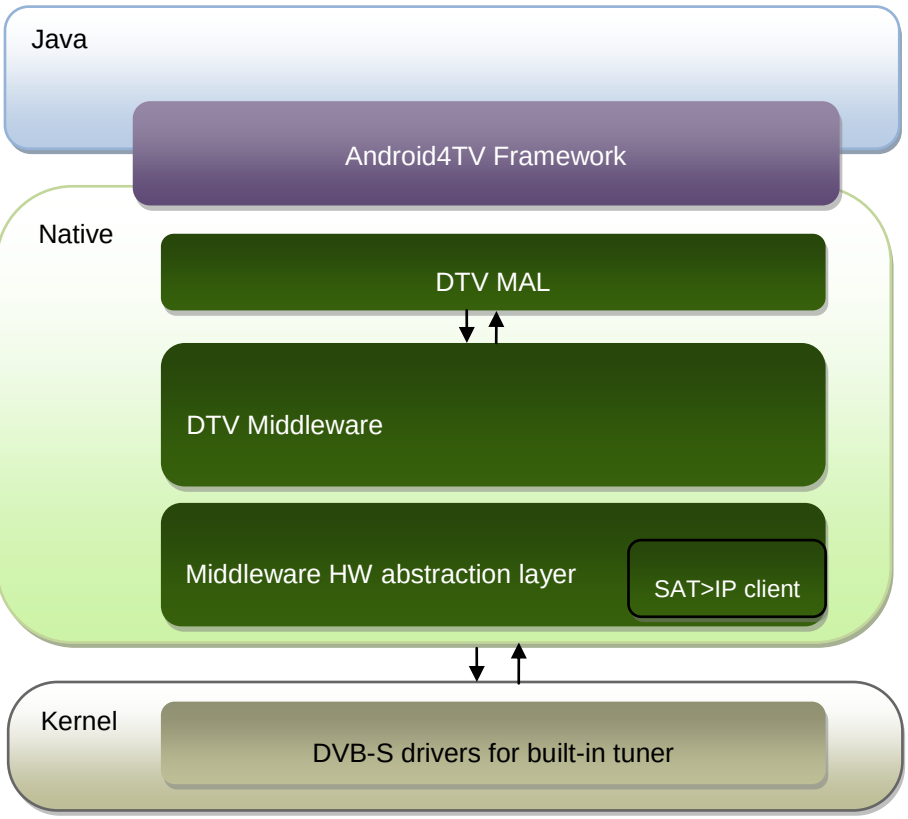

Figure 5 : Live broadcast module architecture with SAT>IP client

3.1.3 EPG

TV Application supports EPG (Electronic Program Guide) functionality. Application is able to extract 7 day EPG data from transport stream and display all extracted data well formatted on the screen. Application is also able to manage scheduled recordings and event reminders of TV content based on information's extracted from EPG.

EPG module implementation can be divided into three functional components: TV Application which provides UI to EPG functionality, EPG package as part of Android4TV framework and EPG engine inside of Comedia engine as part of DTV middleware.

EPG engine which is part of DTV middleware handles extracting of EPG information's by parsing of EIT tables from transport stream. It also handles data requests from upper layer with specified filters and returns well formatted data to upper layer.

TV Application covers following EPG functionality:

- Listing of scheduled events for all services on current frequency as well as listing of events recorded in database during previous channel switching
- Displaying of Now/Next EPG data
- Displaying of events description (event name/description, event start/stop time, parental rating, etc.)
- Possibility to make schedule of events recording

3.1.4 PVR

TV Application has integrated PVR (Personal Video Recorder) module which provides recording of DTV service to a storage device (hard disc, USB drive) as well as reproducing the recorded content (media playback).

PVR module implementation can be divided into three functional components: PVR dialog as part of TV Application, PVR package as part of Android4TV framework and PVR engine which is part of TV middleware.

PVR engine, which is part of TV middleware, hides the complexity of underlying driver logic and it is responsible for routing of transport stream in a PVR context for all supported PVR scenarios. This engine also manages memory storage used for recording of DTV service and handles trick play modes during playback of recoded content.

PVR module supports following features:

- Time Shifting – provides pausing of live DTV service and resume it after some time interval. How long can be pausing time interval depends on available memory space
- Watch & Record – provides recording and watching of the same TV service in the same time
- Playback and Dual Recording – provides watching of one TV service and recording of two TV services in the same time (architecture with two tuners)
- Instant Recording - one key press to start recording
- Scheduled Recording (manual or via EPG) – provides making of schedules for recording of DTV services. Integration of PVR with EPG module enables making of schedules by using of data from EPG.

Recording format is Single Program Transport Stream (SPTS) format – transport stream that contains only one program stream. When PVR playback is requested, recorded transport stream must be routed through demux device that will send elementary streams to A/V decoders.

3.1.5 Teletext module

Android4TV framework implements teletext module which encapsulates teletext functionality and provides teletext API to application.

Teletext module implementation can be divided into following functional components: TV application which provides Java UI to teletext functionality, teletext package as part of Android4TV framework, and teletext engine as part of TV middleware layer.

TV application is responsible for creating of SurfaceView object which has its Surface that will be used for drawing teletext graphics. Teletext graphics drawing is done inside of TV Middleware graphics layer so Surface object is packed into SurfaceBundle object that can be passed through Android4TV framework from TV application to TV middleware.

Android4TV framework display package contains SurfaceBundle and functionalities to set and get SurfaceBundle from Android4TV framework.

Mechanism of extracting of teletext bitmaps from transport stream is provided by teletext engine from TV middleware layer. After teletext bitmap is extracted from the stream, graphics module of the TV middleware layer draws this bitmap on the Surface that is provided through Android4TV framework.

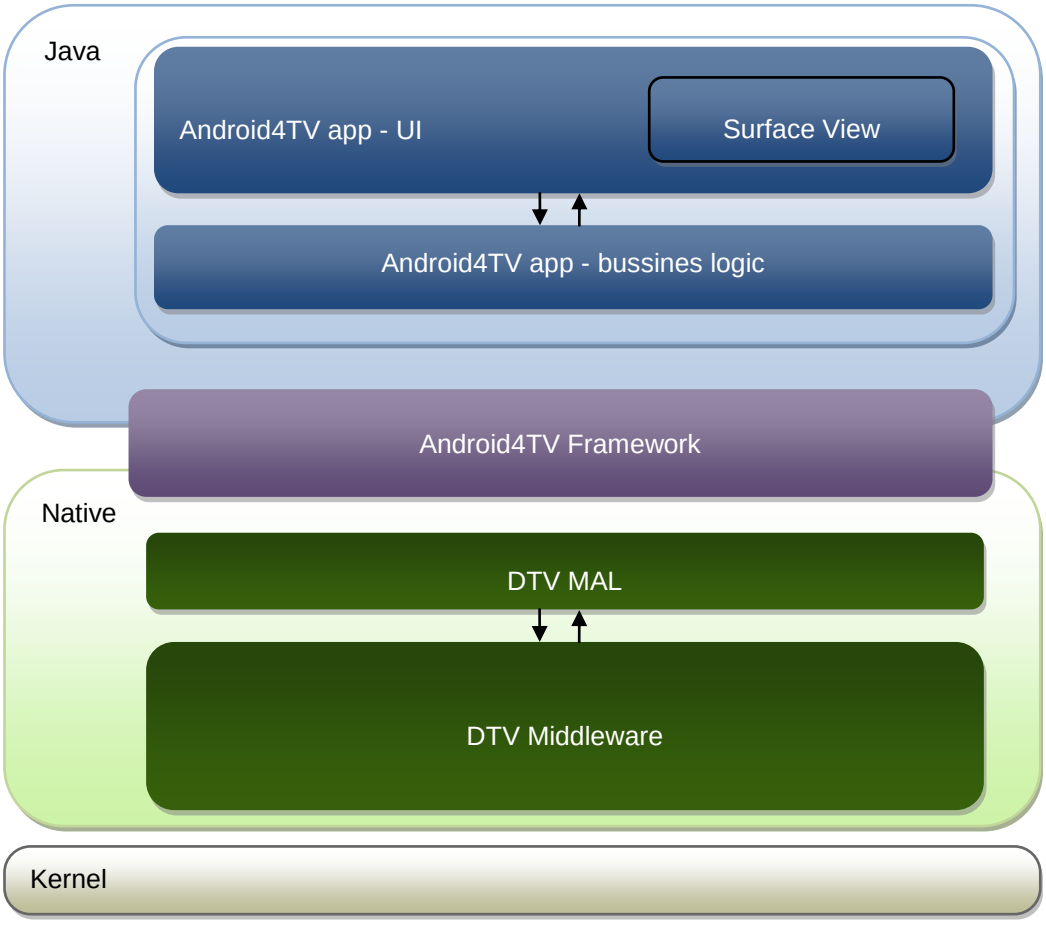

Figure 6: Mechanism of teletext rendering

TV application is able to show teletext information extracted from transport stream. Application extracts teletext data from the stream and makes appropriate bitmaps which are displayed on the screen. In this moment TTX level 1.5 is supported with some features supported from 2.5 version.

TV Application covers following TTX functionality:

- Show/Hide TTX content if available for current service

- Two modes of transparency – displaying of TTX with or without visible live TV stream in background
- Browsing of pages with tabbing navigation through page (by cursor moving)
- FLOF support – browsing of TTX pages by using of four colour buttons (red, green, yellow and blue)

3.1.6 Subtitle

Android4TV framework implements subtitle module which encapsulates subtitle functionality and provides subtitle API to application.

Subtitle module implementation can be divided into following functional components: TV application which provides Java UI to subtitle functionality, subtitle package as part of Android4TV framework, and subtitle engine as part of TV middleware layer.

Application is responsible for creating of SurfaceView object which has its Surface that will be used for drawing subtitle graphics. Subtitle graphics drawing is done inside of TV middleware graphics layer, so Surface object is packed into SurfaceBundle object that can be passed through Android4TV framework from TV application to TV middleware.

Android4TV framework display package contains SurfaceBundle and functionalities to set and get SurfaceBundle from Android4TV framework.

Mechanism of extracting of subtitle bitmaps from transport stream is provided by subtitle engine from TV middleware layer. After subtitle bitmap is extracted from the stream, graphics module of the DTV middleware layer draws this bitmap on the Surface that is provided through Android4TV framework.

TV application is able to show subtitles extracted from transport stream. Application extracts subtitle data from transport stream and makes appropriate subtitle bitmaps which are displayed on the screen.

TV Application covers following subtitle functionality:

- Show/Hide subtitle
- Choosing of available subtitle language

3.2 ANDROID4TV EXTENDED DTV FEATURES

3.2.1 Common Interface (CI)

Android4TV framework has integrated DVB Common Interface (DVB-CI) module for Conditional access functionality. It supports Common Interface (CI) and Common Interface Plus (CI+).

The Common Interface is a part of EN 50221-1997 standard. The Common Interface implements a link between two parts of conditional access functionality:

- **host**: Responsible for tuning to conditional access services and demodulation of RF signal. This is DTV receiver-decoder where module(s) can be connected.
- **module**: The CAM module connects to the host. The module is responsible for CA descrambling.

The host sends an encrypted transport stream to the module. The module decrypts the transport stream and sends it back to the host.

The CI Plus (CI+) extends the CI with additional copy protection between CAM module and the host.

Android4TV framework comes with CI version v1.3.

3.2.2 HbbTV

HbbTV or Hybrid Broadcast Broadband TV is a pan-European initiative aimed at harmonising the broadcast and broadband delivery of entertainment to the end consumer through connected TVs and set-top boxes.

HbbTV module consists of three functional components. On top is web browser (WebView instance placed in a separate dialog which is part of Android4TV application) where HbbTV application is executing. Middle one is HbbTV middleware module (HbbTV engine) and it is using DVB software stack. It is most important module and it is comprised of HbbTV JavaScript plug-ins and HbbTV Application Manager.

DTV Application creates Dialog object that holds instance of WebView object. WebView is class from Android SDK made to allow Java applications to use Webkit browser engine. TV Application provides WebView with HbbTV URL received from HbbTV Application Manager.

In order to expose DVB stack functionality to HbbTV applications, JavaScript plug-ins are used. Plug-ins are implemented using NPAPI (Netscape Plugin Application Programming Interface), which is a cross-platform plug-in architecture used by many web browsers.

The list of implemented JavaScript plug-ins is following:

ApplicationManager plug-in provides currently running application possibility to manage life cycle of applications (start new one or destroy itself), to control visibility of currently running application or to choose which keys it wants to receive

Broadcast plug-in provides interface for integration of content from both broadcast and broadband network into single service. It also provides interface for controlling DTV (service change, access of channel list, selection of AV components which will be rendered). This plug-in communicate with Comedia engine through Application Manager

A/V plug-in provides support for playing and control of streamed multimedia content; this plug-in uses Android Media Framework in order to play audio/video content.

Configuration plug-in provides interface for reading some user preferences such are preferred audio language, preferred subtitle language or country code where DTV is deployed

Capabilities plug-in provides interface for getting information about addition HbbTV capabilities or number of additional HD and SD video decodes available. It is also responsible for send DTV capabilities in XML format to application

ParentalControlManager plug-in provides interface related to parental control system. It contains supported parental rating schemes. Only “dvb-si” scheme shall be supported.

Interfaces of implemented JavaScript plug-ins are defined by [HbbTV] and [DAE] specifications.

Application Manager provides, among other things, communication between web browser and Comedia module. It is responsible for analyzing AIT tables (lifecycle), communication with DSM-CC client etc.

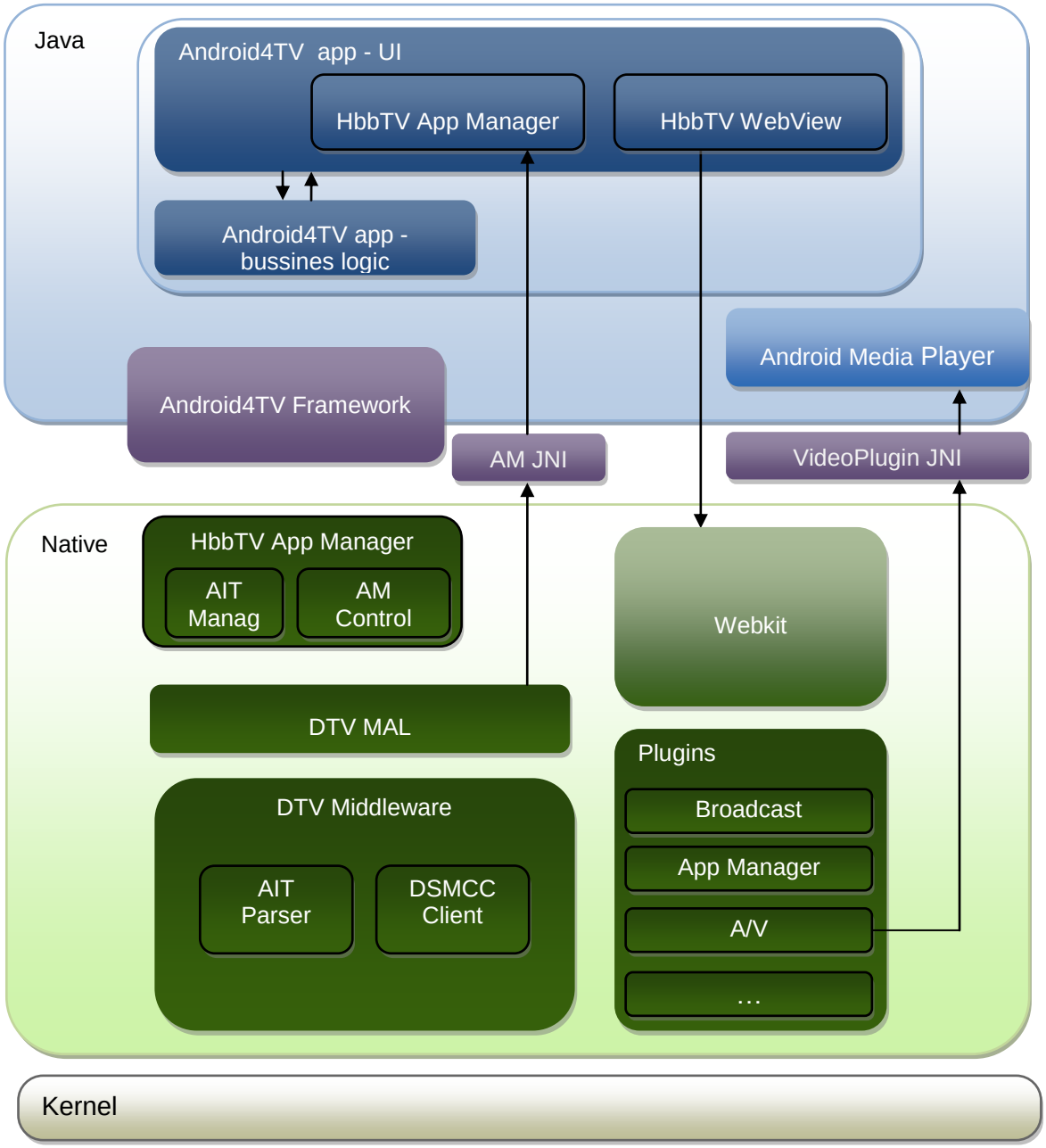

Figure 7: HbbTV module architecture

When DVB stack connects to the new service, AIT manager will acquire and parse AIT table. Upon application descriptor acquisition from AIT, descriptors are delivered to application manager. Application manager selects proper autostart application and send URL to Android4TV application.

After autostart application is loaded, red button is displayed. User can decide to launch different HbbTV application (EPG, teletext etc). Applications can be loaded using http or dvb (DSMCC) protocol.

Since HbbTV module is part of TV Application (as well as all other modules - live broadcast, MHEG, TTX, SUB, Media Browser), screen sharing between all modules is controlled by TV Application and achieved by using of Dialogs. There is a separate Dialog that handles HbbTV functionality as for other modules functionalities such as EPG, TTX or Media Browser.

TV Application supports HbbTV standard for delivery of various services included enhanced teletext, EPG, video-on-demand, interactive advertising, personalization, voting and games.

User of application through HbbTV also has to have possibility to control presentation of broadcast content, control of switching channels, choosing which channel will be presented.

Therefore application should have possibility to control size and position of area used for rendering live TV content. If there are different video, audio or subtitle components being broadcasted (e.g. different languages), user may have possibility to choose between those different components and therefore application should have possibility to control which component will be rendered.

3.2.3 MHEG

MHEG-5, or ISO/IEC 13522-5 is part of a set of international standards relating to the presentation of multimedia information, standardized by the Multimedia and Hypermedia Experts Group (MHEG). The MHEG-5 standard was developed to support the distribution of interactive multimedia applications in client/server architecture across platforms of different types and brands. MHEG-5 defines a final-form representation for application interchange.

MHEG module implementation can be divided on MHEG Dialog as part of TV Application that holds GUI of MHEG application, MHEG package as part of Android4TV framework and MHEG Engine inside of DTV middleware layer.

MHEG Engine is control logic part that interacts with the rest of DTV Middleware. It covers following functions:

- Interprets the MHEG files
- Shows OSD info
- Access table filtering directly from DTV Middleware
- Requests application to start AV playback
- Requests application to process user requests that results in new MHEG file

TV Application has integrated MHEG engine which has ability to display visual objects in a rectangular co-ordinate system with a fixed size, and to play audible objects. A user input device (e.g. remote control or another Android device) can be used with the runtime to allow interaction between the user and the applications. MHEG engine also allows the application to exchange information with a remote server via an IP connection.

The MHEG-5 engine is currently compliant with the D-Book 6.1 specification and the DTG Test Suite. Following features are supported:

- Interactive and broadcast streaming
- True persistent storage
- Persistent storage from RAM
- Interactive channel extension
- Interactive channel encrypted streaming extension

The following optional features are supported:

- Resident program “DBG”
- Cloning of : Link, TokenGroup, ListGroup and Dynamic Lineart

- Extension profile UK 1.06 requirement: All Extension to the UK profile 1.06 features except the unsupported features

Following optional features are NOT supported:

- Ancillary connections
- Free-moving cursor
- Bitmap scaling
- Trick mode
- BIOP_ES_USE association tag
- Text rendering : justified
- Text rendering : line orientation vertical
- Text rendering : start corner other than upper-left

3.2.4 Local Media module

Local media module provides multimedia content browsing and reproducing features. It uses Android Media Framework to implement multimedia reproducing features and FS-LIB Manager to implement multimedia content browsing features.

In order to implement media browsing features local media module communicates with FS-LIB Manager in Java execution environment. FS-LIB manager exposes native FS-LIB API to Java. FS-LIB native engine provides transparent API for local storage browsing.

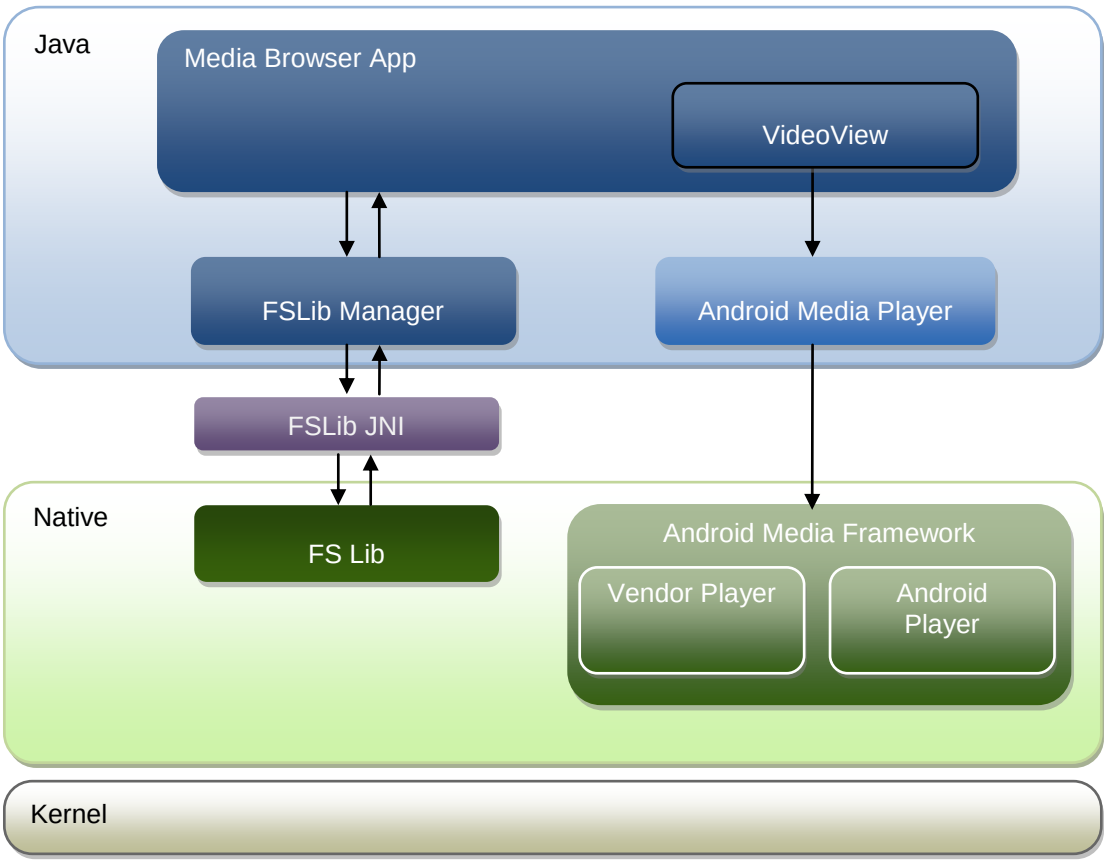

Figure 8: Media Browser module implementation

Local media module supports following features:

- Listing all present folder structure
- Listing content from local storage and mounted USB
- Show meta information's about files
- Playback of local multimedia content (pictures, songs or movies)
- Trick play modes

3.2.5 DLNA

DLNA module implements multimedia content sharing feature between devices according to interoperability guidelines defined by Digital Living Network Alliance. DLNA module covers functionality of following DLNA devices: DMP (Digital Media Player), DMR (Digital Media Renderer), and DMS (Digital Media Server).

DLNA offers following features:

- Displaying all available DLNA servers on local network
- Display names of all shared files during browsing of server content
- Display information about shared files during browsing of server content
- Share directory or file from local content
- Un-share directory or file from local content
- Scan local network for available DLNA servers
- Handle playback requests received from DLNA controllers

DLNA uses Universal Plug and Play (UPnP) for media management, discovery and control. In case when data transferred between DLNA devices on local network have to be secured, DLNA engine will use the DTCP protocol to protect data against unauthorized retransmission and copying. DTCP is a method of protecting audio and audiovisual entertainment content over high-speed, high-bandwidth bidirectional digital interfaces on consumer electronics entertainment and information products. Using DTCP, content can travel between these devices or across a digital home network. In example, when user wants to share recorded TV stream to other DLNA devices, data must be secured with DTCP method.

DLNA player device covers following features:

- Browsing multimedia content from DLNA servers on local network
- Extracting various stream information provided by server: meta information (EXIF, ID3), stream location on network – URL, codec information
- Extracting information about used security protocol if stream is encrypted
- Extracting information about streaming capabilities of DLNA server
- Scanning local network for other DLNA devices

DLNA renderer engine handles playback requests received from DLNA controllers on local network and enables following functionality:

1. Changing device visibility on network
2. Providing media capabilities of device to controller
3. Providing basic state machine for media playback
4. Extracting stream location (URL) and meta information about target stream from controller request
5. Extracting information about used security protocol if stream is encrypted
6. Extracting streaming capabilities of source DLNA server from controller request
7. Providing device audio and video settings to controller
8. Providing state and current position of playback to controller

DLNA server device implements multimedia content sharing to other DLNA devices available on local network. It covers following features:

- Providing list of shared media content to other DLNA devices on network
- Providing streaming of local multimedia content
- Providing various information about shared multimedia content
- Encrypting of stream if necessary and providing security protocol information
- Providing streaming capabilities to other DLNA devices on network
- Parsing streaming requests received from controller or player on network

3.2.6 Insight client

Insight TR-069 service enables the device to be remotely monitored and configured by an Auto-Configuration server (ACS). This service allows ACS to get and set various DTV-related parameters, obtained from the Android4TV Application Framework. Additionally, ACS can initiate a firmware upgrade process on the Android device (implemented using Android recovery console).

Insight CWMP Client is a software block written in C, available for integration to various end-user devices. After the integration of Insight Client, end-user devices become discoverable and accessible by Insight ACS. Insight Client is implemented according to the TR-069 CWMP protocol [1]. Insight CWMP client architecture is shown in Figure 9.

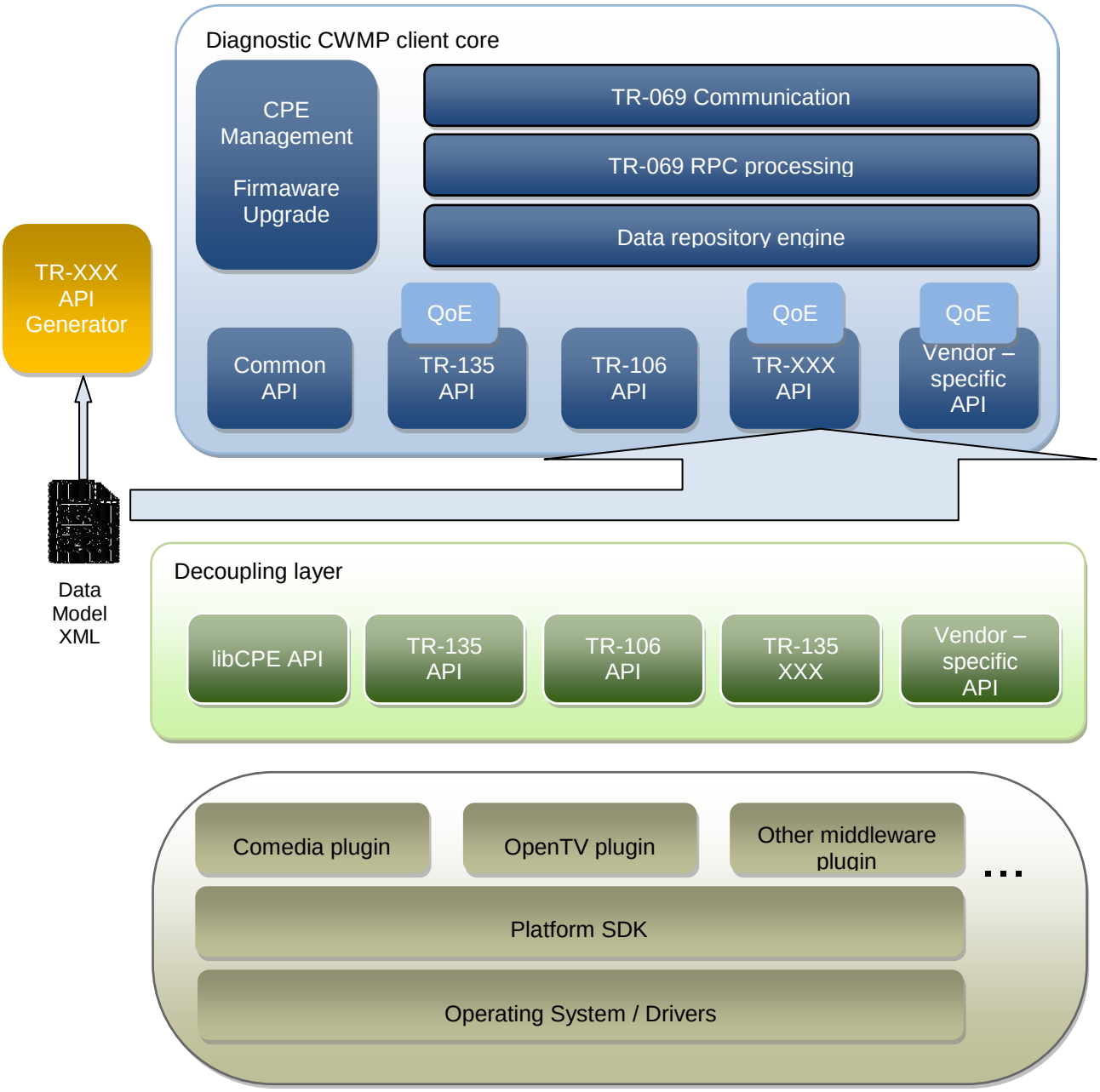

Figure 9: Insight CWMP client architecture

Client is being integrated to the device host software. Host application uses the decoupling layer of the client to invoke all needed functionalities (e.g. to set value of a parameter or to get informed when a value is changed from the ACS). Decoupling layer is provided for a target operating system. Upon function invocation, a message is being posted to the target daemon or service running client core library.

Host application implements an appropriate plugin from which all parameter values are populated and host software updated on request. Client already provides plugins for various target DTV/IPTV middleware, in which case the integration is straightforward. Middleware plugin, once developed, may be reused across different target platforms.

Within the build system of the client, a TR-XXX API generator application is provided. Pre build

script runs this generator by providing XML files specifying data model of the host device. Generator in turns creates appropriate source code files and documentation and automatically extends API of the created core library and the decoupling layer.

Each data model is exposed through a collection of API functions (e.g. TR-135 API, TR-106 API etc). Within this API, host software may register QoE plugins which work with a predefined set of parameters from the data model. Client already provides QoE plugins for several metrics upon TR-135. For example, mean opinion score (MoS) is provided (on the scale 1-5) based on *VideoPerfMon* profile parameters.

The Data repository engine (DRE) acts as a buffer between ACS and the device. Whenever ACS sets a value, it is written to DRE. Host software listens to changes on relevant DRE parameters, and gets informed upon a change through a middleware plugin. Upon each detected change of any value on the STB, host software informs DRE by setting the appropriate value through TR-XXX API.

4 PRODUCT DELIVERABLES

As mentioned before, the integration of Android4TV solution into the Android OS depends on multiple software layers:

- Android4TV based application (i.e. Android4TV_v2.0.apk)
- Android4TV Framework (Android4TV_Framework.apk)
- DTV native layer (libmal.so, libcomedias.so, libchal.so)

Android4TV based application is developed using the Java programming language and the official Android SDK. After being compiled binaries are packaged into *.apk bundle*, the container for the application binary. It contains all of the information necessary to run application on a device, such as compiled .dex files (.class files converted to Dalvik byte code), a binary version of the AndroidManifest.xml file, compiled resources (resources.arsc) and uncompiled resource files for the application.

Android4TV framework as described throughout the document, contains all necesery Java framework extensions required to expose the DTV middleware functionality to the application layer. Its upper layers and API are written in Java programming language, while the lower layer contains native calls to the DTV middeware.

DTV middleware is implemented in native programming language C/C++ and provided within libcomedias.so library.

DTV Hardware Abstraction Layer (HAL) contains the SDK dependent calls and it is provided in the form of libchal.so library.

.

4.1 CHANGES TO THE ANDROID BASE SOURCE

To support HbbTV solution some small amount of changes had to be introduced to the standard Android base source code.

The HbbTV platform combines a profile of the Open IPTV Forum specification with a profile of the DVB specification for signalling and carriage of interactive applications and services in Hybrid Broadcast/Broadband environments. To see full description of HbbTV please see reference TBD.

HbbTV browser platform is a WebKit-based consisting of WebCore and JavaScriptCore parts. To support HbbTV solution some changes to WebCore library (*libwebcore.so)* had to be introduced (Table 2).

WebCore
CSS3 navigation properties
Adding new DOM objects like Channel, KeySet and KeyCode
Mapping of additional colored RC's keys
Modification of SpationNavigation module

CE-HTML
OIPF MIME types
plug-in objects lifecycle
Android focus fix

Table 2: Changes to WebCore library

WebView class as part of the Android Framework is also modified to support mapping of additional colored RC's keys in WebCore library (Table 3).

Android Framework
Webview class

Table 3: Changes to WebView class

4.1.1 Changes to WebCore library

Detail list of changes introduced to WebCore librarary:

- CSS3 navigation properties: nav-left, nav-right, nav-up, nav-down
- Listed properties are part of the CSS3 specification but are not fully implemented in available WebCore builds. Web engine should be able to recognize, parse and navigate on web pages that utilize them.
- New DOM objects:
    - Channel: supports different types of channels
    - KeySet: support key handling between TV and HbbTV application (used by application manager plug-in)
    - KeyCodes: define constants according to CEA 2014 A (CE-HTML) standard

These are the global visible objects from JavaScript.

- Mapping of additional colored RC's keys
- Colored RC's keys (red, green, blue and yellow) are specific to CE-HTML/HbbTV and must be mapped to appropriate Webkit keycodes
- Modification of SpationNavigation module: To support HbbTV application this module is changed in the part that controls focus change. Existing logic/algorithm is changed in the case when two or more elements on the pages are overlapped.
- CE-HTML MIME type support: it supports parsing and rendering of CE-HTML pages (.cehtml extension)
- OIPF MIME types: changes to overcome the problem when Webkit filters MIME types in such a way to down case all the letters in registered plug-ins. MIME types should support capital letters.

- Plug-in objects lifecycle: it solves the issue when Webkit destroys every objects/plugin that is specified as invisible in the CSS.
- Android Focus Fix: Android's current implementation of external focus must be disabled in the case of active HbbTV application.