GY

中华人民共和国广播电影电视行业标准

GY/T 303.1—2016

智能电视操作系统
第1部分：功能与架构

Smart TV operating system—
Part 1: Function and architecture

2016－12－12 发布　　2016－12－12 实施

国家新闻出版广电总局　　发布

目　　次

前　言

GY/T 303《智能电视操作系统》计划发布以下部分：

——第1部分：功能与架构；

——第2部分：安全；

——第3部分：应用编程接口；

——第4部分：硬件抽象接口；

——第5部分：功能组件接口；

——第6部分：可信执行环境接口；

——第7部分：符合性测试。

本部分为GY/T 303的第1部分。

本部分按照GB/T 1.1—2009给出的规则起草。

本部分由全国广播电影电视标准化技术委员会（SAC/TC 239）归口。

本部分起草单位：国家新闻出版广电总局广播科学研究院、华为技术有限公司、中兴通讯股份有限公司、东方有线网络有限公司、深圳创维-RGB电子有限公司、上海联彤网络通讯技术有限公司、阿里云计算有限公司、深圳市茁壮网络股份有限公司、四川长虹网络科技有限责任公司、四川九州电子科技股份有限公司、创维数字技术股份有限公司、深圳市海思半导体有限公司、中国科学院声学研究所、江苏省广电有线信息网络股份有线公司、青岛海信电器股份有限公司、中国科学院信息工程研究所、陕西广电网络传媒（集团）股份有限公司、湖南省有线电视网络（集团）股份有限公司、上海下一代广播电视网应用实验室有限公司、中国科学院软件研究所、北京数码视讯科技股份有限公司、北京永新视博数字电视技术有限公司、北京数字太和科技有限责任公司、上海兆芯集成电路有限公司、晨星软件研发（深圳）有限公司、未来电视有限公司、乐视致新电子科技（天津）有限公司、江苏银河电子股份有限公司、腾讯科技(深圳)有限公司、北京优朋普乐科技有限公司、康佳集团股份有限公司、华数数字电视传媒集团有限公司、山东广电网络有限公司、国家新闻出版广电总局卫星直播管理中心、国广东方网络（北京）有限公司、浪潮集团有限公司、深圳市同洲电子股份有限公司、一九零五互动（北京）科技有限公司、湖南国科微电子股份有限公司、北京海尔集成电路设计有限公司、杭州国芯科技股份有限公司、上海高清数字科技产业有限公司、北京泰合志远科技有限公司、上海英立视数字科技有限公司、北京赛科世纪数码科技有限公司、上海全景数字技术有限公司、上海仪电数字技术有限公司、环球智达科技（北京）有限公司。

本部分主要起草人：盛志凡、朱佩江、陈德林、万乾荣、杨明磊、王继刚、王志国、刘金晓、赵学庆、贾汇东、徐佳宏、杜武平、解伟、王劲林、王明敏、林宝成、杨战兵、熊智辉、付强、严海峰、郭沛宇、同磊、胡益锋、何剑、林远大、袁宏伟、陈亚东、咎元宝、蒋艳山、程伯钦、王之奎、黎政、胡力旗、周芸、孙明勇、黄滔、江四红、何毅进、王磊、赵良福、汤新坤、贾庭兰、付瑞、张定京、马万铮、郭万永、占亿民、徐其桓、蒋新农、张伟、谢振雷、游昌海、裘洪国、孙明松、张震宁、万倩、张晶、王兴军、王佳敏、丁送星、邓泽学、陈家东、叶建隆、叶建荣、张伟明、唐亮、陶春、黄永刚、管丹东、李玮帆、董进刚、曹松涛、钟其元、王欣刚、黄新军、来永胜、王旭升、郭晓霞、冯伟、白伟、孙鹏、熊彬、郑力铮、张雷鸣、吴超、孟庆康、朱允斌、梁志坚、吉峰、仝永辉、姜峰云、施玉海、陈宝霞、朱哲田、朱里越、黑维炜、袁炜、刘新伟、宋晓波、方中华、吴迪、李旭东、苍鹏、刘小卫、吴坚、赵凌、李迎新、白海丽、郑勇、杜晓康、冉大为、陈建、彭卫、张帆、刘刚、魏启任、郭金花、蒲

佳、张清山、南习清、冯浩栘、邓勇、杨启程、李波、郭永伟、于龙朕、陈烨、李秉义、郭志川、杨波涛、窦旻、郗望、刘严鹏、谢天、孙健、白龙、樊义飞、毛帅、赵博文、叶丰、金鼎国、庄珩、毕晶棪、周泽钦、尹承辉、姚世宏、付晶、张剑、刘锦阳、沈丛林、肖辉、冒海波、沈飞、王永乐、姚辉军、高杰、胡波、徐海燕、刘春梅、廖冯军、谢长弘、杨利中、邹书强、张明、李义才、李威青、李婷婷、肖红江、王雅哲、王瑜、李斌、林品廷、黄石华、张金、任立学、李金库、王锋、周元元、彭鹏、沈辉、邱波、潘岩、刘荣军、敖钧、祁涛、倪志斌、高志扬、郝丹、叶睿睿、王征霞、刘勇、张志洋、边详国、姚磊、彭召旺、颉小龙、莫玲生、顿西峰、梁智邦、田明。

引　言

本部分的发布机构提请注意，声明符合本部分时，可能使用涉及本部分有关内容的相关授权的和正在申请的专利如下：

序号	标准章条号	专利名称
1	5、9、10	一种智能电视操作系统
2	5、9、10、附录A	一种智能电视系统
3	8.4	一种在智能电视操作系统中支持全媒体播放的方法及智能电视终端
4	8.6	一种用于智能操作系统的数字版权管理（DRM）方法和系统
5	8.6	一种支持数字版权管理（DRM）的媒体网关/终端实现方法及其设备
6	8.7	一种用于智能操作系统的条件接收方法和系统
7	8.7	一种用于智能操作系统的条件接收方法和系统

本部分的发布机构对于该专利的真实性、有效性和范围无任何立场。

该专利持有人已向本部分的发布机构保证，他愿意同任何申请人在合理且无歧视的条款和条件下，就专利授权许可进行谈判。该专利持有人的声明已在本部分的发布机构备案，相关信息可以通过以下联系方式获得：

专利权利人	联系地址	联系人	邮政编码	电话	电子邮箱
国家新闻出版广电总局广播科学研究院	北京市西城区复兴门外大街2号	孟祥昆	100866	010-86098010	mengxiangkun@abs.ac.cn

请注意除上述专利外，本部分的某些内容仍可能涉及专利。本部分的发布机构不承担识别这些专利的责任。

智能电视操作系统
第1部分：功能与架构

1 范围

GY/T 303的本部分规定了智能电视操作系统的功能及架构相关技术要求。

本部分适用于智能电视操作系统的研发、生产、测试和应用。

2 规范性引用文件

下列文件对于本部分的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本部分。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本部分。

GB/T 17975.1—2010 信息技术 运动图像及其伴音信息的通用编码 第1部分：系统

GB/T 22726—2008 多声道数字音频编解码技术规范

GB/T 28160—2011 数字电视广播电子节目指南规范

GB/T 28161—2011 数字电视广播业务信息规范

GY/T 255—2012 可下载条件接收系统规范

GY/T 257.1—2012 广播电视先进音视频编解码 第1部分：视频

GY/T 258—2012 下一代广播电视网（NGB）视频点播系统技术规范

GY/T 267—2012 下-代广播电视网（NGB）终端中间件技术规范

ECMA-262 ECMAScript语言规范（ECMAScript Language Specification）

TR069 CPE广域网管理协议

http://www.w3.org/TR/html5/

https://www.w3.org/standards/techs/css#stds

https://www.w3.org/TR/DOM-Level-2-HTML/

3 术语、定义和缩略语

3.1 术语和定义

下列术语和定义适用于本部分。

3.1.1

智能电视操作系统 television operating system; TVOS

运行在电视接收终端等终端之上，具备管理系统资源（包括硬件、软件及数据资源）、控制程序执行、支撑应用软件运行等功能的系统软件。

3.1.2

智能电视操作系统双平台软件版本 TVOS-C

能够同时支持Java应用和Web应用的智能电视操作系统软件。

3.1.3

智能电视操作系统单平台软件版本　TVOS-H

仅支持Web应用的智能电视操作系统软件。

3.1.4

进程间通信机制　Binder mechanism

一种通过内核驱动实现客户端和服务端进程间通信的机制。

3.2　缩略语

下列缩略语适用于本部分。

AAC　高级音频编码（Advanced Audio Coding）
AC3　音频编码3（Audio Coding3）
API　应用程序编程接口（Application Programming Interface）
App　应用程序（Application）
ATV　模拟电视（Analog Television）
AV　音视频（Audio Video）
BAT　业务群关联表（Bouquet Association Table）
CA　证书认证机构（Certification Authority）
CDC　互联设备配置（Connected Device Configuration）
CSS　样式级联表（Cascading Style Sheets）
DASH　基于HTTP的动态自适应流（Dynamic Adaptive Streaming over HTTP）
DAVIC　国际数字音频/视频委员会（Digital Audio/Video International Council）
DCAS　可下载条件接收系统（Downloadable Conditional Access System）
DHCP　动态主机配置协议（Dynamic Host Configuration Protocol）
DLNA　数字生活网络联盟（Digital Living Network Alliance）
DOM　文档对象模型（Document Object Model）
DRM　数字版权管理（Digital Rights Management）
DT　设备树（Device Tree）
DTH　卫星直播广播电视（Direct To Home）
DTS　设备树源（Device Tree Source）
DTV　数字电视（Digital Television）
DVB　数字视频广播（Digital Video Broadcasting）
ECEK　加密内容密钥（Encryption Content Secret Key）
ECM　授权控制信息（Entitlement Control Message）
EIT　事件信息表（Event Information Table）
EMM　授权管理信息（Entitlement Management Message）
EPG　电子节目指南（Electronic Program Guide ）
ES　基本码流（Elementary Stream）
FP　基础概要文件（Foundation Profile）
HAL　硬件抽象层（Hardware Abstract Layer）
HCI　人机交互（Human-Computer Interaction）

HDCP 高带宽数字内容保护技术（High -bandwidth Digital Content Protection）
HLS Apple的动态码率自适应技术（HTTP Live Streaming）
HTML 超文本标记语言（Hyper Text Markup Language）
HTTP 超文本传输协议（Hyper Text Transfer Protocol）
IPC 进程间通信（Inter-Process Communication）
IPTV IP电视（IP Television）
JNI Java 本机接口（Java Native Interface）
JS Java脚本语言（Java Script）
MPEG 动态图像专家组（Moving Picture Experts Group）
NGB-H 基于HTML的下一代广播电视网中间件（Next Generation Broadcasting Network-HTML）
NGB-J 基于Java的下一代广播电视网中间件（Next Generation Broadcasting Network-Java）
NIT 网络信息表（Network Information Table）
NVM 非易失性存储器（NonVolatile Memory）
OS 操作系统（Operating System）
OSD 屏幕菜单式调节方式（On-Screen Display）
OTA 空中升级（Over The Air）
OTT 基于开放互联网的视频服务（Over The Top）
PAT 节目关联表（Program Association Table）
PBP 个人基础配置文件（Personal Basis Profile）
PID 包识别码（Packet Identifier）
PMT 节目映射表（Program Map Table）
PP 个人配置文件（Personal Profile）
PPV 每收视一次付费（pay per view）
PSI 节目特定信息（Program Specific Information）
RAM 随机存取存储器（Ramdom Access Memory）
REE 富执行环境（Rich Execution Environment）
SDK 软件开发工具包（Software Development Kit）
SDT 业务描述表（Service Descriptor Table）
SI 业务信息（Service Information）
TApp 可信应用（trust application）
TEE 可信执行环境（Trusted execution environment）
TS 传送流（Transport Stream）
TVM TV虚拟机（TV Virtual Machine）
UPNP 通用即插即用（Universal Plug and Play）
URL 统一资源定位符（Uniform Resource Locator）
UUID 通用唯一识别码（Universally Unique Identifier）
VOD 视频点播（Video On Demand）
XML 可扩展标记语言（Extensible Markup Language）

4 总体要求

4.1 系统功能要求

4.1.1 数字电视直播要求

支持遵循 GB/T 17975.1—2010 和 GB/T 28161—2011 的数字电视直播节目播放。

4.1.2 视频点播要求

支持遵循 GY/T 258—2012 的视频点播节目播放。

4.1.3 互联网电视要求

支持对互联网电视集成播控平台播发的互联网电视节目播放。

4.1.4 本地媒体播放要求

支持智能电视终端本地存储介质内的媒体文件播放。

4.1.5 媒体处理

媒体处理要求如下：

a) 支持数字电视直播、数字电视点播、互联网电视、IPTV、本地视音频和跨屏视音频等不同媒体形态的播放处理；
b) 支持MPEG2、MPEG4、AVS（GY/T 257.1—2012）、AVS+、AVS2、H.264和H.265等视频格式的解码；
c) 支持MPEG Audio LayerII和LayerIII、AAC、AC3和DRA（GB/T 22726—2008）等音频格式的解码；
d) 支持HTTP、HLS和RTSP等流媒体协议的解析和处理；
e) 支持MP4、MKV和AVI等流媒体文件格式的解析和处理；
f) 支持基于ChinaDRM内容保护的加密媒体文件的播放；
g) 支持基于GY/T 255—2012的加密数字电视节目流的播放。

4.1.6 EPG

支持遵循GB/T 28160—2011的EPG节目信息的解析和呈现。

4.1.7 多屏互动支持要求

支持遵循 DLNA 协议的多屏互动功能。

4.1.8 智能家居

智能家居组件应实现对智能家居设备发现、连接建立和操控的管理。

4.1.9 终端管控

终端管控组件应实现对智能电视终端信息和参数的查询、统计、设置、监控和上报等功能，包括恢复出厂设置、终端重启设置、软件升级触发、网络诊断触发等。

4.1.10 数据采集

支持智能电视终端状态、业务应用和用户行为等信息的数据采集和上报功能。

4.1.11 应用软件支持要求

智能电视操作系统双平台软件版本（TVOS-C）支持 Java 应用和 Web 应用，包括基于 HTML5 的 Web 应用。

智能电视操作系统单平台软件版本（TVOS-H）支持 Web 应用，包括基于 HTML5 的 Web 应用。

4.1.12 可升级支持要求

支持通过广播传输通道和宽带 IP 传输通道进行远程系统安全升级，支持本地系统安全升级。

4.2 系统架构要求

应符合第5章所定义的软件架构以及TVOS-C或TVOS-H软件平台要求。

4.3 软件代码树要求

TVOS代码采用层级的目录管理方式，一级目录包括应用程序软件代码目录、功能接口单元软件代码目录、核心功能组件软件代码目录、硬件平台相关软件代码目录、内核软件代码目录、平台特有软件代码目录等；

TVOS-C软件代码和TVOS-H软件代码应置于同一TVOS代码树下，其中，TVOS-C软件代码和TVOS-H软件代码中不重用的软件代码放置于平台特有软件代码目录下对应的子目录中，TVOS-C软件代码和TVOS-H软件代码中重用的软件代码按照代码树目录分类要求分别放置于对应的目录下。TVOS代码树参见附录A。

4.4 系统接口要求

系统接口包括系统应用接口、功能组件调用接口和硬件适配调用接口。

系统应用接口包括Java应用接口和Web应用接口；Java应用接口应遵循GY/T 267—2012中NGB-J相关应用接口要求，兼容Android API相关要求；Web应用接口应遵循GY/T 267—2012中NGB-H接口要求和HTML5相关应用接口要求。

核心功能组件调用接口应能既支持Java应用功能接口单元又支持Web应用功能接口单元的调用。

系统硬件适配调用接口应屏蔽底层不同硬件的差异，支持功能组件通过统一的接口对不同硬件的功能调用。

4.5 系统安全要求

应符合智能电视操作系统安全的要求。

4.6 TVOS 对硬件配置的基本要求

对于TVOS-C软件，智能电视终端的闪存等NVM存储容量配置不低于1GB，RAM存储容量配置不低于1GB，CPU双核以上，频率不低于1GHz。

对于TVOS-H软件，智能电视终端的闪存等NVM存储容量配置不低于256MB，RAM存储容量配置不低于512MB，CPU频率不低于600MHz。

4.7 性能要求

4.7.1 开机时间要求

对于加载TVOS-C的系统，在基本的软硬件配置下，从加电开机到出现第一个开机画面的时间不大于5s。从加电到出现正常图像和伴音的时间不超过50s。

对于加载TVOS-H的系统，在基本的软硬件配置下，从加电开机到出现第一个开机画面的时间不大于5s。从加电到出现正常图像和伴音的时间不超过45s。

4.7.2 直播频道切换时间要求

对于加载TVOS-C的系统，在基本的软硬件配置下，高清节目频道之间相互切换时间不大于2s。

对于加载TVOS-H的系统，在基本的软硬件配置下，高清节目频道之间相互切换时间不大于2s。

5 功能软件架构

智能电视操作系统TVOS由REE部分和TEE部分组成。

TVOS REE部分应采用层次化、模块化软件架构，由内核、硬件抽象（HAL）、功能组件、执行环境、应用框架等5个功能软件层以松耦合方式构建，各功能软件层由多个软件模块以松耦合方式构成。

TVOS TEE部分由Secure OS、TEE HAL和Trusted App构成。TVOS软件功能架构如图1所示。

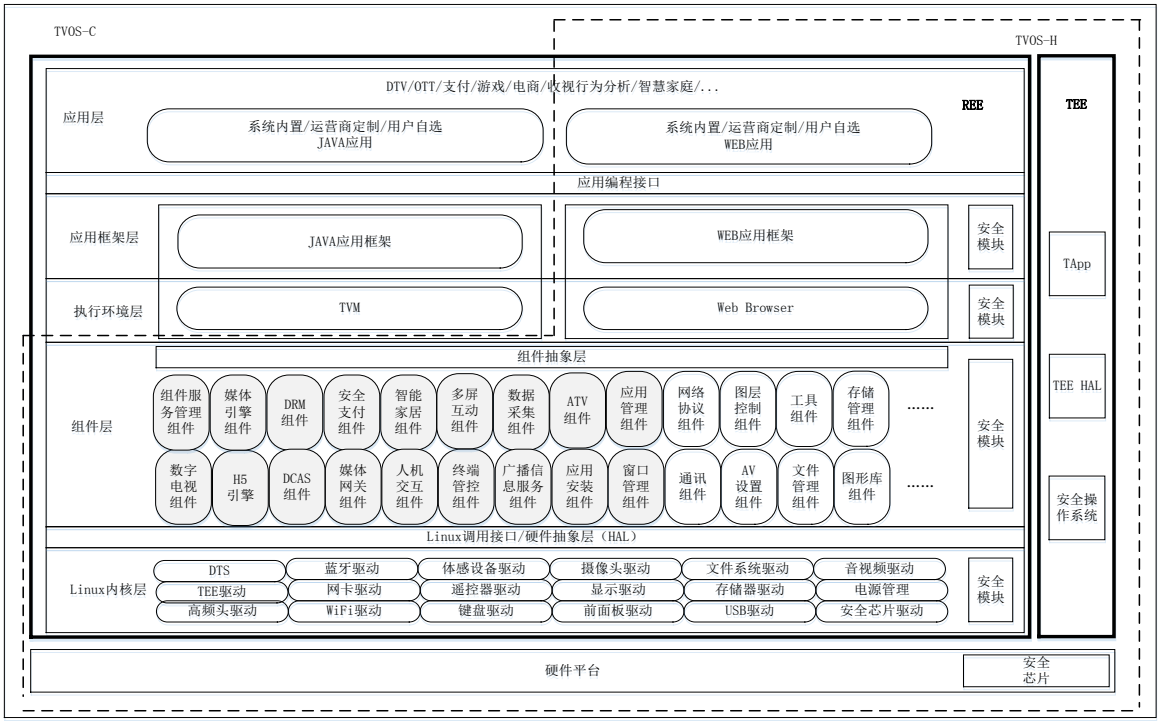


图1 TVOS 软件功能架构

TVOS 内核层应实现基础操作系统功能，包括进程调度、内存管理、虚拟文件系统、网络协议栈、进程间通讯、安全策略和硬件驱动等系统资源的抽象、管理和分配功能，为上层软件提供基础操作系统服务。

TVOS 硬件抽象层（HAL）应实现对 TVOS 硬件平台能力的抽象封装，对同一类型硬件设备采用统一的抽象封装模型，为上层软件对硬件平台能力的访问和控制提供统一的调用接口。

TVOS 功能组件层应实现智能电视操作系统核心功能，为各类应用提供公共服务能力支撑；应包括媒体处理、数字电视、DRM、DCAS、安全支付、智能家居、人机交互、终端管控、应用管理、窗口管理等共用功能组件模块；各共同功能组件模块应采用客户端-服务端模式实现，其中，服务端和客户端运行在不同的进程空间，且使用相同的进程间通信机制实现跨进程通信，服务端负责实现相应组件功能并通过硬件抽象层调用内核层软件模块和底层硬件；共用功能组件模块应同时支持 JAVA 应用和 WEB 应用。

TVOS 执行环境层应实现应用软件和应用适配软件的解释执行环境，支撑 JAVA 应用和 WEB 应用的加载和运行，JAVA 应用执行环境为 TVM，WEB 应用的执行环境为 Web Runtime。

TVOS 应用框架层应实现 JAVA 应用和 WEB 应用与功能组件模块的接口封装适配，JAVA 应用框架包括 NGB-J 功能接口单元和兼容其他 JAVA 应用的接口单元，WEB 应用框架包括 NGB-H 功能接口单元和 HTML5 功

能接口单元。

TVOS 应通过共用内核层和 HAL 层软件，以共用功能组件模块为基础，采用添加其他功能组件模块、拼接或裁剪 JAVA 应用框架和 WEB 应用框架的方式,构建 TVOS-C 或 TVOS-H 平台。TVOS-C 平台应同时支持 Java 应用和 WEB 应用，TVOS-H 平台应仅支持 WEB 应用。

6 内核层

TVOS 内核层应包括 Linux kernel、DTS 和硬件驱动三部分软件。

Linux kernel 应实现进程调度、内存管理、虚拟文件系统、网络协议栈、I/O 管理、进程间通讯和安全保护等基础操作系统功能，与安全芯片协同支撑实现基于硬件安全信任根中的安全信任链校验机制。

DTS 软件模块应基于 DTS 机制实现 Linux kernel 与硬件驱动的解耦。

硬件驱动软件模块应包括各类通用硬件驱动、数字电视相关硬件驱动和安全相关硬件驱动等，支撑TVOS内核对各类硬件设备的通信、访问和管理。

7 硬件抽象层

TVOS HAL层应由多个硬件抽象功能接口模块组成，不同的硬件抽象功能接口模块实现对不同硬件能力及其操控的抽象封装，并为上层软件提供调用相应硬件能力的接口。

各硬件抽象功能接口模块应采用Stub硬件抽象模型实现。Stub硬件抽象模型将一个硬件模块和若干硬件设备以及对它们的操作方法以Stub操作函数的形式，通过将硬件模块ID对应相应的Stub操作函数指针的方式，为上层软件提供相关硬件能力的调用方法，实现对相关硬件能力的操作和控制。

TVOS HAL层Stub硬件抽象模型原理如图2所示。

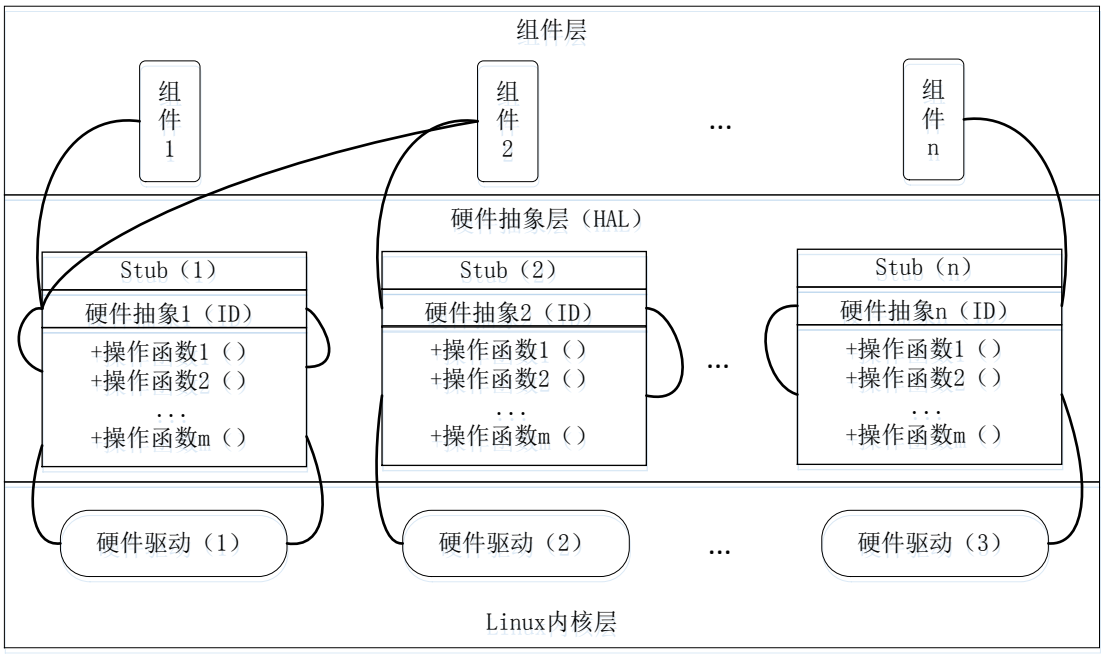


图 2 TVOS HAL 层 Stub 硬件抽象模型原理

TVOS HAL 功能接口模块应包括媒体处理专用硬件 HAL 功能接口模块和通用硬件 HAL 功能接口模块等两

大类。

媒体处理专用硬件 HAL 功能接口属于硬件抽象接口。

8 组件层

8.1 组件模型

TVOS 组件应由服务端和客户端组成，服务端和客户端运行在不同的进程空间，且使用 Binder 机制实现跨进程通信。服务端负责实现相应组件功能并通过硬件抽象层调用内核层软件模块和底层硬件；组件服务端主要包括服务实现和服务 Stub 等软件模块；组件服务端是一个系统常驻的运行实例，一个组件服务端运行实例服务多个不同的组件客户端运行实例；组件客户端主要包括客户端实现、服务 Proxy 和客户端 API 等软件模块。共用功能组件模块的服务端和客户端均应采用 C/C++编程语言实现。

组件模型原理如图 3 所示。其中，BnFooService 为服务 Stub，BpFooService 为服务 Proxy。

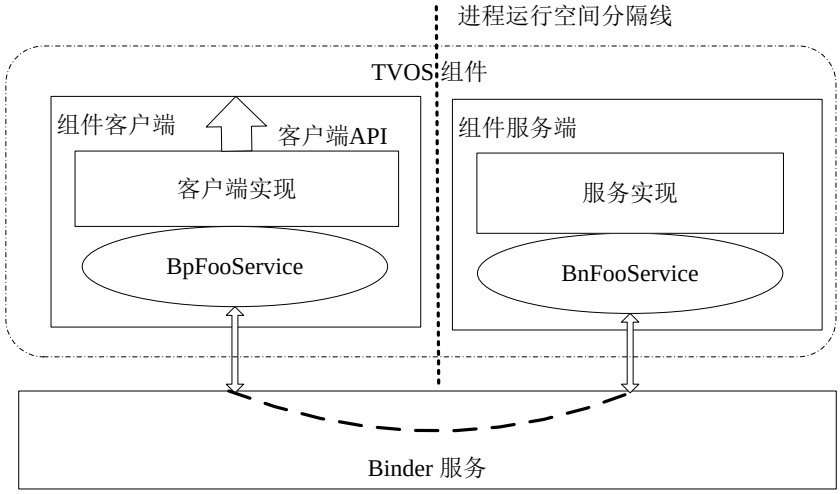


图 3 组件模型框图

组件服务端和客户端之间的协同工作需要组件服务管理器进行支持。

组件服务端应向组件服务管理器注册相应服务端信息，客户端应通过向组件服务管理器查询对应组件服务端的相关信息，实现对组件服务端的调用。

组件服务管理器应提供组件命名解析、维护组件名和组件实例对应关系、检查对组件服务端的访问权限、实现对组件服务端的访问权限控制等功能。

组件与组件服务管理器协同工作原理如图 4 所示。

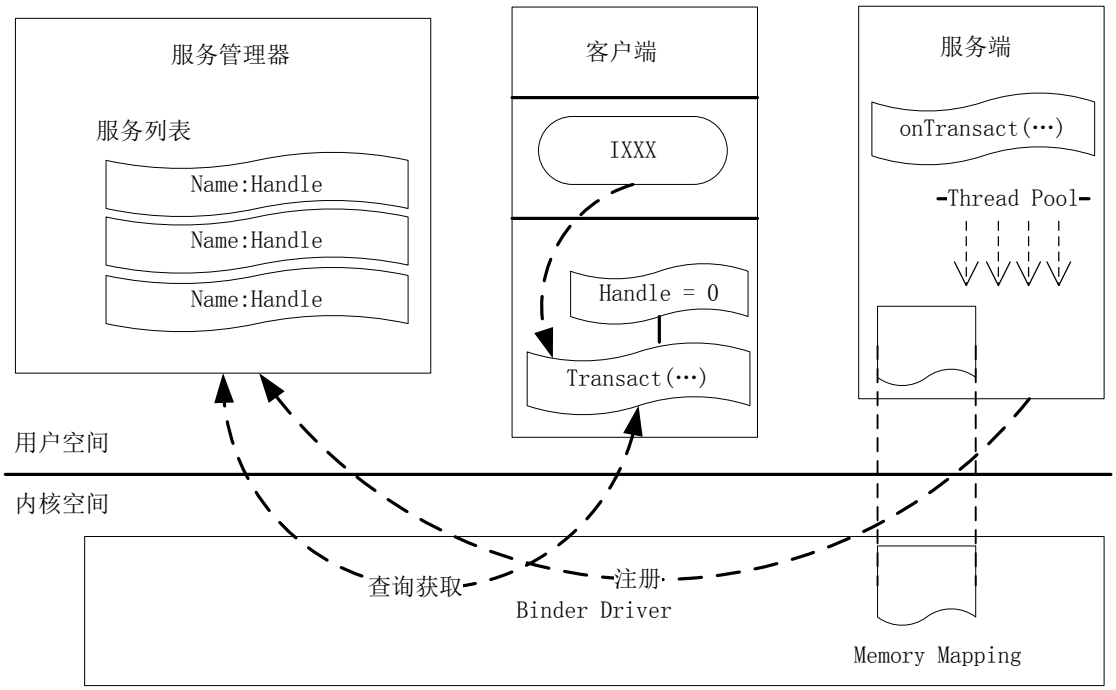


图 4　组件与组件服务管理器协同工作原理

8.2　组件服务管理器组件

8.2.1　功能

组件服务管理器组件应集中管理系统内所有组件，提供对 TVOS 组件注册的功能；应能检查组件是否为有效注册，包括检查组件是否合法、检查组件是否重复注册、检查组件是否可分配足够 Binder 内存资源；并为其他软件模块和应用程序提供查找组件并获取组件客户端的功能。组件服务管理器组件是提供组件管理功能的特殊组件，并遵循 TVOS 组件模型。

8.2.2　组件实现和调用方式

TVOS 组件服务管理器组件应按照组件模型实现，组件实现和调用方式如图 5 所示。

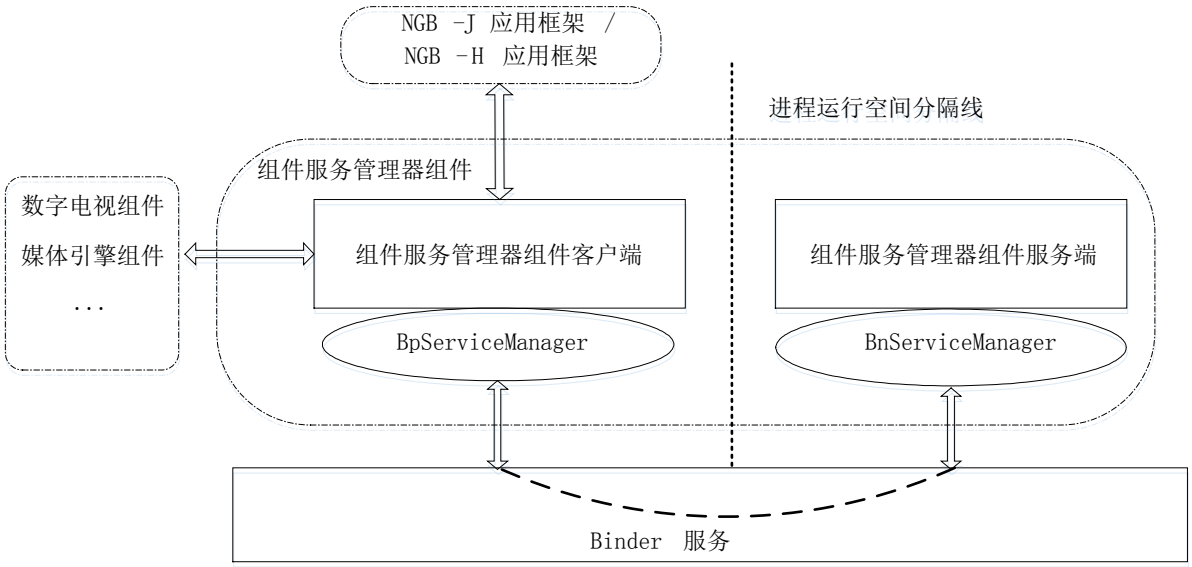


图 5 服务管理组件实现和调用方式

图 5 中，BnServiceManager 为服务 Stub，BpServiceManager 为服务 Proxy。

8.2.3 功能架构与模块

组件服务管理器组件由组件注册、组件有效性检查和组件查询模块组成，组件服务管理器组件架构如图 6 所示。

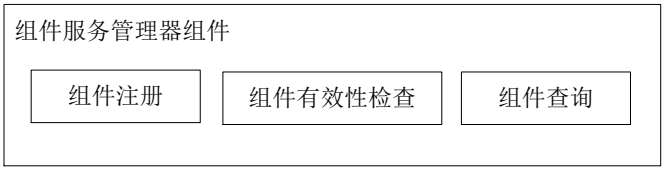


图 6 组件服务管理器组件功能架构与模块

组件注册模块负责响应其他组件注册到组件服务管理器组件的请求。

组件有效性检查模块负责检查组件的权限是否合法，组件是否重复注册以及是否有资源完成注册。

组件查询模块负责提供组件查询和组件客户端获取的功能。

8.2.4 接口

组件服务管理器组件应通过客户端向其他软件模块提供组件注册的接口，向其他软件模块和其他应用提供组件查询和组件客户端获取的接口。具体到接口，组件服务管理器组件提供了 add_service、check_service、get_service、list_service 四个调用接口。

8.2.5 与其他软件模块的协同

组件服务管理器组件被其他组件和其他应用依赖，为其他组件和应用提供组件注册和查询功能。组件服务管理器组件依赖系统底层 Binder 基础库和 Binder 驱动完成对其他组件的管理。

8.3 数字电视组件

8.3.1 功能

数字电视功能组件模块应实现用于各类数字电视广播协议 PSI/SI 数据和数据广播协议数据的搜索、过滤、获取、解析、存储和管理，实现对解调设备的调谐解调控制，为相关应用提供功能接口，为数字电视直播等相关应用提供接口和相应能力支撑；与媒体引擎组件、DCAS 组件和 DRM 组件等配合协同，支撑相关应用完成电视直播、节目导视、电视图文广告、视频点播、节目录制和时移、数据广播、频道预览等数字电视业务功能。

数字电视功能组件应支持对 GB/T 17975.1—2010 和 GB/T 28161—2011 标准所定义的协议和表格解析，应实现对有线、地面无线和卫星等多模数字电视终端的支持。

8.3.2 组件实现和调用方式

数字电视组件应按照组件模型实现，组件调用方式如图 7 所示。

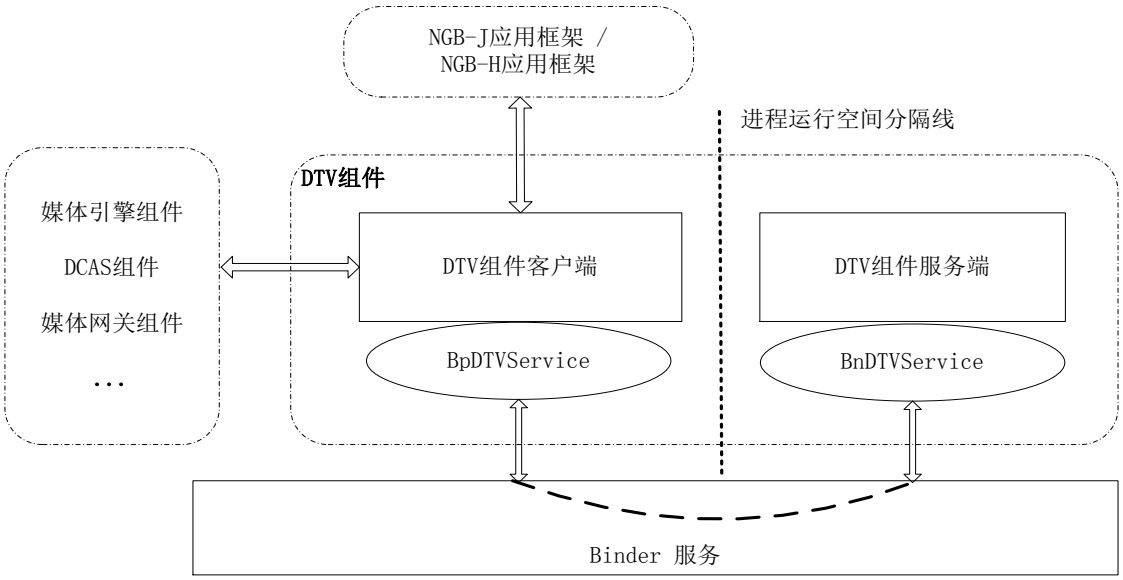


图 7 DTV 组件实现和调用方式

图 7 中，BnDTVService 为服务 Stub，BpDTVService 为服务 Proxy。

8.3.3 功能架构与模块

数字电视组件服务端由 TUNER 模块、DataEngine 模块、SCAN 模块、DB 模块、EPG 模块和 DT 模块组成，功能架构如图 8 所示。

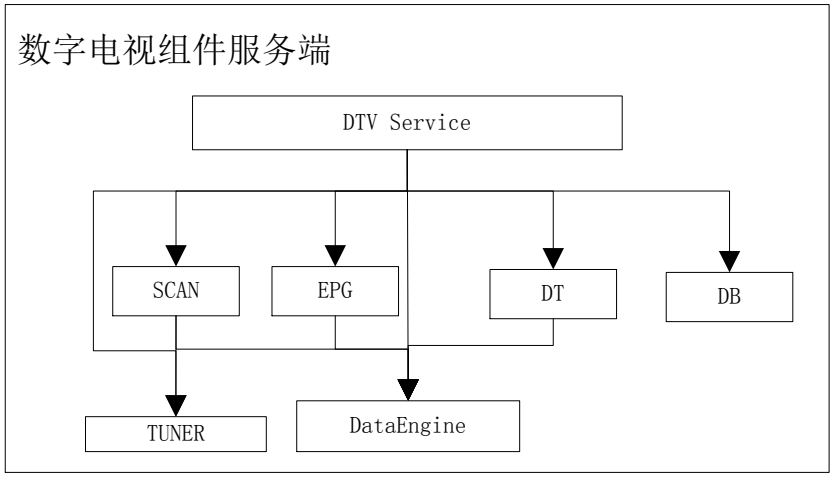


图 8　数字电视组件功能架构

TUNER 模块应实现对有线、无线、卫星等广播信道解调器的信道调谐和解调控制。

DataEngine 模块应实现音视频广播节目相关基础表格 section 数据的过滤和缓存，并实现对底层过滤器管理和传送流数据包 PID 的冲突管理。

SCAN 模块应实现对音视频广播节目相关基础表格 section 数据的解析，获取 NIT、PAT、PMT、SDT 等 PSI/SI 节目相关表格信息，支持自动搜索、单频点手动搜索和按频段手动搜索等多种搜索方式。

DB 模块应实现对 NIT、PAT、PMT、SDT 等 PSI/SI 节目相关表格信息的保存，为其他软件模块提供查询服务。

EPG 模块应实现对 EIT 节目事件信息表格 section 数据的解析，生成 EPG 信息数据，为其他软件模块提供查询服务。

DT 模块应实现对 TDT 和 TOT 时间相关表格 section 数据的解析，生成时间数据，为其他软件模块提供查询服务。

8.3.4　接口

数字电视组件应通过客户端向其他软件模块提供 TUNER 控制、节目搜索、SI/PSI 数据获取、EPG 数据获取、节目信息查询等调用接口。此部分接口属于功能组件接口。

8.3.5　与其他软件模块的协同

数字电视组件应与媒体处理组件、DCAS 组件以及 NGB-J 和 NGB-H 相关功能接口单元协同工作，协同关系如图 9 所示。

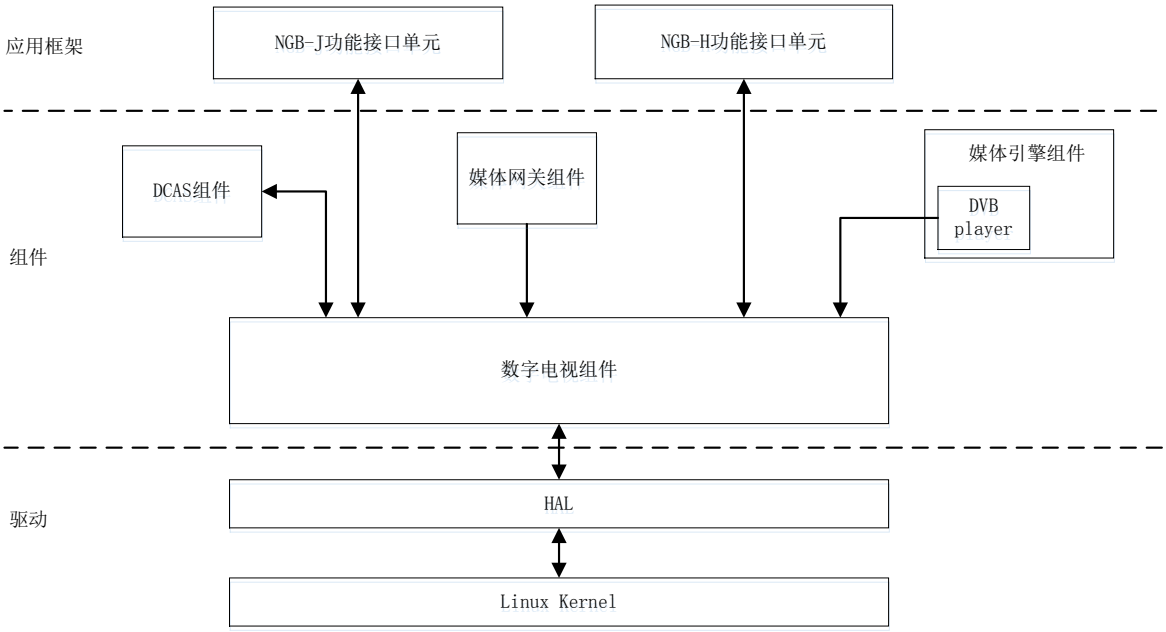


图 9　DTV 组件与其他软件模块的关系示意图

8.4　媒体引擎组件

8.4.1　功能

媒体引擎功能组件模块应实现对各类媒体音视频格式和协议的解析，与底层硬件协同实现各类媒体音视频播放、录制、转发，包括数字电视直播和点播、互联网电视、IPTV、游戏音视频和本地媒体文件等相关视音频格式的解码和音视频播放功能。

数字电视直播应支持非加扰和加扰的数字电视直播码流播放功能，支持切台、音量设置和窗口设置等播控功能。

数字电视点播应支持 VOD 点播码流播放功能，支持节目播放、暂停、恢复、停止、快进、快退、选时和音量设置等播控功能。

互联网电视应支持非加密和加密的 OTT 码流播放功能，支持节目播放、暂停、恢复、停止、快进、快退、选时和音量设置等播控功能。

本地媒体文件播放应支持本地媒体文件多音视频格式的音视频播放，支持节目播放、暂停、恢复、停止、快进、快退、选时和音量设置等播控功能。

8.4.2　组件实现和调用方式

媒体引擎组件应按照组件模型实现，组件调用方式如图 10 所示。

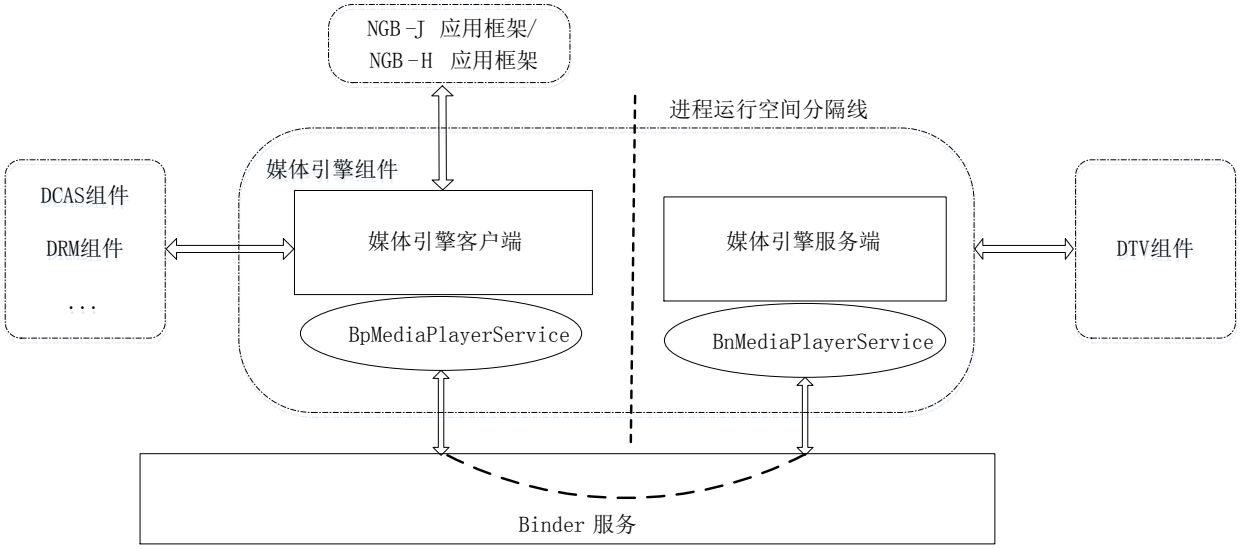


图 10　媒体引擎组件实现和调用方式

图 10 中，BnMediaPlayerService 为服务 Stub，BpMediaPlayerService 为服务 Proxy。

8.4.3　基础架构与机制

媒体引擎组件服务端应采用基于插件式的管道流水线全媒体处理架构实现，包括播放器管理器，DVBPlayer、VODPlayer、OTTPlayer 和 LocalPlayer 等媒体播放器，媒体播放管道管理器，DVBPlayer Pipeline、VODPlayer Pipeline、OTTPlayer Pipeline 和 LocalPlayer Pipeline 等播放器 Pipeline 模块，功能架构图 11 所示。

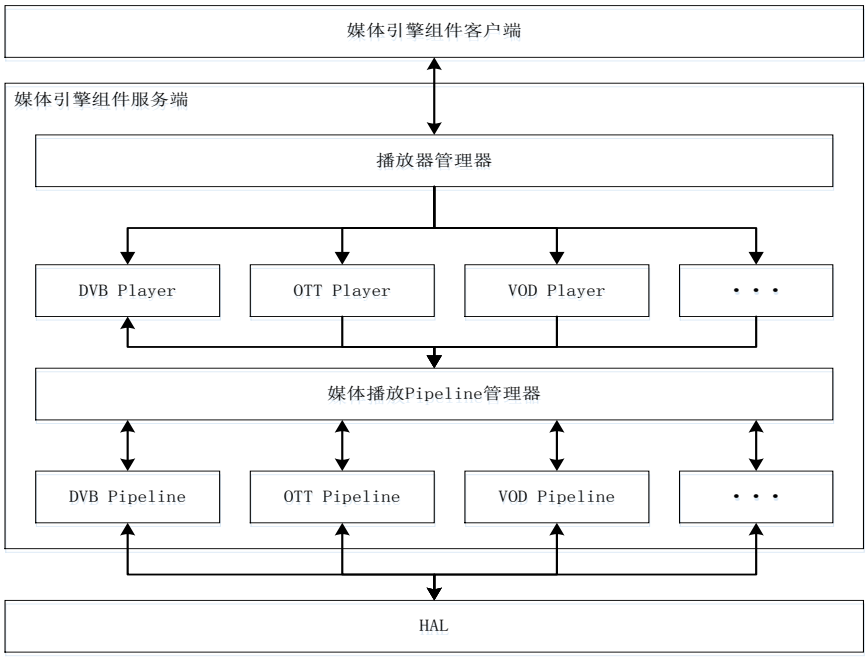


图 11　媒体引擎组件功能架构

播放器管理器应实现播放器的注册、种类选择、创建、销毁和状态等播放器管理功能，为其他软件模

块提供媒体播放服务。

DVBPlayer、VODPlayer、OTTPlayer 和 LocalPlayer 等各类播放器应按照相关的媒体播控协议实现相应媒体的播放控制逻辑。

媒体播放管道管理器应负责插件元件、Pipeline 搭建和 Pipeline 运行等管理功能。其中，插件元件是指媒体引擎组件中负责单一媒体处理功能的插件单元，包括 Source、Typefind、Demux、ProtocolParse、VideoDecoder、VideoSink、AudioDecoder 和 AudioSink 等不同类型的插件元件。

DVBPlayer Pipeline、VODPlayer Pipeline、OTTPlayer Pipeline 和 LocalPlayer Pipeline 等播放器 Pipeline 应按照媒体播放管道管理器的指令，通过对插件元件的组合实现相应媒体播放器的媒体播放功能。

播放器 Pipeline 应按照媒体播放管道管理器的要求，从 Source、Typefind、Demux、ProtocolParse、VideoDecoder、VideoSink、AudioDecoder 和 AudioSink 等不同类型元件中选择恰当的插件元件，放置于相应的 Pipeline 节点，播放器 Pipeline 工作机制如图 12 所示。

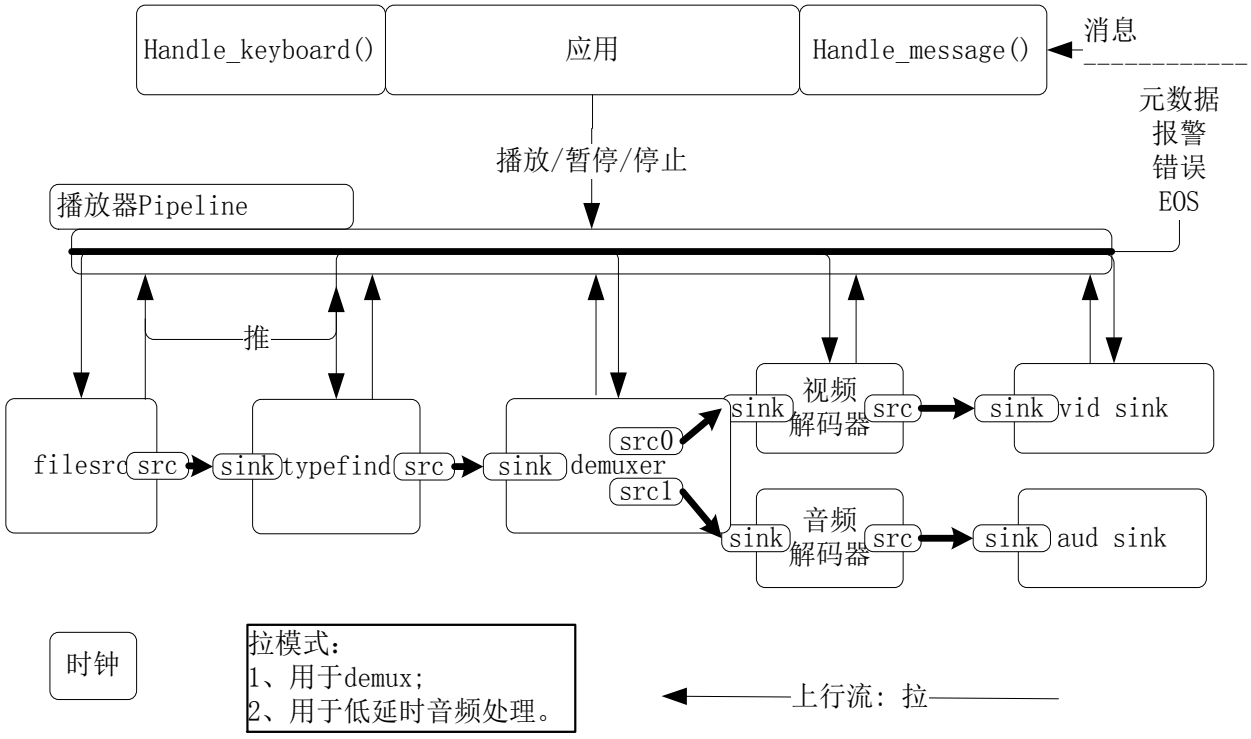


图 12　播放器 Pipeline 工作机制

媒体引擎组件应按照其他软件模块的媒体播放请求，由播放器管理器通过相应的媒体播放器，控制媒体播放管道管理器选择相关插件元件，搭建相应的媒体播放 Pipeline，实现为相应播放请求软件模块提供媒体播放服务；对应着每一种媒体播放请求，媒体引擎组件应有一个相应的媒体播放器相对应，相应媒体播放功能应通过相应媒体播放器 Pipeline 按照相应媒体播放器的播控逻辑实现；媒体播放器应能根据需要可扩展，构建媒体播放器 Pipeline 的插件元件也应能开放地扩展，从而实现对全媒体解析和播放的支撑。媒体引擎组件应能按照资源管理策略对播放器等资源进行调度和管理。

8.4.4　核心功能模块

8.4.4.1　基础插件元件

基础插件元件应包含 Source、Typefind、ProtocolParse、Demux、VideoDecoder、VideoSink、AudioDecoder 和 AudioSink 等类型：

a) Source 元件负责获取媒体流，包括本地播放元件 FileSource、句柄播放元件 FdSource、HTTP 协议解析元件 SoupHttpSource 等；
b) Typefind 元件负责对 source 插件传递的 MIME 类型，确定 demux 类型；
c) ProtocolParse 元件负责对各类媒体协议进行解析，包括 HLS Parse 和 DASH Parse 等元件；
d) Demux 元件负责对媒体数据进行解复用，分离出音频、视频、内嵌字幕 ES 流，并分别传递给解码插件元件解码；
e) Video Decoder 元件负责视频解码，可通过调用 HAL 接口，对视频 ES 流进行硬件解码；
f) Video Sink 元件负责视频输出，通过 vout 接口输出解码后的视频帧；
g) Audio Decoder 元件负责音频解码，可通过调用 HAL 接口，对音频 ES 流进行硬件解码；
h) Audio Sink 元件负责音频输出，通过 aout 接口输出解码后的 PCM 音频帧。

8.4.4.2 播放器管道管理器

媒体播放管道管理器应实现插件元件、Pipeline 搭建和 Pipeline 运行等管理功能。

媒体播放管道管理器应能接受不同插件元件的注册，具备相关插件元件标识能力，支持插件元件的自动查找和选择。

媒体播放管道管理器应支持自动渐进式的 Pipeline 搭建，即 Pipeline 的组建首先从创建 source 节点开始，由所创建的 source 节点结合播放器的相关需求选择恰当的插件元件作为紧随其后的节点元件，而下一级节点的元件选择由上一级节点元件结合播放器的需求确定，如此渐进式地构建整个播放器 Pipeline。

媒体播放管道管理器应通过对控制流、数据流和状态机的操作和控制以及时钟优先级的选择实现对播放器 Pipeline 的运行管理。

8.4.4.3 DVBPlayer 功能模块与 DVBPlayer Pipeline

DVBPlayer 是媒体引擎组件中实现数字电视直播的功能模块，应支持非加扰和加扰的直播码流播放功能，支持切台、音量设置和窗口设置等播控功能。

DVBPlayer 功能模块应包含频率调谐、节目信息获取、节目解扰和节目播放等子功能模块，功能模块框架及与其他模块的关系如图 13 所示。

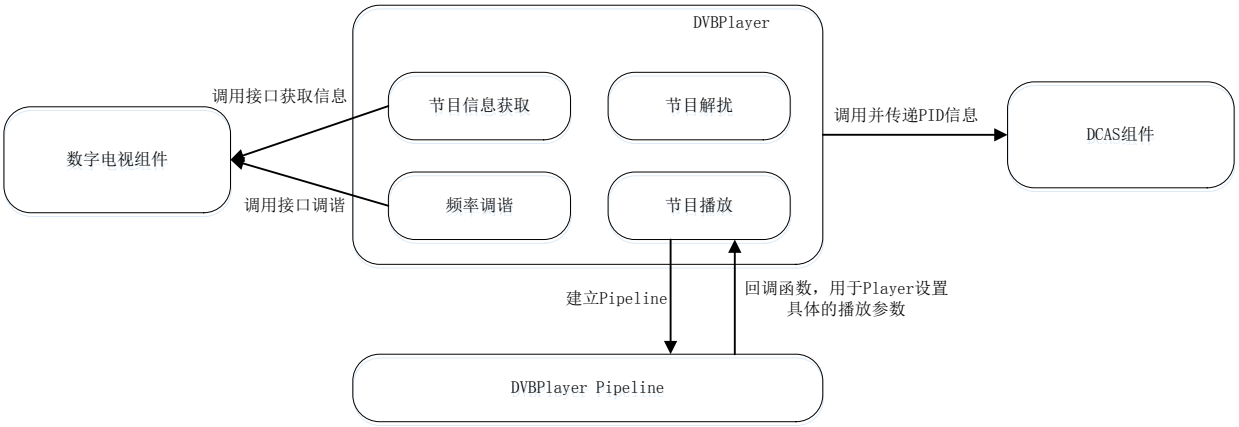


图 13 DVBPlayer 功能模块框架图

频率调谐子模块应实现对数字电视组件 TUNER 模块相关接口的调用，通过数字电视组件实现对相关直

播电视节目频道的频率调谐。

节目信息获取子模块应实现对数字电视组件 DB 模块相关接口的调用，通过该 DB 模块获取相关直播电视节目的相关节目信息。

节目解扰子模块应实现对 DCAS 组件相关接口的调用，将相关直播电视节目加扰参数信息传递给 DCAS 组件，以实现对相应直播电视节目的解扰。

节目播放子模块应与媒体播放管道管理器协同，通过媒体引擎相关插件元件的组合，构建 DVBPlayer Pipeline，以实现直播电视节目播放功能。

DVBPlayer Pipeline 应按照 DVBPlayer 的要求，在媒体播放管道管理器的控制下，实现相应媒体播放功能，DVBPlayer pipline 工作原理示意图如图 14 所示。

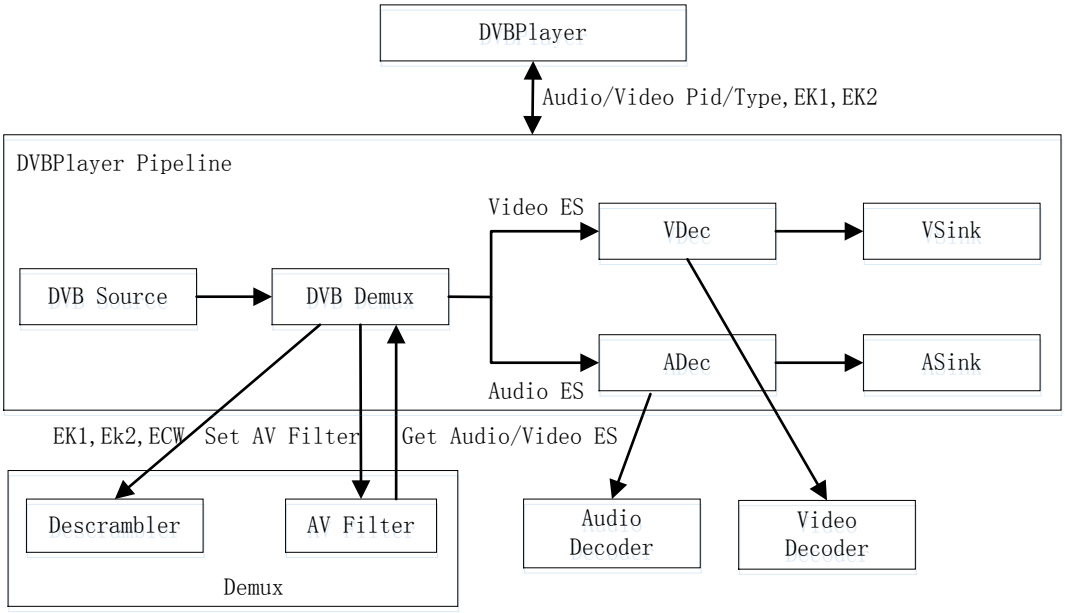


图 14　DVBPlayer Pipeline 工作原理示意图

8.4.4.4　VODPlayer 功能模块与 VODPlayer Pipeline

VODPlayer 是媒体引擎组件中实现数字电视点播播放的功能模块，应支持 VOD 点播码流播放功能，支持节目播放、暂停、恢复、停止、快进、快退、选时等播控功能；VODPlayer 应支持 GY/T 258—2012 标准相关协议的解析，可支持对 NGOD VOD 标准协议的解析。

VODPlayer 功能模块应包括频率调谐、节目信息获取、信令交互协议处理和节目播放等功能子模块，功能模块框架及与其他模块的关系如图 15 所示。

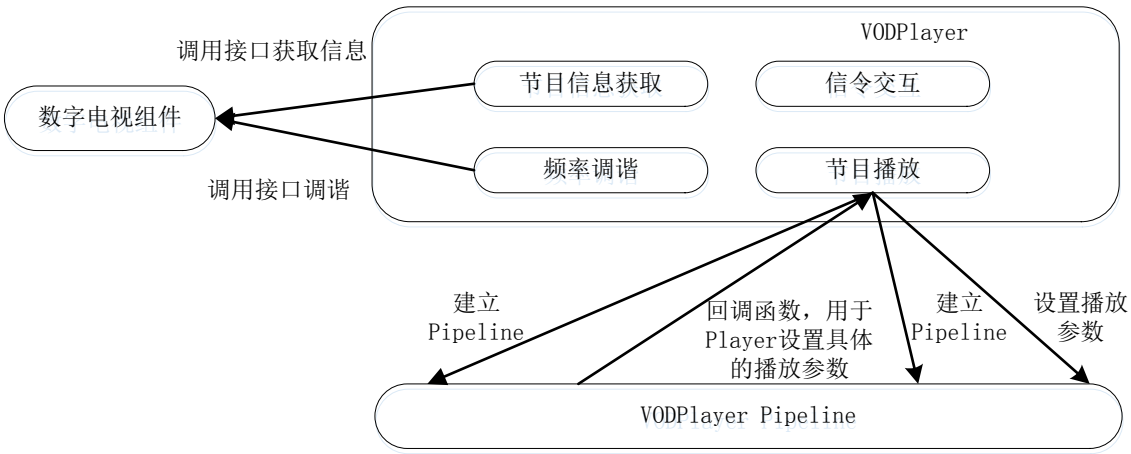


图 15　VODPlayer 功能模块框架及工作机制示意图

频率调谐子模块应实现对数字电视组件 TUNER 模块相关接口的调用，通过 TUNER 模块实现对相关点播节目频道的频率调谐。

节目信息获取子模块应实现对数字电视组件 DB 模块相关接口的调用，通过 DB 模块获取相关点播节目的节目信息。

信令交互协议处理子模块应实现与前端数字电视 VOD 系统的信令交互，控制节目播放子模块实现对基于 IPQAM 通道或 IP 宽带通道的 VOD 音视频点播流的播放。

节目播放子模块应与媒体播放管道管理器协同，通过媒体引擎相关插件元件的组合，构建 VODPlayer Pipeline，以实现点播电视节目的播放功能。

VODPlayer Pipeline 工作原理示意图如图 16 所示。

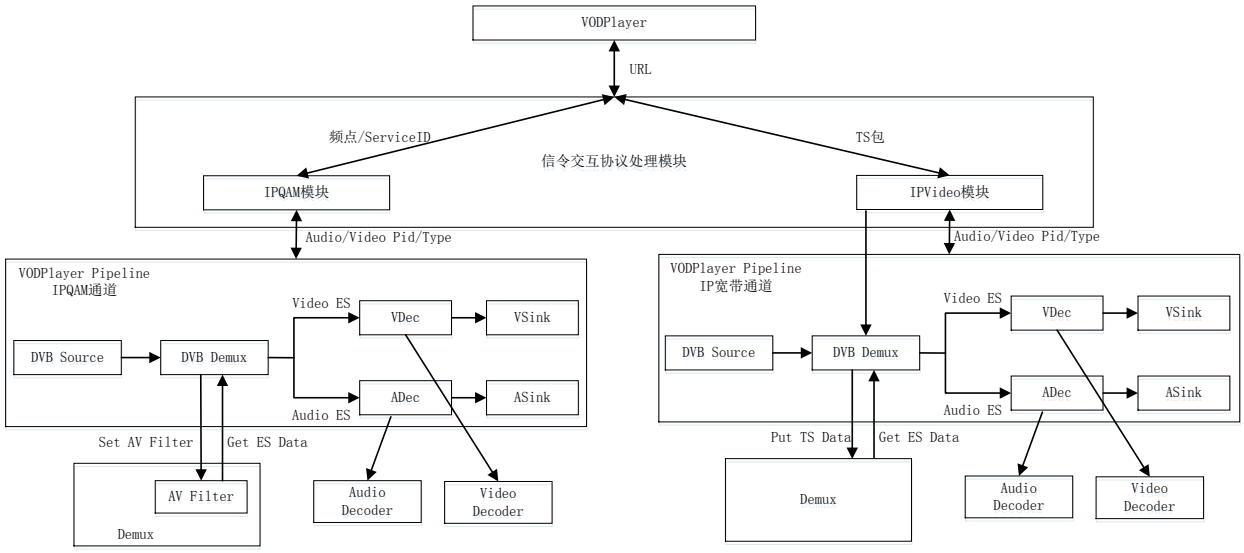


图 16　VODPlayer Pipeline 工作原理示意图

8.4.4.5　OTTPlayer 功能模块与 OTTPlayer Pipeline

OTTPlayer 是媒体引擎组件中实现互联网电视播放的功能模块，应支持基于 HttpProgressive、HLS 等流媒体协议的点播码流播放功能，支持节目播放、暂停、恢复、停止、快进、快退、选时等播控功能。

OTTPlayer pipeline 工作原理示意图如图 17 所示。

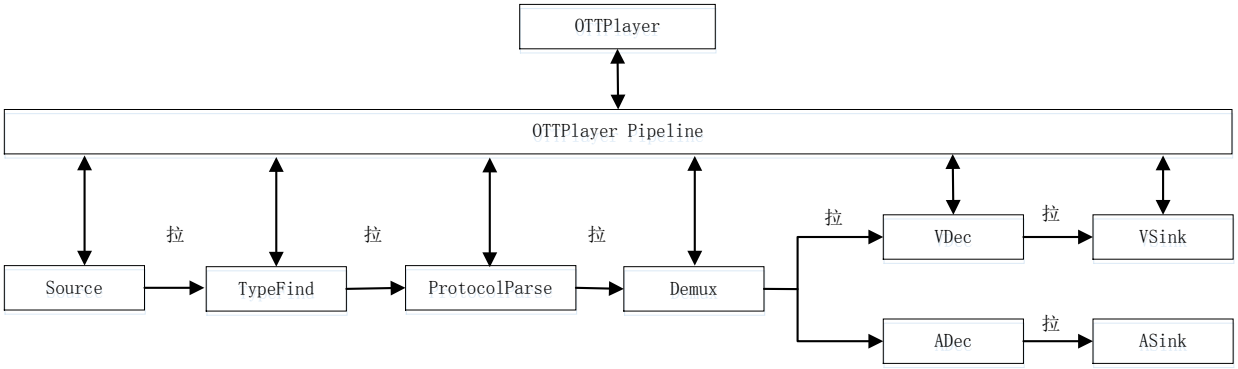


图 17　OTTPlayer Pipeline 工作原理示意图

在上述 OTTPlayer Pipeline 中，如果 Source 节点采用 HttpProgressive 插件元件，并且没有 ProtocolParse 元件节点，就形成了支持 HTTP Progressive 流媒体协议的 OTTPlayer Pipeline，如图 18 所示。

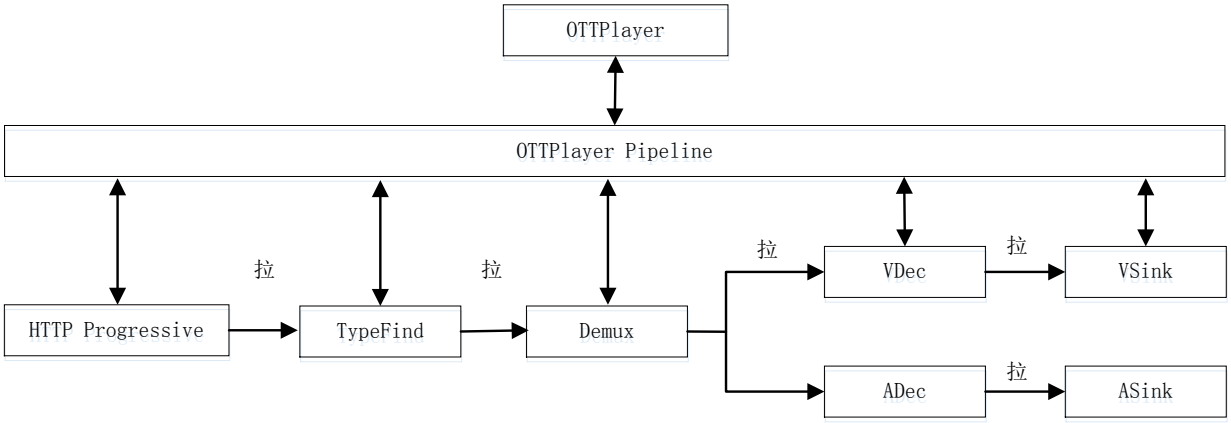


图 18　支持 HTTP Progressive 流媒体协议的 OTTPlayer pipeline

在图 17 所示的 OTTPlayer Pipeline 中，如果 Source 节点采用 HttpProgressive 插件元件，并且 ProtocolParse 元件节点采用 HLS Parse 插件元件，就形成了支持 HLS 流媒体协议的 OTTPlayer pipeline，如图 19 所示。

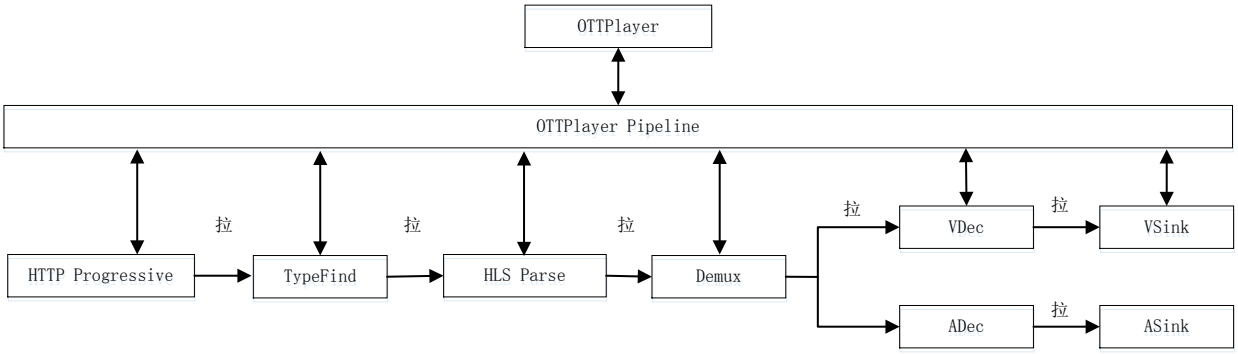


图 19　支持 HLS 流媒体协议的 OTTPlayer pipeline

在图 17 所示的 OTTPlayer Pipeline 中，如果 Source 节点采用 HttpProgressive 插件元件，并且 ProtocolParse 元件节点采用 DASH Parse 插件元件，就形成了支持 DASH 流媒体协议的 OTTPlayer Pipeline，如图 20 所示。

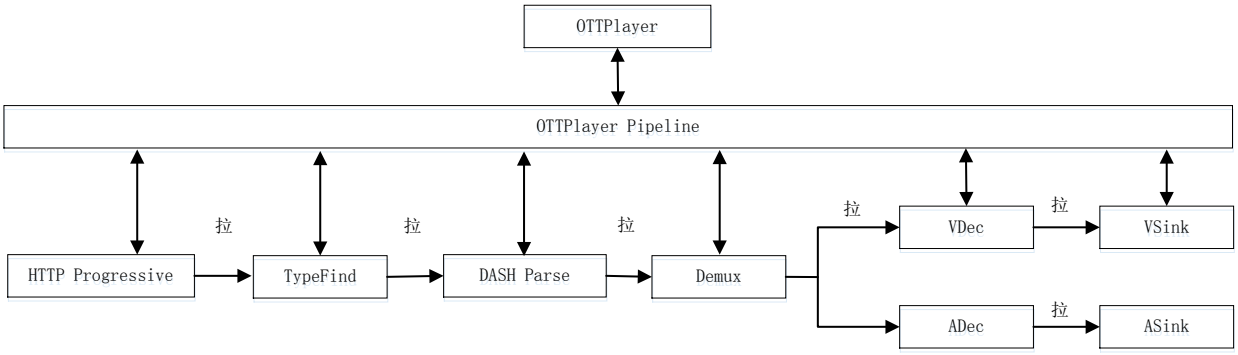


图 20　支持 DASH 流媒体协议的 OTTPlayer pipeline

在图 17 所示的 OTTPlayer Pipeline 中，如在 VDec 插件元件增加解密功能，就形成了支持加密互联网电视节目播放功能的 OTTPlayer Pipeline，如图 21 所示。

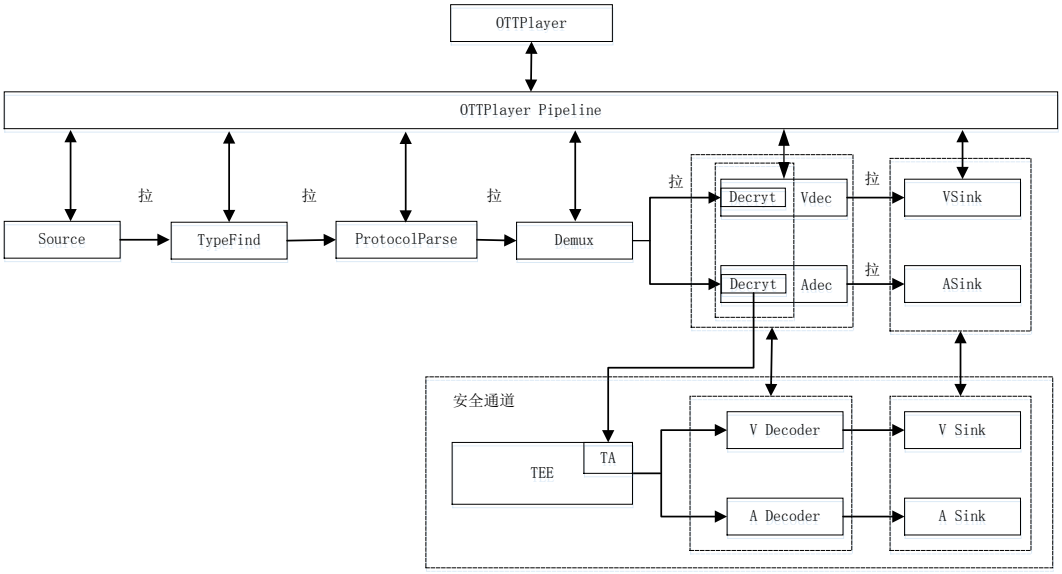


图 21　支持加密互联网电视节目播放功能的 OTTPlayer Pipeline

8.4.4.6　LocalPlayer 功能模块与 LocalPlayer Pipeline

本地播放器（LocalPlayer）是媒体引擎组件中实现本地媒体文件播放的功能模块，应支持对多种媒体格式文件的解析，支持节目播放、暂停、恢复、停止、快进、快退、选时等播控功能。

LocalPlayer pipeline 工作原理示意图如图 22 所示。

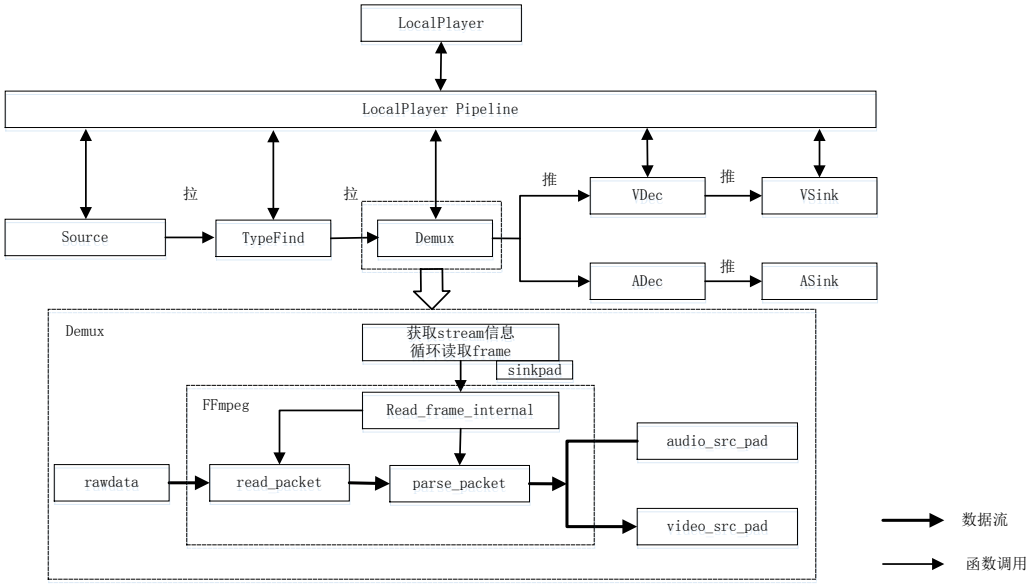


图 22　LocalPlayer Pipeline 工作原理示意图

8.4.5　接口

媒体引擎组件通过组件客户端统一为其他软件模块提供调用接口。组件客户端对外提供的调用接口应能根据媒体播放器功能的扩展需求而进行相应扩展。此部分接口属于功能组件接口。

8.4.6　与其他软件模块的协同

媒体引擎组件与其他软件模块的关系示意图如图 23 所示。

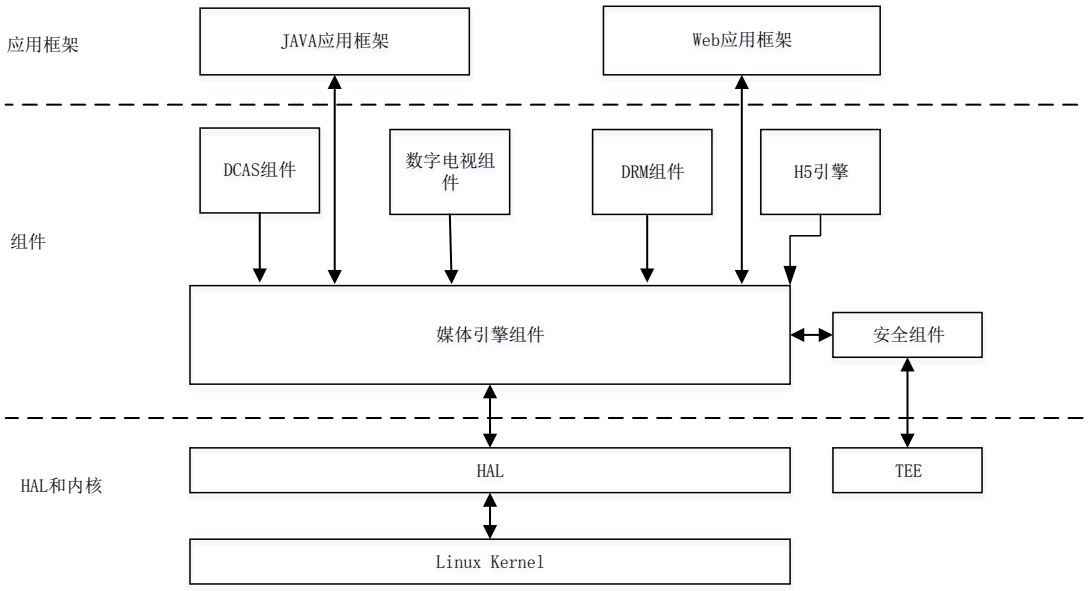


图 23　媒体引擎组件与其他软件模块的关系示意图

8.5　H5 引擎

8.5.1　功能

H5 引擎组件应实现 H5 网页的下载、解析、渲染、排版、呈现和脚本执行等功能；应与应用管理组件协同实现 WEB 应用的挂起、中止和销毁等功能；应与窗口管理组件协同实现输入等事件转发功能；应实现与DTV组件、DCAS组件、DRM组件、媒体引擎组件和人机交互等其他组件的适配和调用；对上应为Web Runtime提供统一的 H5 引擎接口，对下应具备调用 HAL 硬件抽象接口的能力。

H5 引擎组件应支持 HTML5、JavaScript 和 CSS 等 W3C 相关标准，并实现对 WEB 应用框架层相关 JS API 接口的适配，包括 NGB-H、DCAS API 和广播信息服务 API 等的适配。

8.5.2 组件实现和调用方式

H5引擎组件采用服务端和客户端模型，组件服务端负责实现页面的下载、解析、渲染、排版、呈现和脚本执行，组件客户端负责与Web Runtime、应用管理组件协同实现应用的挂起、中止和销毁等组件服务端运行实例的管理，与窗口管理组件协同实现输入等事件转发功能。服务端和客户端都可以运行多个实例，每个客户端和服务端运行实例分别运行在不同的进程空间。Web Runtime和应用管理组件等其他软件模块可以创建不同的客户端运行实例，Web Runtime可以创建多个不同的组件服务端运行实例，一般而言，一个WEB应用对应一个相应的组件服务端运行实例。H5引擎组件的实现和调用方式如图24所示。

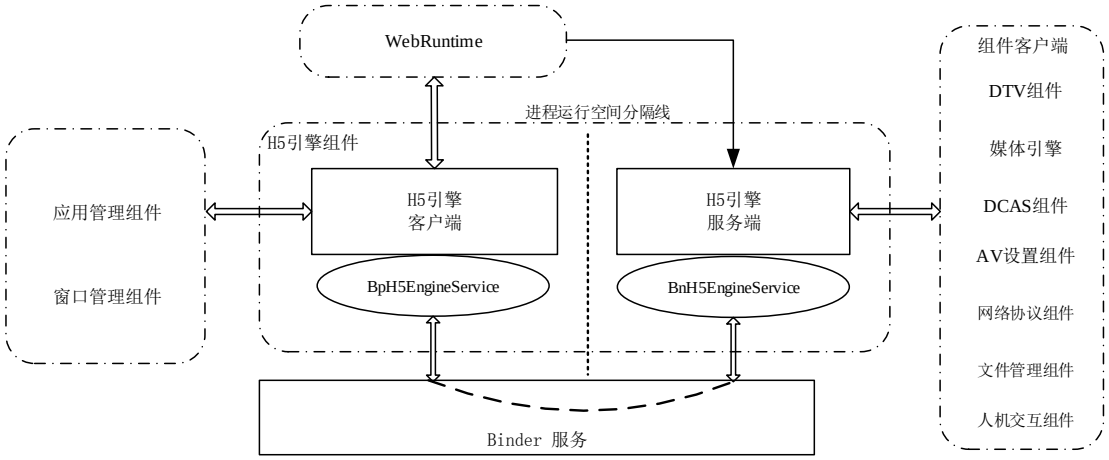


图 24 H5 引擎组件的实现和调用方式

图 24 中，BnH5EngineService 为服务 Stub，BpH5EngineService 为服务 Proxy。

8.5.3 功能架构与模块

H5 引擎组件服务端包含 ContentShell、ContentBrowser、Graphics、Network、window、ContentRender、HTML 渲染、JS 脚本执行、plugin、Media、Font、Diagnosis 和 Database 等功能模块。组件服务端的功能架构如图 25 所示。

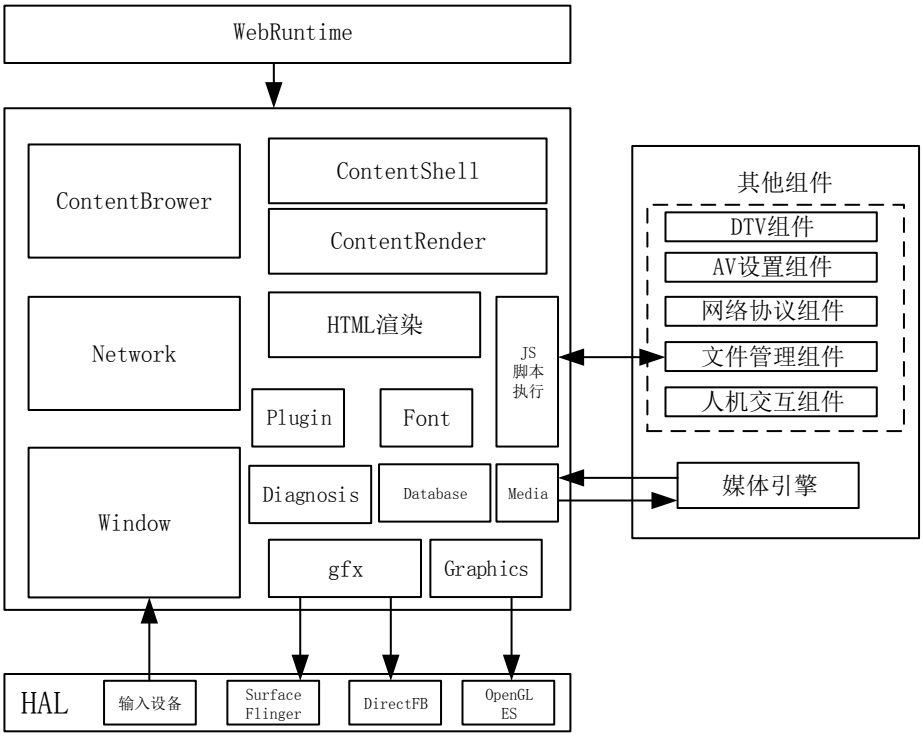


图 25　组件服务端的功能架构

ContentShell 模块负责提供 H5 引擎接口，供 WebRuntime 调用。

ContentBrowser 模块负责接收 ContentShell 命令，管理 Graphics、Network、window 消息调度，为 ContentRender 模块中转消息。

ContentRender 模块负责管理 HTML 渲染、JS 脚本执行、plugin、Media、Font、Diagnosis 和 Database 消息调度。

HTML 渲染模块负责页面资源加载、内容和格式解析、布局计算、渲染显示等功能。

JS 脚本执行模块负责 JavaScript 脚本的解析和运行，通过脚本控制和访问 DOM 树，实现动态交互。

Plugin 模块负责 H5 引擎的插件管理，使得浏览器能够以插件的方式执行外部程序。

Graphics 模块负责绘图和图像合成相关功能，支持 GPU 硬件加速。

Network 模块负责从网络下载各种类型的资源。

Media 模块负责与媒体引擎组件的对接，实现网页媒体播放功能。

Diagnosis 模块负责实现程序崩溃捕获机制，支持跨平台的崩溃转储，并且能够进行异常统一分析。

Window 模块负责窗口管理、UI 事件交互机制和显示元素的实现。

Database 模块负责数据库管理，支持 SQLite 和 leveldatabase。

8.5.4　接口

H5 引擎组件服务端和客户端都对外提供接口。组件服务端提供的接口包括 fork、contentshell 和 contentbrowser 接口，组件客户端提供的接口包括输入事件接口、窗口绘制事件接口、应用管理接口。

Fork 接口支撑组件服务端以参数方式创建运行实例。

Contentshell 接口支撑组件服务器端以定制或配置方式启动自运行服务。

ContentBrowser 接口提供应用入口加载，历史记录前进与后退等管理服务，用户代理设置服务。

输入事件接口提供遥控器等输入设备的消息接收。

窗口绘制事件接口提供绘制、重绘、无效区域事件的分发。

应用管理接口提供应用的运行、暂停、中断、销毁命令。

此部分接口属于功能组件接口。

8.5.5 与其他软件模块的协同

H5 引擎在 TVOS 中的位置及周边关联组件如图 26 所示。

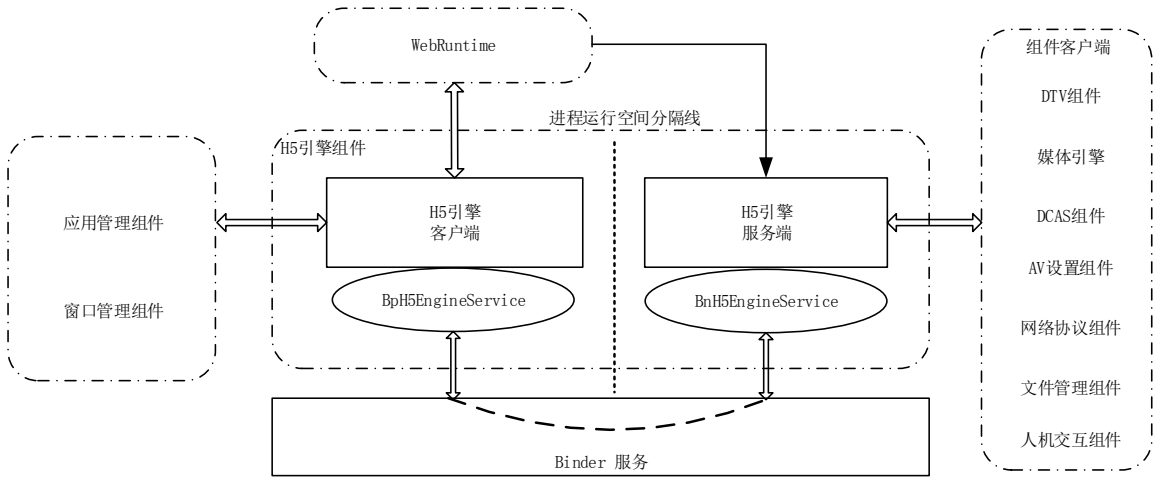


图 26 H5 引擎与其他软件模块的协同

8.6 DRM 组件

8.6.1 功能

DRM组件应实现对DRM App注册、注销和运行状态的管理，实现DRM App与DRM TApp之间的消息传递，实现媒体引擎组件与DRM App和DRM TApp之间的消息传递，支持DRM App和DRM TApp与媒体组件协同实现对加密媒体流的解密，为应用框架层功能接口单元和功能组件层的其他组件提供DRM调用接口。

8.6.2 组件实现和调用方式

DRM 组件应按照组件模型实现，组件调用方式如图 27 所示。

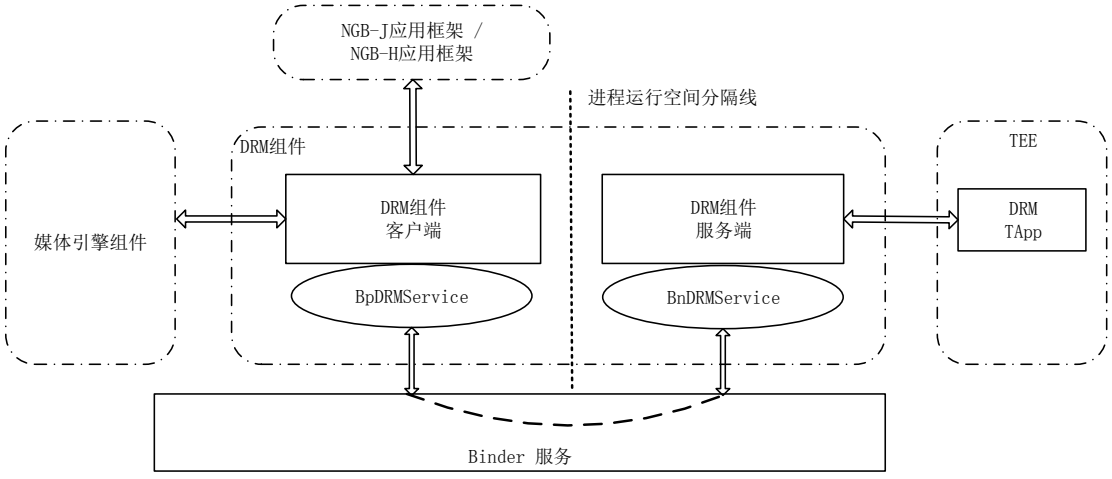


图 27 DRM 组件实现和调用方式

图 27 中，BnDRMService 为服务 Stub，BpDRMService 为服务 Proxy。

8.6.3 功能模块与架构

DRM 组件服务端由 DRM 应用管理器和消息管理器组成，功能架构如图 28 所示。

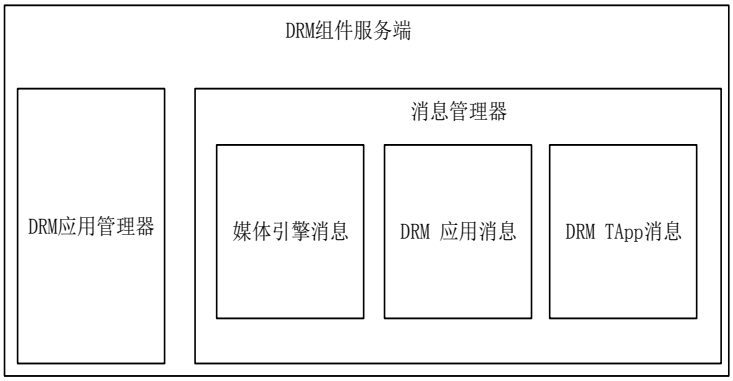


图 28 DRM 组件功能架构图

DRM 应用管理器应实现管理 DRM 应用，维护 DRM 应用标识，维护 DRM 应用与 DRM TApp 的对应关系，接收来自 DRM Server 中其他服务模块的请求，执行对 DRM 应用的签名验证、加载、调用及卸载。

消息管理器应实现媒体引擎组件和 DRM 应用、DRM 应用和 DRM TApp 之间的消息传递功能。消息传递的类型包括接收媒体引擎组件发送的加密内容共享内存地址，转发给 DRM 应用；接收 DRM 应用层发送的加密内容共享内存地址、加密的内容加密密钥信息，转发给 DRM TApp；接收 DRM TApp 的内容解密结果和解密后数据的安全缓冲区地址，转发给 DRM 应用。

8.6.4 接口

DRM 组件应通过客户端向其他软件模块提供权限获取及判断、内容解密等调用接口。此部分接口属于功能组件接口。

8.6.5 与其他软件模块的协同

DRM 组件同应用框架层、组件层、DRM TApp、媒体引擎组件相关功能接口单元协同工作，协同关系示意图如图 29 所示。

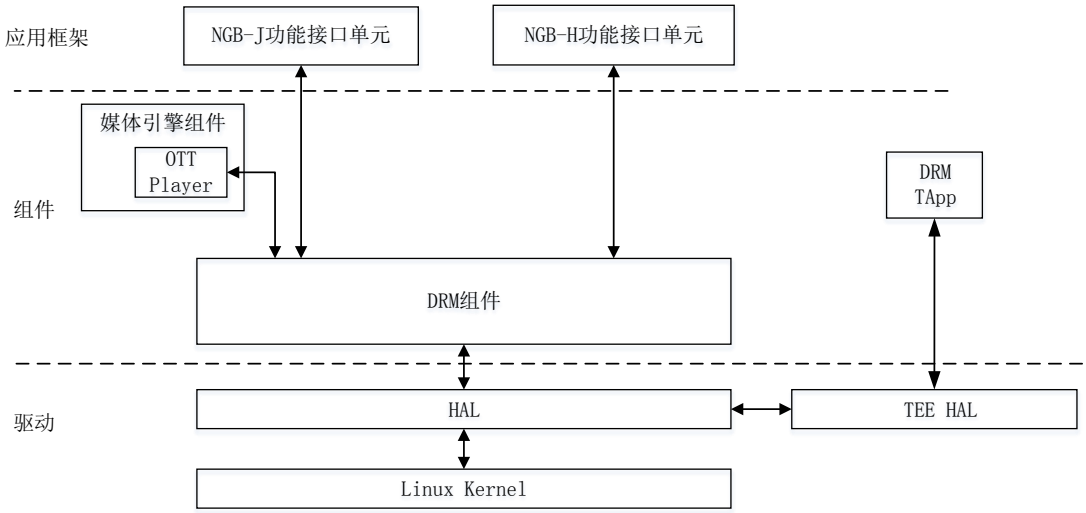


图 29　DRM 组件与其他软件模块的关系示意图

8.7　DCAS 组件

8.7.1　功能

DCAS组件应与DTV组件协同实现带内传输条件接收授权控制信息和授权管理信息的接收和转发，应与相关网络协议栈模块协同实现带外传输条件接收授权管理信息的接收和转发，应为DCAS App与DCAS TApp提供信息交换通道，应支撑媒体引擎组件与DCAS App和DCAS TApp实现信息交互，实现对DCAS App的注册和管理，应支持CA版本、Chip ID和授权状态等CA相关信息的查询，以及OSD显示和指纹显示等信息的接收和转发。

8.7.2　组件实现和调用方式

DCAS 组件按照组件应按照组件模型实现，组件调用方式如图 30 所示。

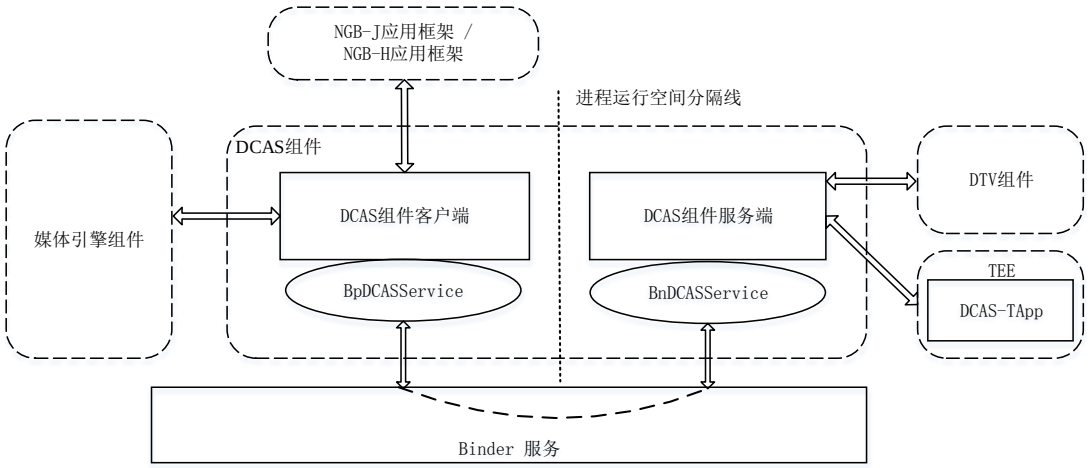


图 30　DCAS 组件实现和调用方式

图 30 中，BnDCASService 为服务 Stub，BpDCASService 为服务 Proxy。

8.7.3　功能模块与架构

DCAS 组件应包括 CA 数据接收转发、CA 应用管理和可信应用接口三个功能模块，组件架构如图 31 所示。

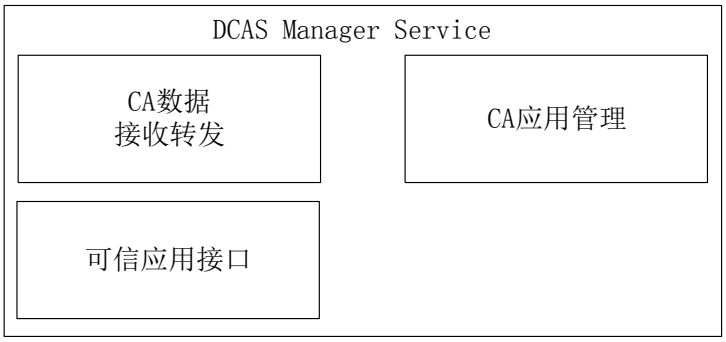


图 31　DCAS 组件逻辑架构

CA 数据接收转发模块负责按照接收到的数字电视解扰参数信息，从 DTV 组件或网络协议栈模块获取 ECM 和 EMM 数据，并向 DCAS App 转发。

可信应用接口模块负责与 DCAS TApp 接口，并为 DCAS App 与 DCAS TApp 提供信息交换通道。

CA 应用管理模块负责 DCAS 应用的注册和注销管理，负责按照条件接收系统标识实现媒体加密流与 DCAS 应用的匹配，负责从媒体引擎接收数字电视节目视频流标识 videoPid、音频流标识 audioPid、条件接收应用标识 CASystemId、授权控制信息标识 ecmPid、授权管理信息标识 emmPid 以及码流路径 StreamPath 等数字电视解扰相关参数并转发给相匹配的 DCAS 应用，负责支持对 CA 版本、Chip ID 和授权状态等 CA 相关信息的查询，以及 OSD 显示和指纹显示等信息的接收和转发。

8.7.4　接口

DCAS 组件应提供以下两类接口：

a)　解扰操作通知接口

用于接收 DCAS 解扰通知信息，包括启动解扰和停止解扰通知信息等，供媒体引擎组件调用。

b)　CA 应用接口

用于 DCAS-App 与 DCAS 组件之间的数据交互。

此部分接口属于功能组件接口。

8.7.5　与其他软件模块的协同

DCAS 组件同其他组件的关系如图 32 所示。

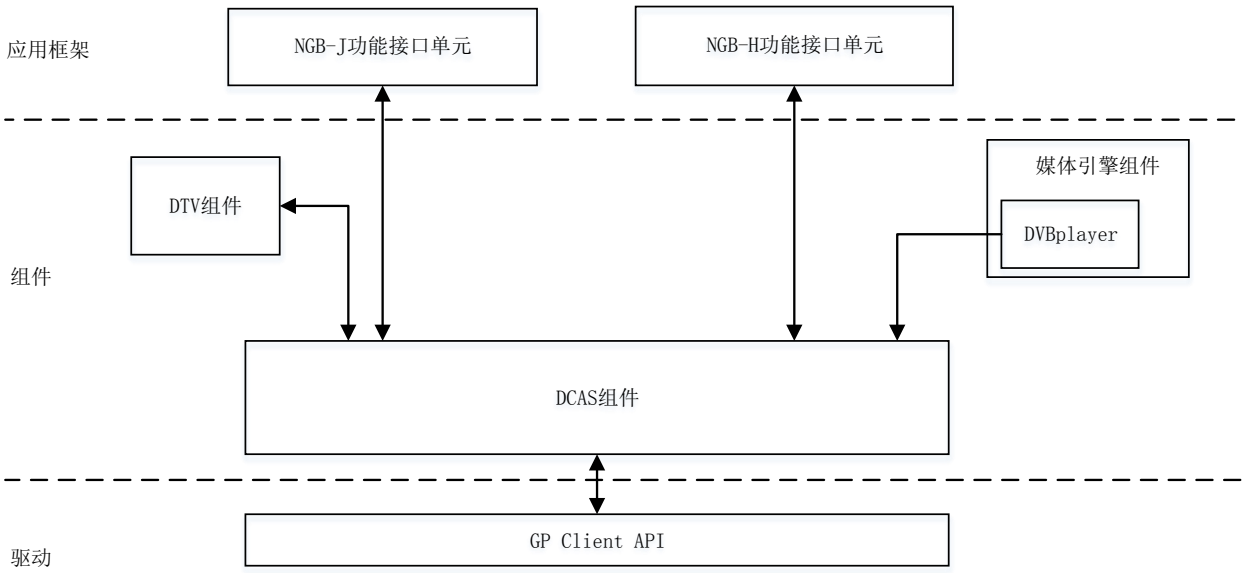


图 32　DCAS 组件同其他组件的关系

媒体引擎组件需要调用 DCAS 组件的解扰操作通知接口。

DCAS App 需要通过 NGB-J 或 NGB-H 相关功能接口单元调用 DCAS 组件 CA 应用接口。

DCAS 组件需要依赖 DTV 组件数据获取相关接口。

DCAS 组件需要依赖 TEE Client HAL 模块接口。

8.8　安全支付组件

8.8.1　功能

安全支付组件模块应实现订单签名、用户账号密码安全输入、支付校验码安全输入和验证等安全支付相关功能，为安全支付 App 和安全支付 TApp 提供信息和数据交换通道，为 JAVA 和 WEB 应用框架提供调用接口。安全支付组件应支持基于不同支付系统的支付应用 App 和支付应用 TApp。

8.8.2　组件实现和调用方式

安全支付组件应按照组件模型实现，组件调用方式如图 33 所示。

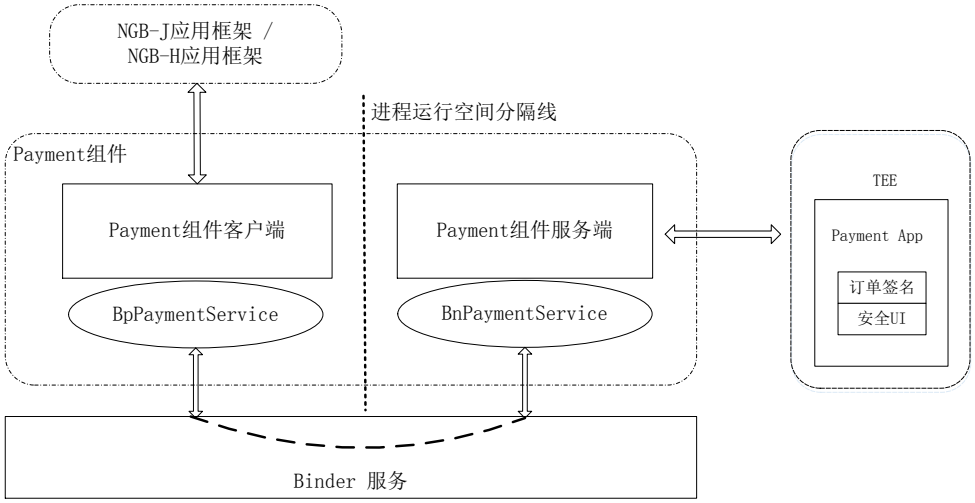


图 33　安全支付组件模型

图 33 中，BnPaymentService 为服务 Stub，BpPaymentService 为服务 Proxy。

8.8.3　功能模块与架构

安全支付组件服务端由 Message Manager 模块、Payer Manager 模块、Payer 模块和 TApp Service 模块组成，功能架构如图 34 所示。

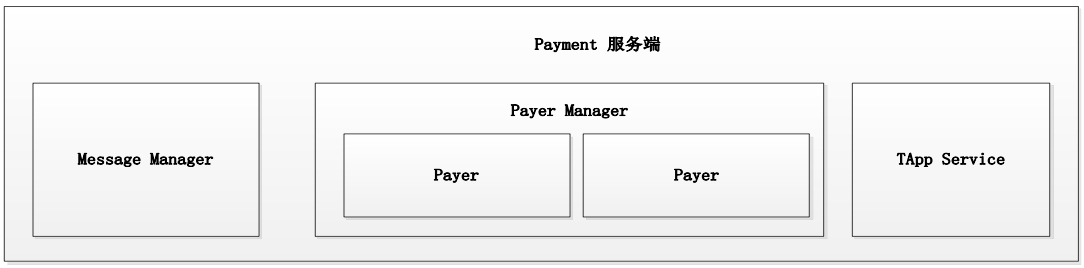


图 34　Payment Server 功能模块

Message Manager 应为安全支付 App 和安全支付 TApp 之间的支付相关消息和数据传递提供通道。

Payer Manager 应支撑安全支付 TApp 向安全支付 App 提供安全支付相关功能，如订单签名、加密的用户账号密码等。

TApp Service 应通过 TEE Client API 实现对 Payment TApp 的调用，以获取 Payment TApp 提供的安全支付服务，包括基于二维码扫描的支付支撑服务和基于用户账号密码的支付支撑服务。

8.8.4　接口

安全支付组件应通过客户端向其他软件模块提供的接口具体如下：

a)　安全支付组件 Payment Client 接口；

b)　安全支付组件的 Payment TApp 接口。

此部分接口属于功能组件接口。

8.9　智能家居组件

8.9.1 功能

智能家居组件应实现对智能家居设备发现、连接建立和操控的管理；提供对第三方智能家居互联协议的转换和适配接口，支持对第三方智能家居互联协议的扩展，实现对相应智能家居设备的管理；实现对智能家居设备 WiFi 网络参数的配置。

应支持 AllJoyn 等第三方智能家居互联协议，支持 AirKiss 等网络参数配置协议，支持标准蓝牙协议。

8.9.2 组件实现和调用方式

智能家居组件应按照组件模型实现，组件调用方式如图 35 所示。

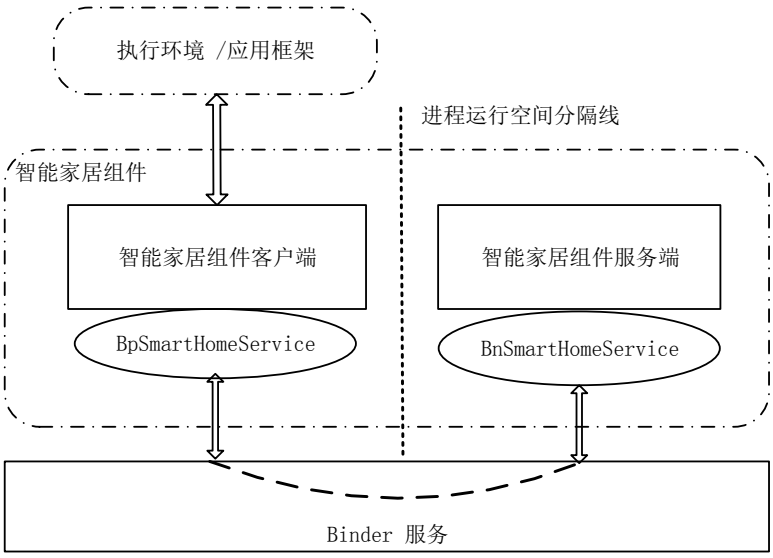


图 35 智能家居组件实现和调用方式

图 35 中，BnSmartHomeService 为服务 Stub，BpSmartHomeService 为服务 Proxy。

8.9.3 功能模块与架构

智能家居组件由设备配置模块、智能家居管理模块、协议转换适配模块组成，其结构如图 36 所示。

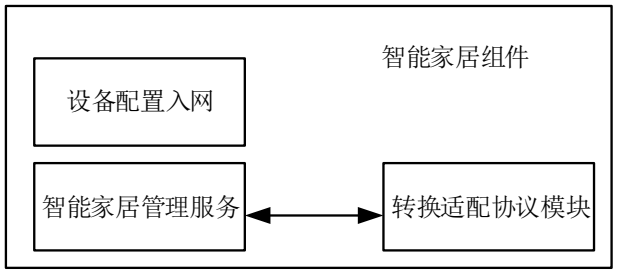


图 36 智能家居组件架构

设备配置模块应实现对基于 AirKiss 等网络参数配置协议的智能家居设备进行入网参数配置。

智能家居管理模块应实现对智能家居设备的发现、连接和操控等管理，支持 ALLJoyn 等第三方智能家居互联协议，支持相关安全互联功能，为其他软件模块提供操控智能家居设备的调用接口。

协议转换适配模块应提供对第三方智能家居互联协议的转换和适配接口，支持对第三方智能家居互联协议的扩展。

8.9.4 接口

智能家居组件主要为其他软件模块提供三类接口：

a) 智能家居组件设备联网接口：负责实现对 WiFi 或蓝牙智能家居设备进行网络参数配置的接口；
b) 智能家居组件设备管理接口：负责实现管理智能家居设备的接口；
c) 智能家居组件协议转换适配接口：负责为其他软件模块提供第三方智能家居互联协议的转换和适配接口。

此部分接口属于功能组件接口。

8.10 人机交互

8.10.1 功能

人机交互组件应实现对遥控器、键盘、鼠标、游戏手柄和移动终端等输入设备的输入信息处理；对语音操控输入和传感器输入的信息处理；对基于游戏的触屏输入与游戏手柄输入的适配；对上层软件模块和其他组件提供调用接口。

8.10.2 组件实现和调用方式

人机交互组件应按照组件模型实现，组件调用方式如图 37 所示。

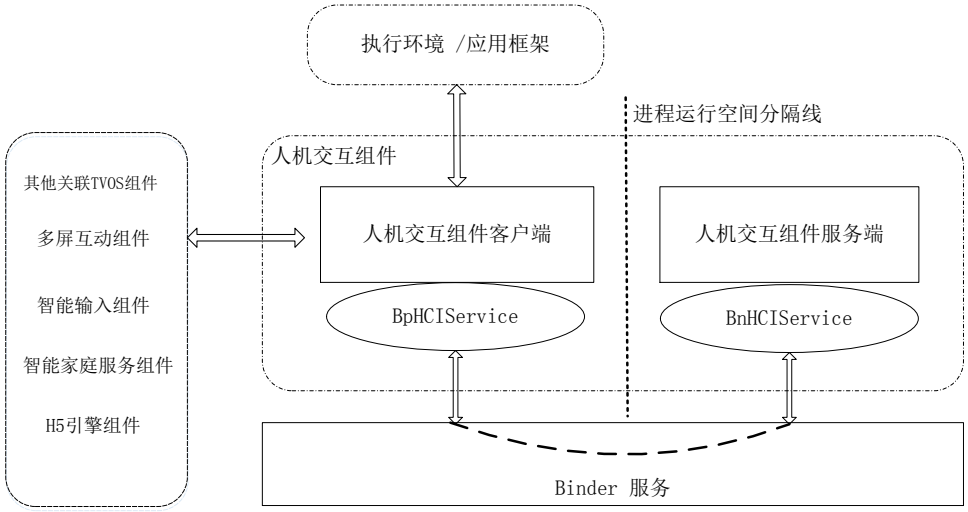


图 37 人机交互组件实现和调用方式

图 37 中，BnHCIService 为服务 Stub，BpHCIService 为服务 Proxy。

8.10.3 功能架构与模块

人机交互组件由键盘与鼠标消息处理、语音消息处理、Sensor 消息处理和虚拟控制消息处理等子功能模块组成，其结构如图 38 所示。

键盘与鼠标消息处理模块负责接收插入 TVOS 系统的键盘和鼠标数据以及经过虚拟控制接口转换成的虚拟键盘鼠标数据，提供按键、鼠标事件接口，并对键盘和鼠标输入的消息进行处理。

Sensor 消息处理模块负责接收插入 TVOS 系统的 Sensor 数据和通过虚拟控制接口转换成的虚拟 Sensor 数据，提供 Sensor 数据接口，实现对 Sensor 裸数据算法处理，读取 Sensor 数据或针对收到的 Sensor 数据实现体感动作的模拟。

语音消息处理模块负责接收通过 USB 麦克输入的数据，提供语音消息识别的处理框架和接口，支持云端语音消息识别引擎协同完成对语音消息的识别和处理，并将语音信息识别数据提供给上层语音控制应用，以实现语音控制功能。

虚拟控制消息处理模块负责对智能移动终端消息和游戏手柄消息进行适配处理。智能移动终端消息处理支持从智能移动终端获取数据，通过相应虚拟 Manager 接口，将所获取数据转化为模拟键盘、鼠标和 Sensor 的输入事件，并根据相应的事件进行操控；游戏手柄适配处理实现主流游戏手柄与非游戏手柄的输入适配，实现对触屏游戏的输入事件转换，实现按键与触摸、Sensor 的动作映射。

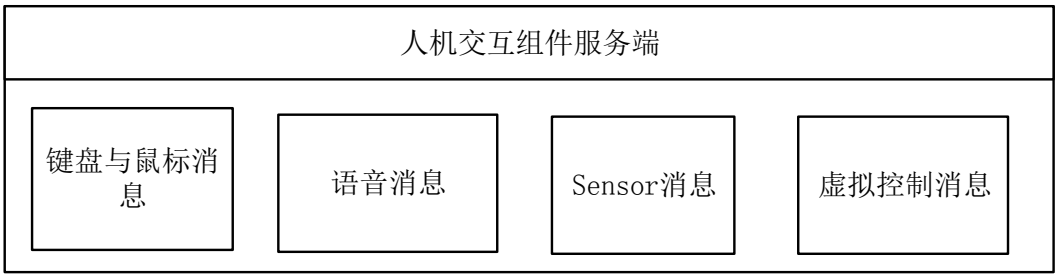


图 38　HCI Server 模块

8.10.4　接口

人机交互组件为其他软件模块提供按键事件接口、触屏/鼠标事件接口、事件注入接口、体感控制接口和语音消息接口。此部分接口属于功能组件接口。

8.10.5　与其他软件模块的协同

人机交互组件支持与多屏互动组件、智能家居组件和 H5 引擎组件等组件的协同，并通过应用框架层功能单元软件模块向不同应用提供调用接口。人机交互组件与其他软件模块的协同如图 39 所示。

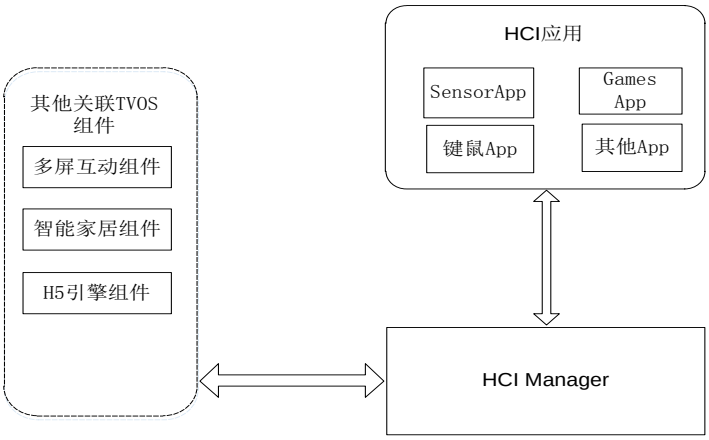


图 39　人机交互组件与其他软件模块的协同

8.11 多屏互动组件

8.11.1 功能

多屏互动组件应实现手机、平板、电视等多设备间图片和视音频多媒体内容的传送和播控操作，实现跨设备的屏幕 UI 操控，应支持 UPNP、DLNA 和 Miracast 等协议。

8.11.2 组件实现和调用方式

多屏互动组件应按照组件模型实现，组件调用方式如图40所示。

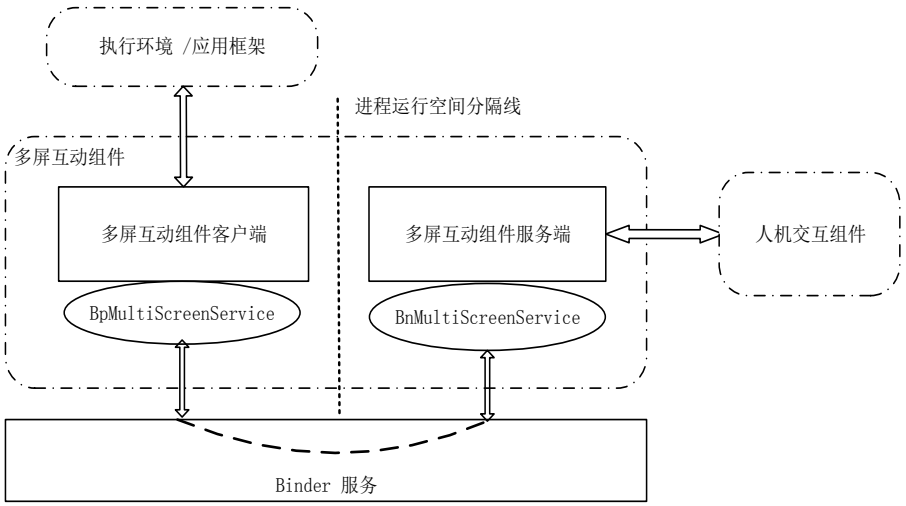


图 40 多屏互动组件模型图

图 40 中，BnMultiScreenService 为服务 Stub，BpMultiScreenService 为服务 Proxy。

8.11.3 功能架构与模块

多屏互动组件包括设备发现模块、设备连接模块、设备控制模块和跨屏播控模块如图 41 所示。

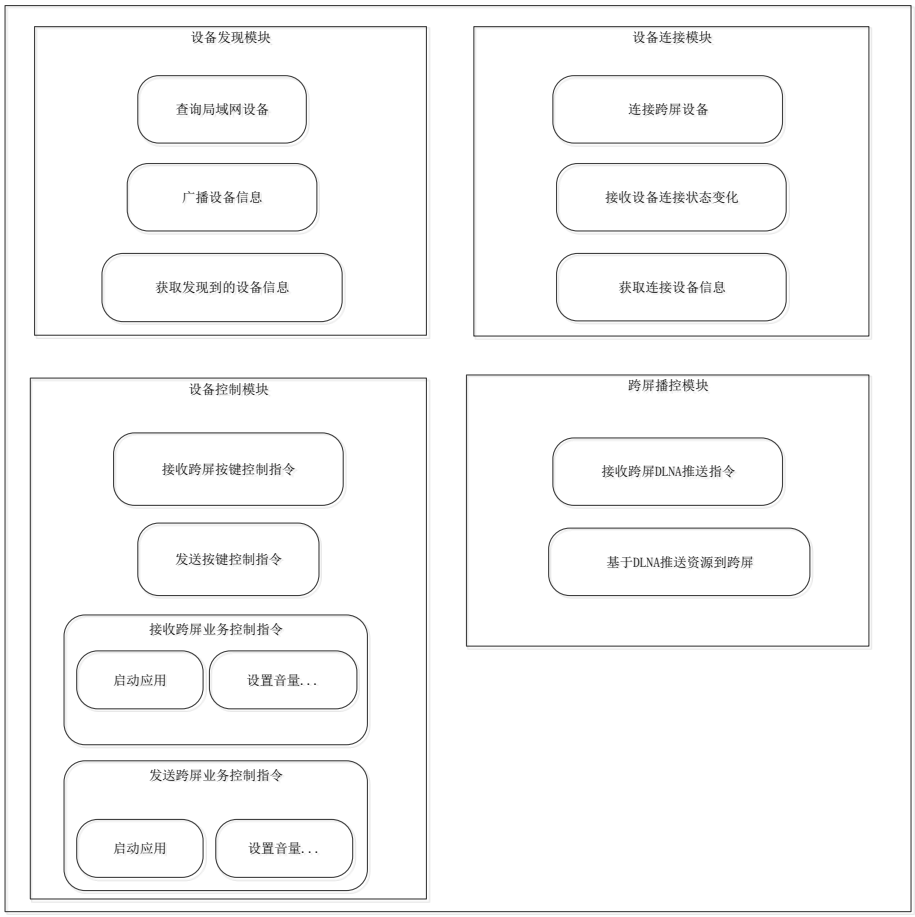


图 41　多屏互动组件功能架构

设备发现模块负责实现设备发现功能，包括主动发现其他设备以及被其他设备发现的功能，存储所发现设备的相关信息并为其他软件模块提供查询接口，支持 UPNP 等协议。

设备连接模块负责实现设备连接功能，接收跨屏设备的连接状态变化消息，获取连接设备的信息。支持 UPNP 等协议。

设备控制模块负责实现跨屏 UI 操控功能，包括接收跨屏的按键操控指令、发送按键操控指令到跨屏、接收跨屏的业务控制指令，以及发送业务控制指令到跨屏的功能。这些业务指令包括启动应用，设置音量等。

跨屏播控模块负责实现本地媒体文件远程播放和远程媒体文件本地播放功能，包括接收跨屏的图片，音视频推送播放指令，以及推送图片及音视频到跨屏。跨屏播控模块应支持 DLNA 等协议。

8.11.4　接口

多屏互动组件提供设备发现接口、设备连接接口、跨屏 UI 操控接口和跨屏播控接口。此部分接口属于功能组件接口。

8.11.5　与其他软件模块的协同

多屏互动组件的设备控制模块具备接收跨屏的按键操控指令功能，该功能需要调用到人机交互组件来注入虚拟按键。

8.12　终端管控

8.12.1 功能

终端管控组件应实现 TR069 协议族报文的解析与封装，实现对智能电视终端信息和参数的查询、统计、设置、监控和上报等功能，包括恢复出厂设置、终端重启设置、软件升级触发、网络诊断等：

a) 应能对 TR069 协议族报文进行解析和封装；
b) 应能查询终端的软件信息、硬件信息、网络信息、应用信息、Wi-Fi 信息、CA 卡信息等，并能及时上报相关信息查询结果；
c) 应能设置终端功能的相关控制管理参数，包括相关功能的开启、关闭、定时开启和定时关闭等控制参数，设置网络诊断触发、终端状态告警和 WIFI 网络配置等相关参数；
d) 应能控制终端的版本更新、出厂设置恢复、应用缓存清除和系统重启等操作；
e) 应能主动上报事件信息，包括按照预置的条件和时间间隔上报等；
f) 应能对终端管控 App 的注册、注销和运行状态进行管理；
g) 应能对从终端管控 App 发来的信息逐一进行数字签名校验，只有通过数字签名校验后的才能予以执行；
h) 应能对上报信息进行数字签名。

8.12.2 组件实现和调用方式

终端管控组件应按照组件模型实现，组件调用方式如图 42 所示。

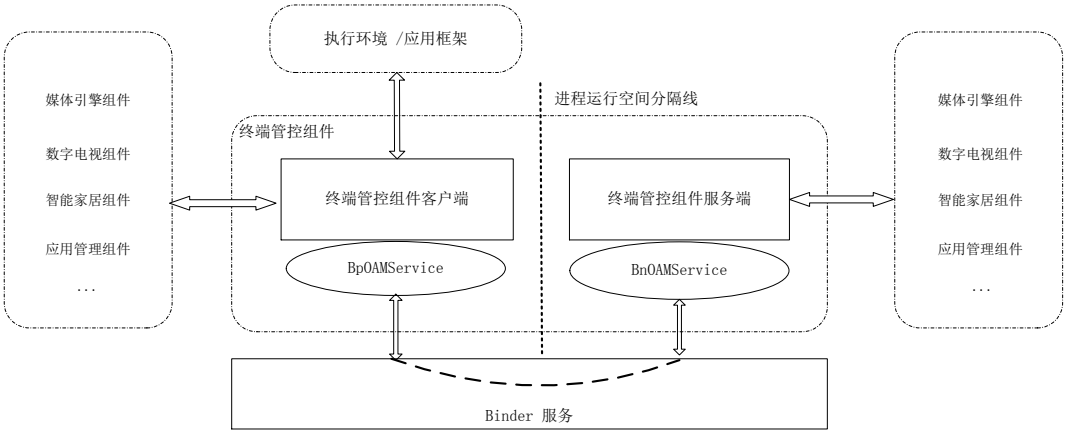


图 42 终端管控组件实现和调用方式

图 42 中，BnOAMService 为服务 Stub，BpOAMService 为服务 Proxy。

8.12.3 功能架构与模块

终端管控服务端由接收与上报、数字签名、TR069 协议族、采集与设置、管控 App 管理等模块组成，终端管控服务端架构如图 43 所示。

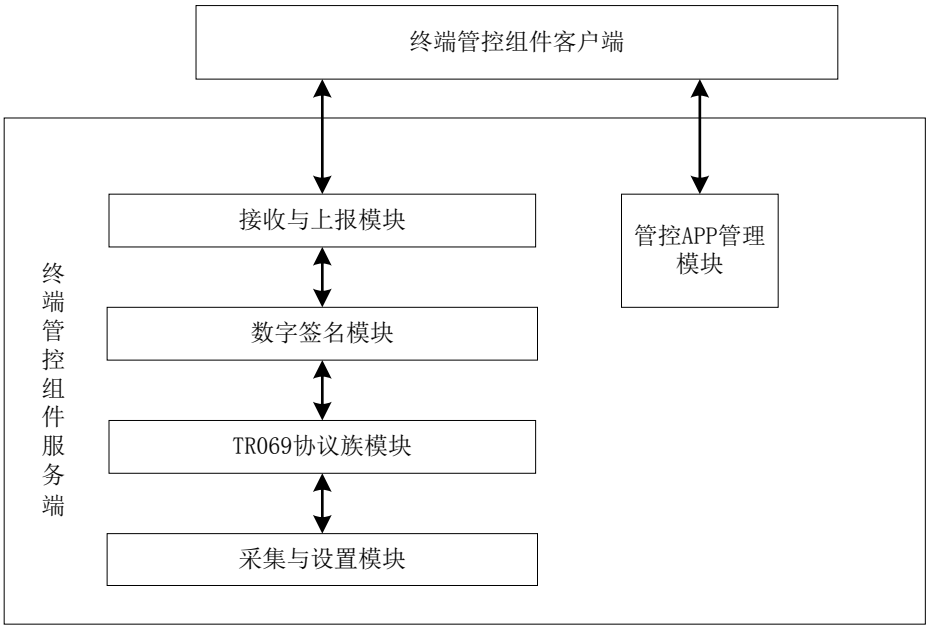


图 43　终端管控组件架构

接收与上报模块负责与终端管控 App 协同实现终端管控信息的接收和终端状态信息的上报。

数字签名模块负责实现对终端管控 App 发来的信息进行数字签名校验，负责实现对上报信息的数字签名。

TR069 协议族模块负责实现 TR069 协议族。

采集与设置模块负责与其他软件模块协同完成对终端状态参数的查询和设置。

管控 App 管理模块负责对管控 App 实施注册与注销机制。

8.12.4　接口

终端管控组件应通过客户端向管控 App 提供管控命令与信息上报的数据传送接口，同时为管控 App 提供注册与注销接口。此部分接口属于功能组件接口。

8.12.5　与其他软件模块的协同

终端管控组件应与媒体引擎组件、数字电视组件、智能家居组件、应用管理组件以及其他组件协同工作，协同关系如图44所示。

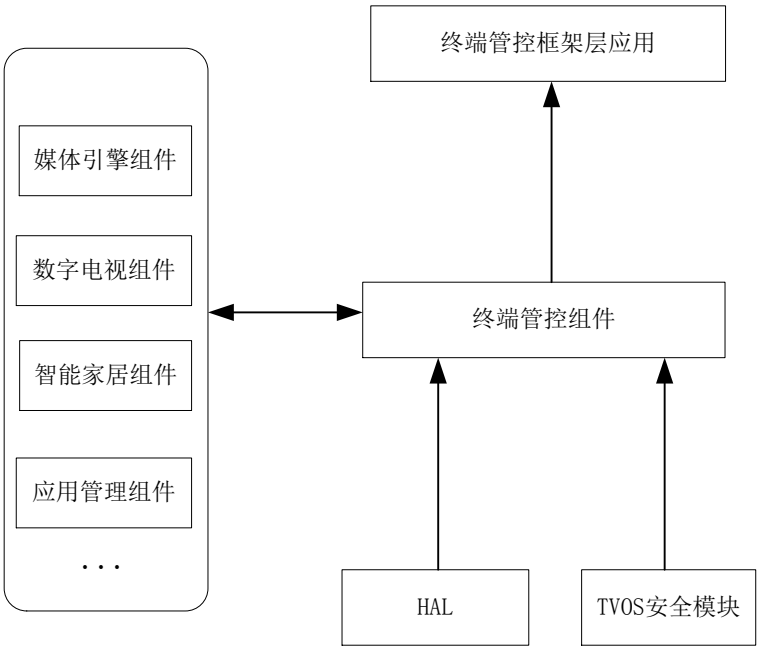


图 44　终端管控同其他组件的协同关系

8.13　数据采集组件

8.13.1　功能

数据采集是负责实现 TVOS 智能终端业务和用户行为等相关信息数据采集和上报的功能组件。

8.13.2　组件实现和调用方式

数据采集组件应按照组件模型实现，组件调用方式如图 45 所示。

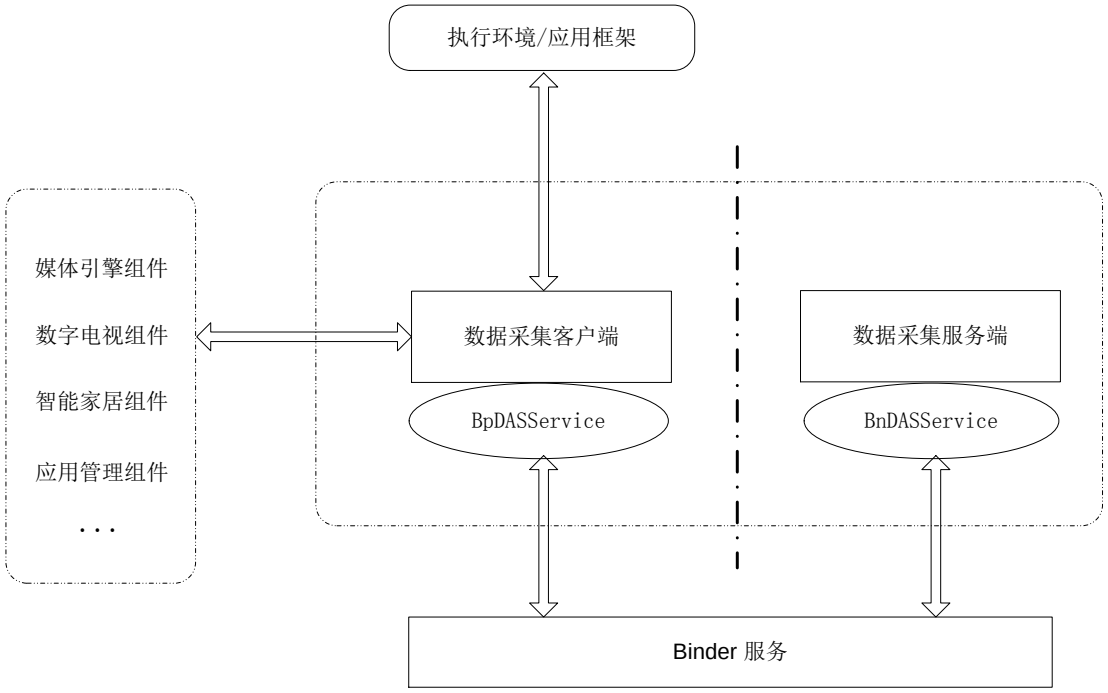


图 45　数据采集实现和调用方式

图 45 中，BnDASService 为服务 Stub，BpDASService 为服务 Proxy。

8.13.3　功能架构与模块

数据采集服务端由接收与上报、数据采集、采集 App 管理模块等模块组成，其架构如图 46 所示。

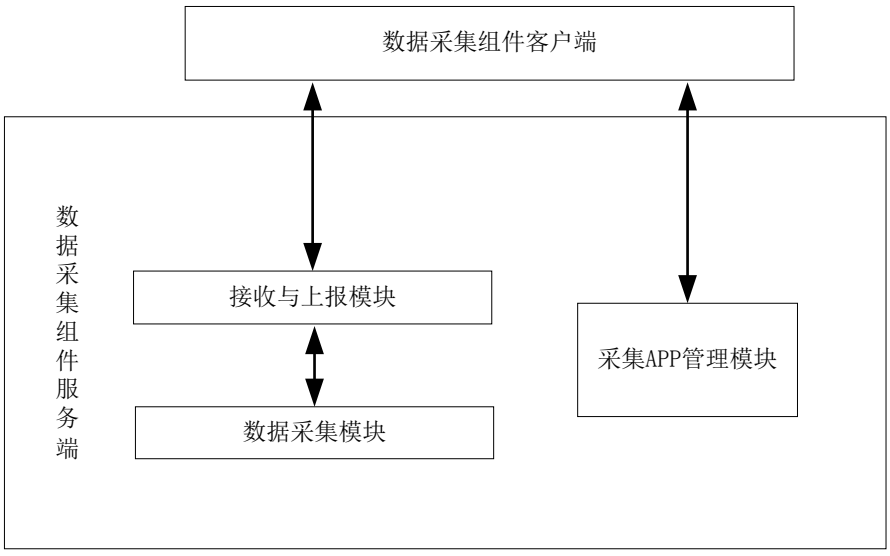


图 46　数据采集组件架构

接收与上报模块负责与数据采集 App 协同实现数据采集指令的接收和采集数据信息的上报。
数据采集模块负责与其他软件模块协同完成对终端业务的数据采集。
采集 App 管控模块负责对数据采集 App 实施注册与注销机制。

8.13.4 接口

数据采集组件应通过客户端向采集 App 提供信息传送接口，同时为采集 App 提供注册于注销接口。此部分接口属于功能组件接口。

8.13.5 与其他软件模块的协同

数据采集组件应与媒体引擎组件、数字电视组件、智能家居组件、应用管理组件以及其他组件协同工作，协同关系如图47所示。

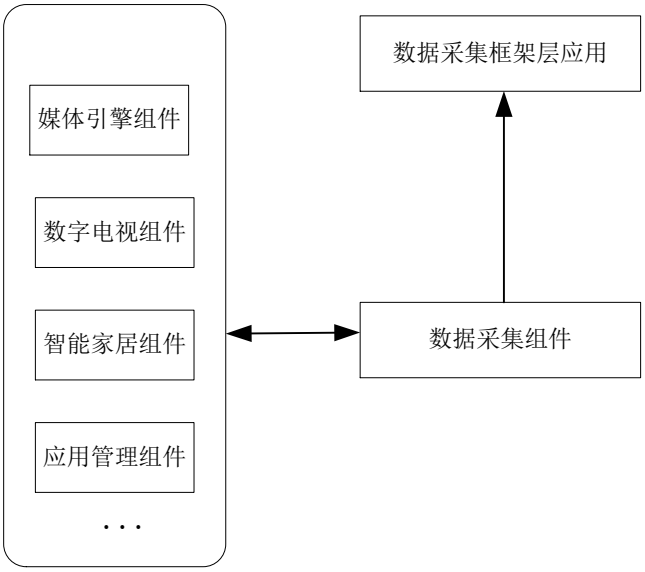


图 47 数据采集同其他组件的协同关系

8.14 广播信息服务组件

8.14.1 功能

广播信息服务组件应与数字电视组件协同，实现广播信息服务的监测、接收和处理，支撑应急广播、信息服务、广告、OSD 文本更新等相关业务。

广播信息服务组件应能通过 DTV 组件监控 BAT 表、NIT 表和过滤广播信息服务的 Section 数据。

广播信息服务组件应能通过 DCAS 组件获取终端对应的 CA 用户标识，并以此为判据，实现终端应急广播和广告等信息的精准接收。

8.14.2 组件实现与调用方式

广播信息服务组件应按照组件模型实现，组件调用方式如图 48 所示。

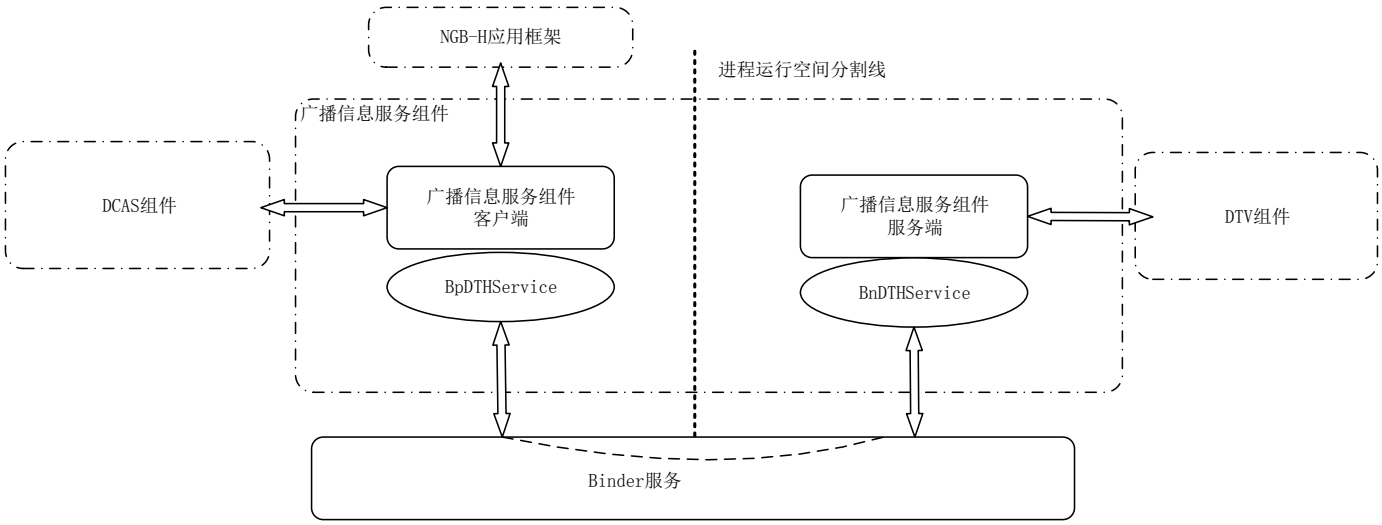


图 48　广播信息服务组件模型

图 48 中，BnDTHService 为服务 Stub，BpDTHService 为服务 Proxy。

8.14.3　功能模块与架构

广播信息服务组件应包括应急广播信令信息监测和转发，信息服务数据接收，广告内容更新，OSD 文本更新等功能模块，其结构如图 49 所示。

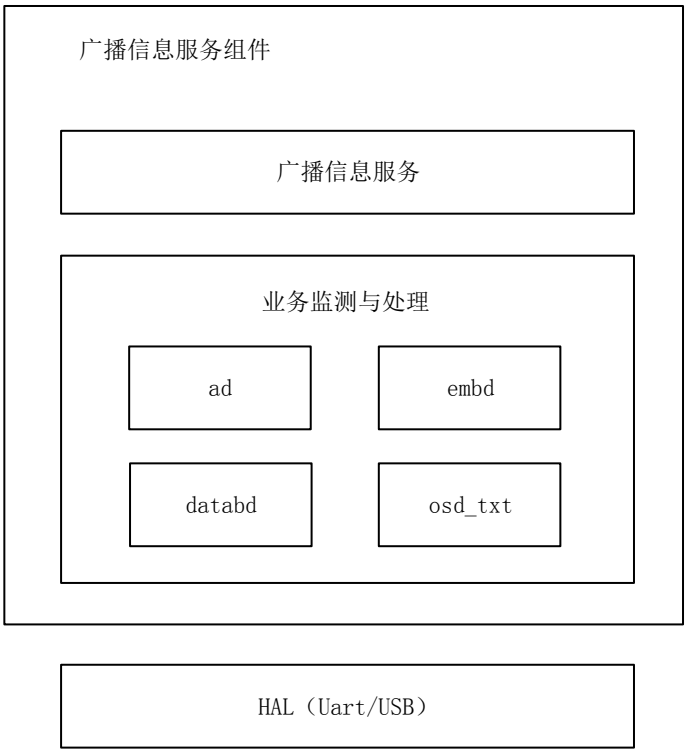


图 49　广播信息服务组件内部逻辑架构

应急广播信令信息监测和转发功能模块（embd）负责监控、接收和处理应急广播信令信息，并按所接收信令信息的要求，与其他软件模块协同，使终端切换到应急广播频道。

信息服务数据接收功能模块(databd)接收并存储基于数据广播方式传输的信息服务数据。

广告内容更新功能模块（ad）负责监控广告图片和数据更新，包括开机画面、主菜单广告位、节目浏览广告位的开机广告更新，以及节目条信息、音量条图片的实时广告更新。

OSD 文本更新功能模块（osd_txt）负责监控 OSD 更新信令信息、接收和解析 OSD 提示信息，并支撑其他软件模块实现终端界面 OSD 文本信息显示更新，包括基础提示信息、条件接收提示信息、模块提示信息、PPV 提示信息等的更新。

8.14.4 接口

广播信息服务组件必须通过广播信息服务组件客户端访问，广播信息服务组件客户端提供的访问接口如表 1 所示。

表 1 广播信息服务组件接口

序号	功能单元	说明
1	应急广播	应急广播相关接口
2	信息服务	信息服务接口
3	广告	广告数据接收状态查询及广告数据获取接口
4	OSD 更新	OSD 文本更新状态及数据获取等接口

8.14.5 与其他组件的关系

广播信息服务组件需要与 DTV 组件、DCAS 组件及底层 HAL 进行交互。对应的关系如图 50 所示。

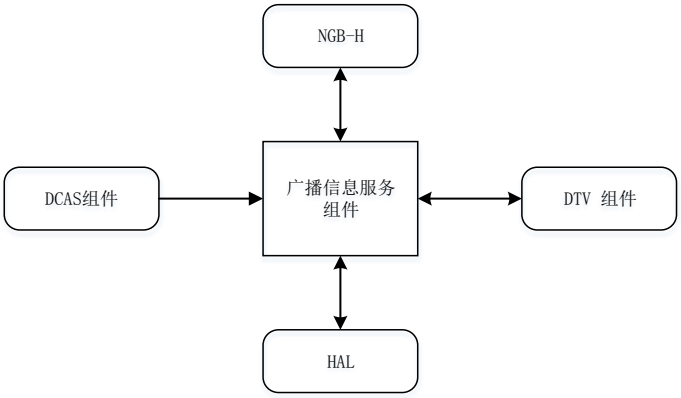


图 50 广播信息服务组件与其他组件关系

8.15 ATV 组件

8.15.1 功能

ATV 组件应实现搜台、频道管理、通道管理及 TV 相关设置参数管理等功能，为相关应用提供接口和相应能力的支撑。搜台包括自动搜台和手动搜台；频道管理包括频道切换及频道存储；通道管理包括通道切换及相关信息获取；TV 相关设置参数管理包括图像和声音等基本参数的设置及存储。

8.15.2 组件实现和调用方式

ATV组件应按照组件模型实现，组件调用方式如图51所示。

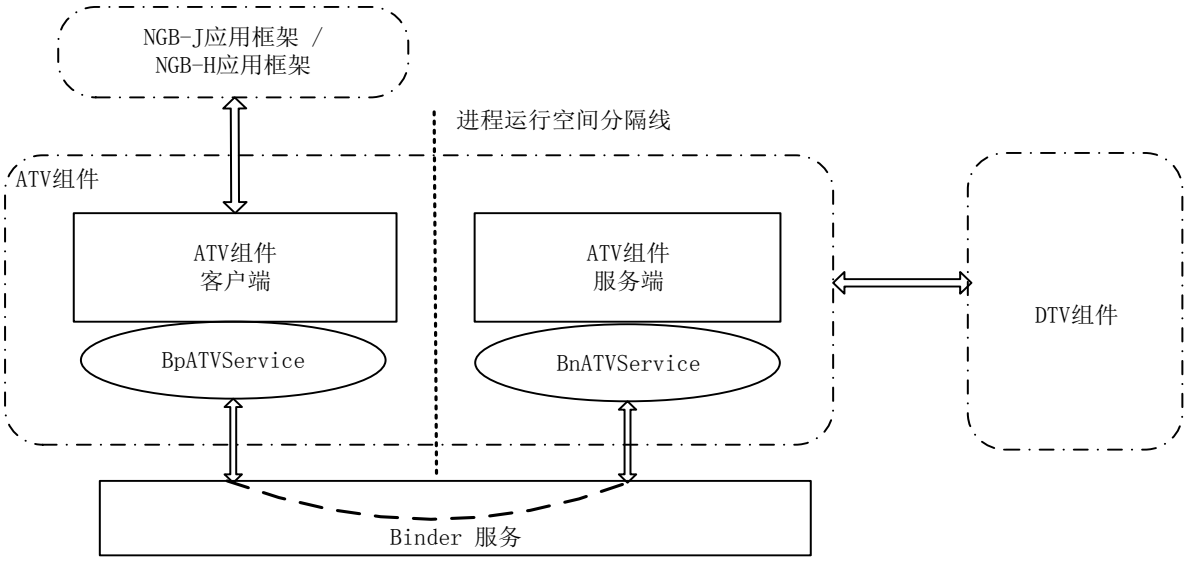


图 51　ATV 组件实现和调用方式

图 51 中，BnATVService 为服务 Stub，BpATVService 为服务 Proxy。

8.15.3　功能架构与模块

ATV 组件由 ChannelManager 模块、SourceManager 模块、TvSetting 模块及 DataManager 模块组成，ATV 组件架构如图 52 所示。

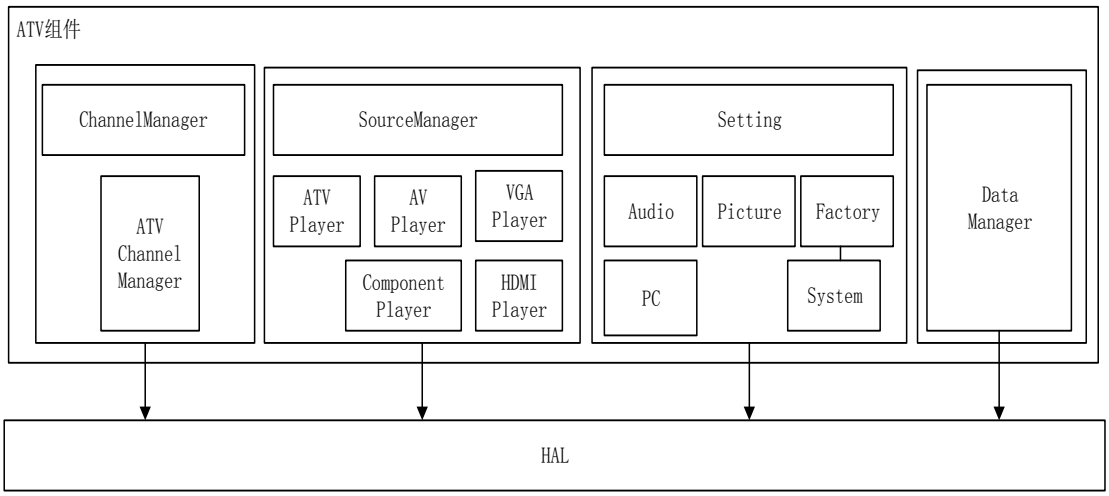


图 52　ATV 组件逻辑架构

ChannelManager 负责实现 ATV 自动/手动搜台、频道存储及管理。

SourceManager 负责实现对 source 的切换，包含 ATV、AV、VGA、Component、HDMI 等 source。SourceManager 作为 source 的核心控制管理者，实现对 source 的生命周期的控制和管理，并根据需求对传入参数进行解析，确定切换到的 source，将该 source 唤醒，同时将正在使用的 source 销毁，并记录当前 source 和 presource，供管理使用和支持 source 的查询，完成后将控制权交给唤醒的 source。

TvSetting 负责实现 TV 相关 Audio、Picture、PC、Factory、System 参数的设置与获取功能。Audio 主要包括音量、声音模式等声音相关设置；Picture 主要包括亮度、对比度、清晰度、饱和度等图像相关

设置；Pc 主要包括相位、时钟等 PC 相关设置。Factory 主要包括工厂模式相关设置；System 主要包括 I2C、GPIO 等系统相关设置。

DataManager 负责实现与 TV 相关的数据存储的功能。存储数据主要包括 Audio、Picture、PC、Factory、System、Source 及 ATV 相关参数。

8.15.4 接口

ATV 组件通过 ATV 组件客户端向外提供调用接口，所提供的访问接口主要有如下几部分：

a) Source 控制接口是 ATV 组件为上层软件模块和其他组件提供通道切换和状态查询接口；
b) Channel 控制接口是 ATV 组件为上层软件模块和其他组件提供频道搜索和频道切换相关接口；
c) TvSetting 接口是 ATV 组件为上层软件模块和其他组件提供图像、声音、系统、工厂等相关设置接口。

此部分接口属于功能组件接口。

8.15.5 与其他软件模块的协同

ATV 组件主要与系统中数字电视组件和 NGB-J 和 NGB-H 应用接口单元协同工作，完成电视相关的业务功能，各模块的协同关系如图 53 所示。

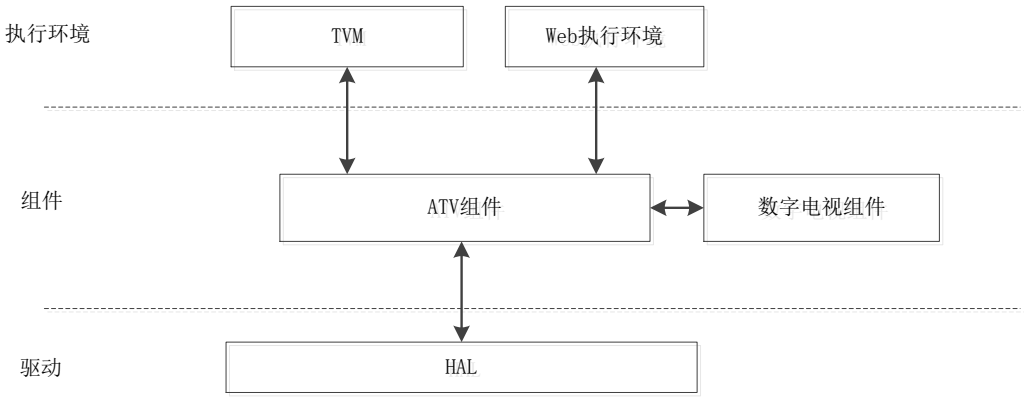


图 53 ATV 组件与其他软件模块间关系

ATV 组件与其他软件模块关系描述如下：

a) 与 NGB-J 和 NGB-H 的协同：NGB-J 和 NGB-H 调用 ATV 组件的组件接口，进行封装和转换成对应的 JAVA 接口和 JS 接口，让应用可以通过 JAVA 和 JS 接口访问 ATV 组件的相关功能；
b) 与数字电视组件的协同：主要通过调用数字电视组件的 Player 接口实现数字电视通道与其他通道的切换；
c) ATV 组件与 HAL 层的协同：ATV 组件通过 HAL 层的接口实现对底层硬件功能的调用。

8.16 应用安装组件

8.16.1 功能

应用安装组件应实现对 JAVA 和 WEB 应用程序包的解析；实现 JAVA 和 WEB 应用程序包的安全验证，包括完整性和合法性验证及权限审查等；实现 JAVA 和 WEB 应用的安装、卸载和更新等功能；提供对 JAVA 和 WEB 应用安装信息的查询；支撑应用管理组件实现对已安装应用的管理。

8.16.2 组件实现和调用方式

TVOS 应用安装组件应按照组件模型实现，组件实现和调用方式如图 54 表示。

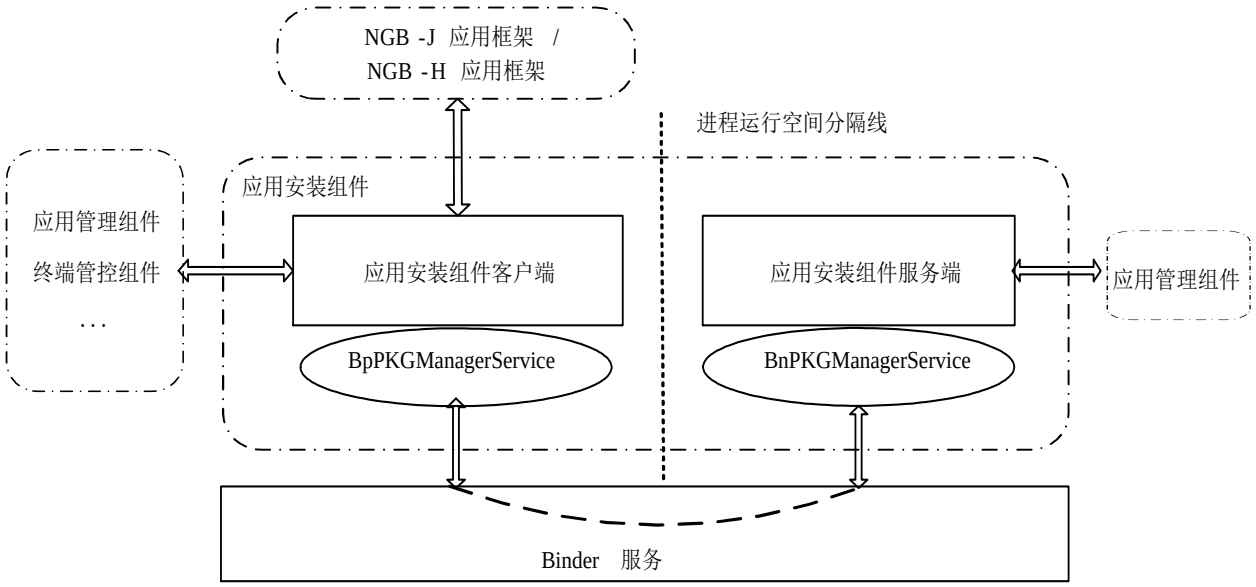


图 54 TVOS 应用安装组件实现和调用方式

图 54 中，BnPKGManagerService 为服务 Stub，BpPKGManagerService 为服务 Proxy。

8.16.3 功能架构与模块

应用安装组件由应用包解析、应用安装、应用安全和应用查询等模块组成，应用安装组件架构如图 55 所示。

图 55 应用安装组件功能架构与模块

应用包解析模块负责对待安装的应用程序包进行解析，并检查相应应用程序所需安装空间。

应用安全模块负责对待安装的应用程序进行完整性和合法性进行校验，并检查相应应用程序的权限。

应用安装模块负责应用的安装、卸载和更新。

应用查询模块负责为其他软件模块和应用对已安装应用提供信息查询。

8.16.4 接口

应用安装组件应通过客户端向其他软件模块提供 JAVA 和 WEB 应用的安装、卸载、更新等调用接口；应提供对已安装 JAVA 和 WEB 应用安装信息的查询调用接口。此部分接口属于功能组件接口。

8.16.5 与其他软件模块的协同

应用安装组件与应用管理组件形成相互依赖的关系，应用安装组件既与应用管理组件协同实现其相关功能，又为应用管理组件提供支撑。应用安装组件为其他软件模块和应用提供信息查询服务功能。

8.17 应用管理组件

8.17.1 功能

应用管理组件应实现对 JAVA 和 WEB 应用注册、注销和运行等的管理，实现对 JAVA 和 WEB 应用的启动、停止、暂停、恢复和退出等运行状态生命周期管理，包括对应用相关功能单元的运行状态生命周期管理；实现不同软件模块间和不同应用间的通信机制，包括广播消息和定向消息机制；实现不同软件模块间和不同应用间的数据共享机制，包括共享数据的提供方式、目标数据的寻找方式和跨进程高效数据获取方式，以及数据变化的通知机制；支撑应用的相关功能单元通过窗口管理组件创建相应的窗口，并支撑应用的功能单元与窗口管理组件的对应窗口建立跨进程通信；支撑应用安装组件实现应用的安装和更新。

8.17.2 组件实现和调用方式

应用管理组件应按照组件模型实现，组件实现和调用方式如图 56 所示。

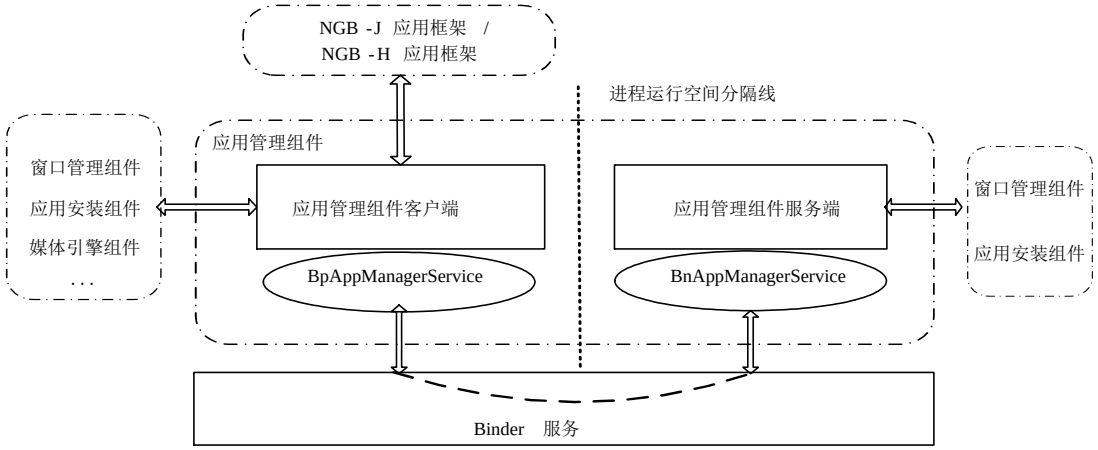


图 56 应用管理组件实现和调用方式

图 56 中，BnAppManagerService 为服务 Stub，BpAppManagerService 为服务 Proxy。

8.17.3 功能架构与模块

应用管理组件包括应用内存管理、进程管理、应用调度管理、应用生命周期管理、消息广播管理和数据共享管理等模块，应用管理组件功能架构与模块如图 57 所示。

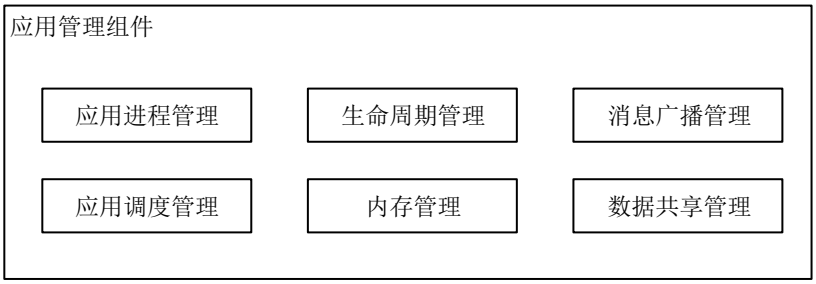


图 57 应用管理组件功能架构与模块

应用进程管理模块负责对应用的注册、注销和运行等进行管理，为应用分配唯一标识符，为应用的运分配相应的资源。

应用内存管理模块负责应用的内存申请管理和内存使用管理。

应用生命周期管理模块负责应用启动、运行、挂起、恢复以及关闭等运行状态管理及状态切换管理。

应用调度管理模块负责调度和管理不同应用的状态切换，包括激活、挂起、后台运行等。

消息广播管理模块负责为不同软件模块间和不同应用间的通信提供广播消息和定向消息机制，并对相关消息的传递进行管理。

数据共享管理模块负责实现数据共享机制，支撑和管理不同软件模块间和不同应用间的数据共享。

8.17.4 接口

应用管理组件应通过客户端提供应用基本功能单元的功能调用接口，包括具备 UI 的基本功能单元及不含 UI 的基本功能单元的调用接口；应提供系统已运行应用相关信息的查询接口。此部分接口属于功能组件接口。

8.17.5 与其他软件模块的协同

应用管理组件与应用安装组件和窗口管理组件形成相互依赖的关系，应用管理组件既与应用安装组件和窗口管理组件协同实现其相关功能，又为应用安装组件和窗口管理组件提供支撑。应用管理组件为其他组件和应用的实现和运行提供支撑。

8.18 窗口管理组件

8.18.1 功能

窗口管理组件应与应用管理组件协同实现应用程序窗口的创建，实现应用程序窗口的显示和状态管理，实现对应用程序窗口输入事件的接收及向相应应用程序功能单元的分发，实现与相关应用程序功能单元的消息传递，实现应用程序窗口的安全管理机制。

8.18.2 组件实现和调用方式

窗口管理组件应按照组件模型实现，组件实现和调用方式如图 58 表示。

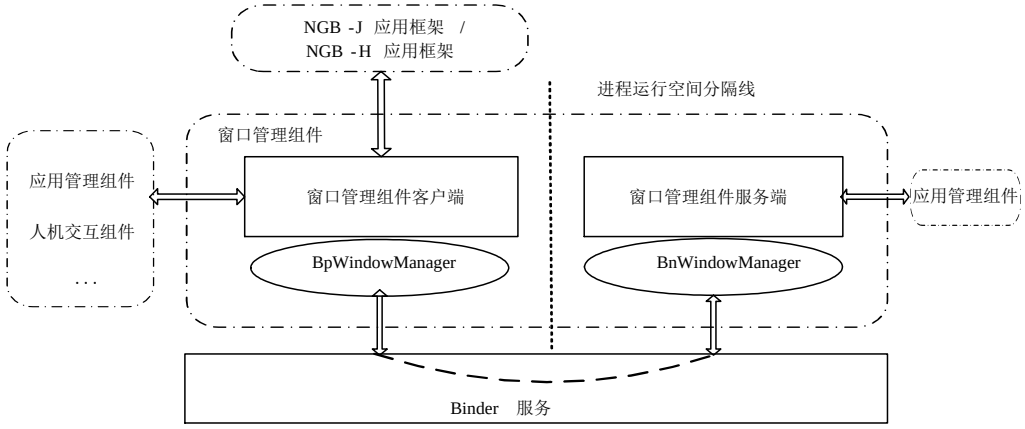


图 58 TVOS 窗口管理组件实现和调用方式

图 58 中，BnWindowManager 为服务 Stub，BpWindowManager 为服务 Proxy。

8.18.3 功能架构与模块

窗口管理组件服务端由窗口创建、窗口消息、窗口显示和窗口事件等模块组成，窗口管理组件架构如图 59 所示。

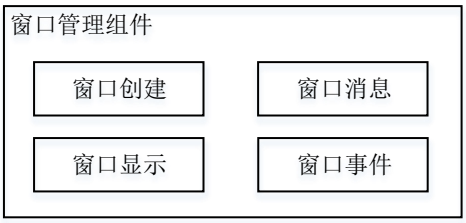


图 59　窗口管理组件架构

窗口创建模块负责实现应用程序窗口的创建。

窗口消息模块负责与应用程序相关功能单元间实现对应用程序窗口创建、销毁和重绘等消息的传递。

窗口显示模块负责实现已创建应用程序窗口的显示模式、显示次序、显示位置和显示策略等的管理。

窗口事件模块负责实现对应用程序窗口输入事件的接收，以及将已接收的窗口输入事件向相应应用程序功能单元的分发。

8.18.4　接口

窗口管理组件应通过客户端向其他软件模块提供窗口管理调用接口，包括显示管理、消息管理、安全管理、动画及系统 UI 等调用接口；应提供窗口会话管理的调用接口；应提供应用窗口添加、删除、更改和属性设置的调用接口。此部分接口属于功能组件接口。

8.18.5　与其他软件模块的协同

窗口管理组件与应用安装组件形成相互依赖的关系，窗口管理组件既与应用管理组件协同实现其相关功能，又为应用管理组件提供支撑。窗口管理组件与底层硬件相关的图层叠加和显示模块协同实现窗口的显示，从底层硬件相关的输入模块接收窗口输入事件。

9　应用执行环境

9.1　TVM

9.1.1　功能

TVM 执行环境应为 JAVA 应用及其所调用的应用框架层相关功能接口实例提供解释和运行环境，支撑 JAVA 应用的加载和运行，通过一个 JAVA 应用对应一个相应 TVM 虚拟机实例的方式实现对 JAVA 应用的进程隔离，并与应用管理组件协同实现对 JAVA 应用的启动、暂停、恢复、重启等生命周期管理，以及对 JAVA 应用资源访问的权限管理。TVM 应支持基于 JVM 和 DALVIK 的 JAVA 应用。

9.1.2　架构与实现机制

TVM Runtime由应用模型转换器、字节码转换器、Java ME支持模块和Java虚拟机等组成。TVM Runtime架构与实现机制如图60所示。

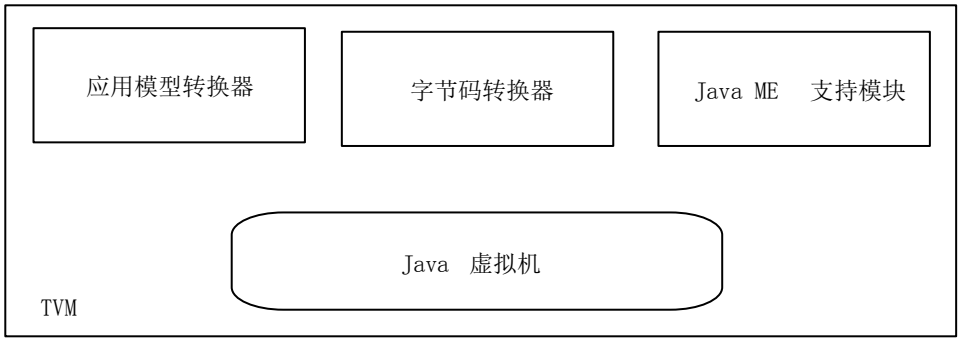


图 60 TVM 架构

应用模型转换器负责将 JAVA ME Xlet/MIDlet 应用模型自动转换为 Android 应用模型，并将.jar 和.jad 形态的 JAVA 应用包转换为.apk 形态的 JAVA 应用包。

字节码转换器负责将标准的 JAVA 字节码转换为 Dalvik 字节码，以供 TVM Java 虚拟机使用。

JAVA ME 支持模块负责实现基于 JSR 规范的相关运行环境配置等功能，为 Java ME Xlet/MIDlet 应用的运行提供支撑，支持 CDC1.1.2、FP1.1.2、PBP1.1.2 和 MIDP2.0 规范。

JAVA 虚拟机负责支持运行 Dalvik 可执行格式 JAVA 应用程序，并且能用单独的虚拟机实例支持一个相应的 JAVA 应用进程。

9.2 Web Runtime

9.2.1 功能

Web Runtime 应与 H5 引擎协同实现对 WEB 应用运行生命周期的管理功能，包括 WEB 应用的加载、启动、挂起和销毁等，实现对 WEB 应用资源访问的权限管理功能，包括 WEB 应用的权限检查、权限申请、权限修改等，实现对 WEB 应用隔离等应用安全管理功能，实现 WEB 应用策略管理，包括应用资源分配、应用进程驻留方式等。

9.2.2 架构与实现机制

Web Runtime 与 H5 引擎组件密切协同，共同创建并为 WEB 应用提供运行执行环境，实现对 WEB 应用的运行管理、权限管理、安全管理和策略管理等功能。

Web Runtime 由应用运行管理模块、权限管理模块、安全管理模块和策略管理模块组成。

应用运行管理模块负责创建 H5 引擎组件服务端运行实例，形成 WEB 应用的基础运行环境，将 WEB 应用加载到基础运行环境中，并启动 WEB 应用的运行，同时对 WEB 应用运行生命周期进行管理。

权限管理模块在 WEB 应用运行时负责对 WEB 应用资源访问的权限进行管理。

安全管理模块负责对不同的 WEB 应用在运行时进行安全管理，包括进程隔离和数据隔离等。

策略管理模块负责对 WEB 应用运行方式进行策略管理，包括为不同应用配置独占进程或共享进程的 H5 引擎组件运行方式等。

Web Runtime 架构如图 61 所示。

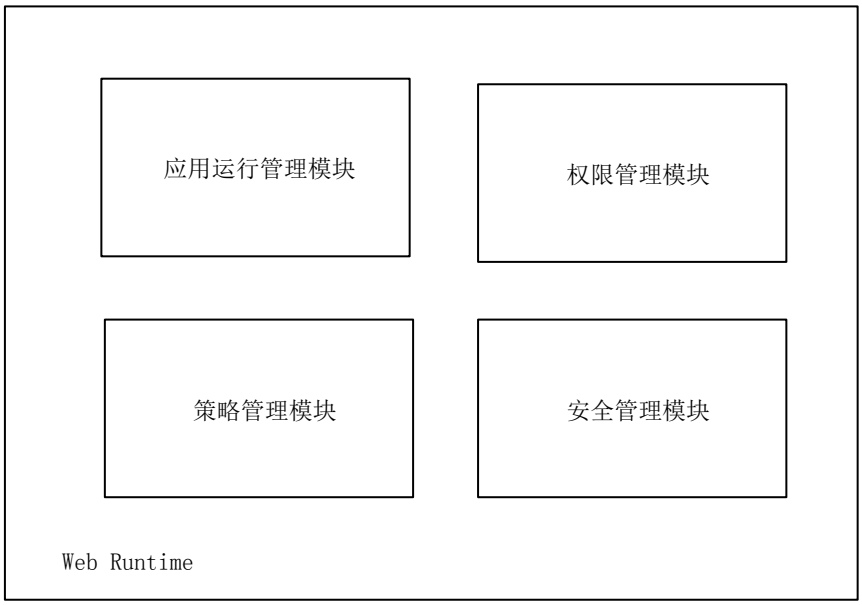


图 61　Web Runtime 架构

Web Runtime与H5引擎组件协同机制如图62所示。

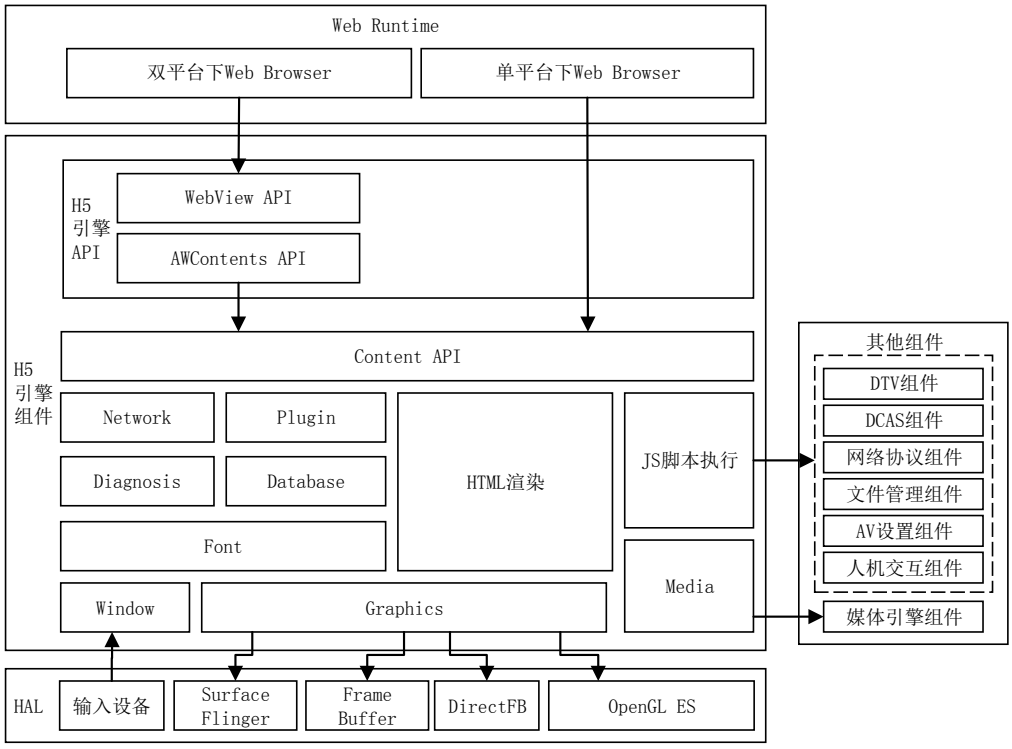


图 62　Web Runtime 与 H5 引擎组件协同机制

10　应用框架

10.1　Java 应用框架

10.1.1 Java 应用框架架构

Java 应用框架由 NGB-J 功能接口单元和扩展功能接口单元组成。

10.1.2 NGB-J 功能接口单元

10.1.2.1 功能

NGB-J功能接口单元应对各功能组件模块的接口进行JNI封装，并以Java对象的方式向JAVA应用提供调用接口，支撑应用实现EPG节目指南、频道列表、电视节目播放等数字电视相关业务功能。NGB-J功能接口单元主要包括DAVIC单元、单向广播网络接入单元、广播协议处理单元、双向宽带接入单元、人机交互单元、AV设置单元、媒体处理单元、消息管理单元和应用引擎单元等。

10.1.2.2 主要功能接口单元

主要功能接口单元如下：

a) DAVIC 功能接口单元

协同DTV组件，实现数字电视业务相关的基本对象以及异常处理，支撑数字电视相关应用运行，遵循DAVIC 1.4.1，包括Mpeg、DVB、sections、net、net.dvb、net.tuning、resources等。详细接口见GY/T 267—2012的附录S。

b) 单向广播功能接口单元

协同DTV组件，实现有线数字电视下的单向广播网络接入，包括频点的频率、调制方式、符号率等参数控制及信号强度和质量等信息的获取。详细接口见GY/T 267—2012的附录G。

c) 双向宽带网络功能接口单元

协同网络服务，实现双向宽带网络相关接入，包括双向网络的连接管理和数据操作等。详细接口见GY/T 267—2012的附录C。

d) 人机交互功能接口单元

协同人机交互组件，实现人机交互的输入和输出功能，输入把包括遥控器、鼠标、键盘、前面板按键等输入设备发送的用户指令封装成按键消息，输出通过前面板或显示屏幕反馈信息。详细接口见GY/T 267—2012的附录J。

e) 设置功能接口单元

协同AV设置组件，实现音视频输出参数设置功能，包括音频输出的端口状态、声道类型、全局音量和音量状态等，视频输出的端口状态、窗口匹配模式、亮度、对比度、饱和度、制式和透明度等。详细接口见GY/T 267—2012的附录L。

f) 媒体播放功能接口单元

协同媒体组件，实现媒体播放功能，包括播放、控制、语言选择、事件处理、异常处理等。详细接口见GY/T 267—2012的附录M。

g) 消息管理功能接口单元

实现消息的管理和分发，包括消息事件、消息事件监听器和消息管理器三个功能模块，支撑应用程序实现消息的监听和获取。详细接口见GY/T 267—2012的附录N。

h) 应用引擎功能接口单元

协同DTV组件，实现数字电视业务基础功能，包括节目搜索、节目指南、信息搜索等，支撑数字电视相关应用的运行。详细接口见GY/T 267—2012的附录Q。

i) DCAS 功能接口单元

协同DCAS组件，实现DCAS应用管理、ECM/EMM数据处理和DCAS数据交互等功能，支撑数字电视应用播放DCAS加密节目。详细接口见GY/T 255—2012的附录C。

j） 终端管控数据采集单元

协同终端管控组件和其他组件，作为后台服务获取其他服务和组件的参数的获取，并获取到的数据提供给终端管控和数据采集组件。

10.1.2.3 与功能组件的协同

框架层的NGB-J接口依赖组件层各组件提供功能支持，通过JNI调用组件层各组件C层客户端接口实现，其接口关系如图63所示。

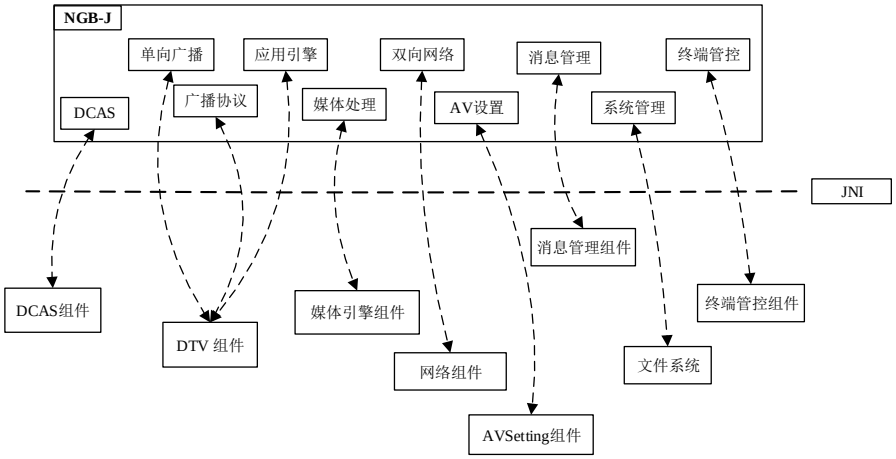


图 63 框架层的 NGB-J 接口

10.1.3 扩展功能接口单元

TVOS Java应用框架层接口应扩展支持Android API。此部分接口属于应用编程接口。

10.2 Web 应用框架

10.2.1 WEB 应用框架架构

Web应用框架由H5功能接口单元和NGB-H功能接口单元组成。

10.2.2 H5 功能接口单元

H5功能接口单元应能支持HTML5（http://www.w3.org/TR/html5/）、CSS（https://www.w3.org/standards/techs/css#stds）、JavaScript（ECMA-262）和DOM（https://www.w3.org/TR/DOM-Level-2-HTML/）等不同类型的接口。

HTML5接口单元应支持HTML5标准中定义的接口。

CSS接口单元应支持CSS3标准中定义的接口。

JavaScript接口单元应支持JavaScript1.9标准中定义的接口。

DOM接口单元应支持DOM2标准中定义的接口。

此部分接口属于应用编程接口。

10.2.3 NGB-H 功能接口单元

10.2.3.1 NGB-H 功能

NGB-H功能接口单元应对各功能组件模块的接口进行JS接口封装，以JS对象的方式向WEB应用提供调用接口，支撑应用实现EPG节目指南、频道列表、电视节目播放等数字电视相关业务功能。NGB-H功能接口单元主要包括单向广播网络接入单元、广播协议处理单元、双向宽带接入单元、人机交互单元、AV设置单元、媒体处理单元、消息管理单元、应用引擎单元、DCAS单元和广播信息服务单元等。

NGB-H 功能接口单元应支持 GY/T 267—2012 中 NGB-H 部分定义的接口。

10.2.3.2 功能接口单元

NGB-H 的功能接口包含有如下单元：

a) 单向广播网络接入单元，负责实现与单向广播网络接入相关的功能模块，如 DVB 调谐解调等；
b) 广播协议处理单元，负责实现广播协议处理；
c) 双向宽带网络接入单元，定义了与双向宽带网络接入控制相关的功能模块：
d) 人机交互单元，包含设备输入控制以及面板等显示控制功能；
e) AV 设置单元，进行音视频参数获取及设置操作；
f) 媒体处理单元，对音视频以及图片等多媒体文件进行处理；
g) 消息管理单元，捕获系统消息、应用层消息并处理；
h) 系统管理单元，包含数据管理、文件管理、OTA 升级管理等跟系统相关的模块；
i) 应用管理单元，对应用下载运行进行管理控制；
j) 应用引擎单元，提供频道管理、信息搜索、节目指南管理等功能；
k) DCAS 单元，负责提供 DCAS 应用相关的注册、过滤器设置、与 TApp 通讯等功能；
l) 广播信息服务单元，负责提供广播信息服务相关的监测、接收和处理等功能接口。

此部分接口属于应用编程接口。

10.2.3.3 与功能组件的协同

NGB-H 功能接口单元通过 H5 引擎组件实现其他功能组件的 client 接口，与相关的组件中定义 server 服务接口对应。

NGB-H 框架的相关接口实现依赖于各组件对外的接口功能定义，如 DvbTune 对象需要跟 DTV 组件交互；Mediaplayer 对象需要与媒体引擎组件交互，与各组件的关系如图 64 所示。

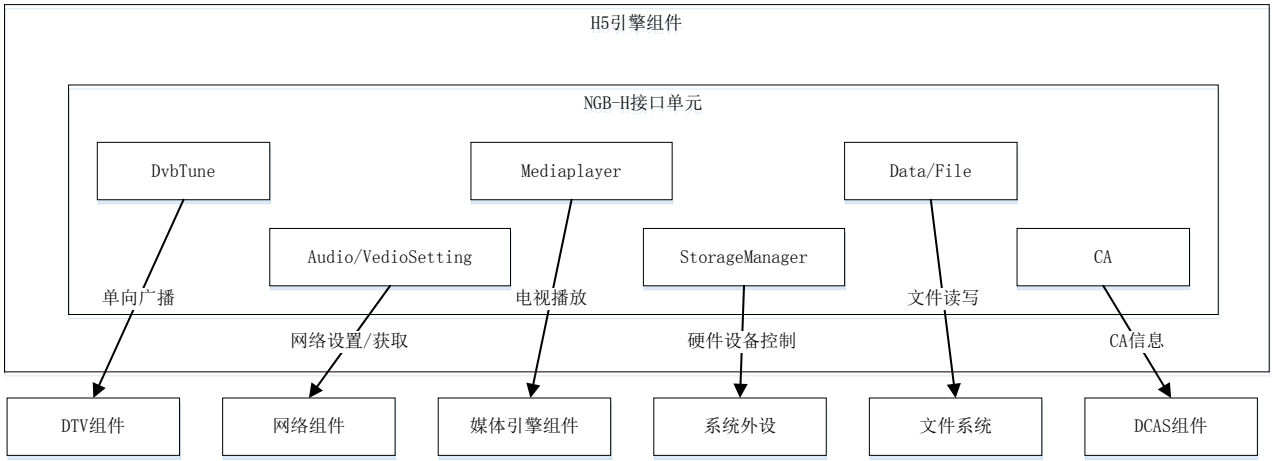


图 64 NGB-H 与其他组件接口关系

11 系统功能实现

TVOS系统启动、DTV直播、DTV点播、DRM系统功能、安全支付、智能家居、多屏互动、终端管控和数据采集、应用管理、窗口管理和应用安装等系统功能的实现参见附录B。

附 录 A
（资料性附录）
TVOS 代码树

TVOS 代码树采用多级目录结构，不同功能的 TVOS 软件代码存放在不同的目录下。

TVOS 代码树主要包括 app、component、device、framework、kernel、platform、runtime、test 八个一级目录：

a) app 目录主要用来存放 TVOS 的基础应用程序，如 DTV 直播应用程序、VOD 点播应用程序，launcher 桌面应用程序等；
b) component 目录主要用来存放 TVOS 的核心功能组件，如 atv 模拟电视功能组件、dtv 数字电视功能组件、gstreamer 媒体处理功能组件、dcas 条件下载 CA 功能组件、DRM 功能组件等；
c) device 目录主要用来存放和底层芯片平台相关的 HAL 接口以及和芯片平台编译相关的目录，如何智能卡相关的 HAL 接口、前端 source HAL 接口、音视频输出 HAL 接口等以及海思芯片、其他芯片所特有的编译工具；
d) framework 目录主要用来存放和数字广播电视相关的 NGB-H、NGB-J 功能模块；
e) kernel 目录主要用来存放 linux kernel 功能模块；
f) platform 目录主要用来存放 TVOS-C 双平台和 TVOS-H 单平台相关的平台代码；
g) runtime 目录主要用来存放运行虚拟机相关的功能模块，包含兼容 java 应用的 TVM 和兼容 WEB 应用的 Blink\V8；
h) test 目录主要用来存放和系统功能测试相关的代码。

TVOS-C 软件代码和 TVOS-H 软件代码应能置于同一 TVOS 代码树下，TVOS-C 软件代码和 TVOS-H 软件代码不重用的部分分别放在 platform 目录中对应的子目录 tvos-c 和 tvos-h 下；TVOS-C 软件代码和 TVOS-H 软件代码重用的部分按照代码树目录分类要求分别放在其他相应的目录下，其中，TVOS-C 软件代码是指同时支持 JAVA 应用和 WEB 应用的 TVOS 软件版本代码，TVOS-H 软件代码是指仅支持 WEB 应用的 TVOS 软件版本代码。

TVOS 代码树结构如图 A.1 所示。

```
├─── app          （TVOS 应用程序软件代码）
│    ├─── dtv
│    ├─── launcher
│    └─── vod
├─── component    （TVOS 核心功能组件软件代码）
│    ├─── atv
│    ├─── dcas
│    ├─── drm
│    ├─── dtv
│    ├─── gateway
│    ├─── gstreamer
│    ├─── multiscreen
│    ├─── secureos
│    └─── weblink
```

```
├─── device      （TVOS HAL 和芯片硬件平台相关软件代码）
│    ├─── hal
│    ├─── hisilicon
│    └─── mstar
├─── framework  （TVOS NGB-H 和 NGB-J 功能接口单元软件代码）
│    ├─── ngb-h
│    └─── ngb-j
├─── kernel      （linux kernel 软件代码）
├─── platform    （TVOS-C 和 TVOS-H 平台特有软件代码）
│    ├─── tvos-c
│    ├─── tvos-h
├─── runtime     （TVOS 执行环境相关软件代码）
└─── test        （TVOS 系统功能测试相关软件代码）
```

图 A.1　TVOS 代码树结构

TVOS-C 和 TVOS-H 平台相关代码树结构如图 A.2 和图 A.3 所示。

```
├─── platform
│
tvos-c
                ├─── abi
                ├─── art
                ├─── bionic
                ├─── bootable
                ├─── build
                ├─── cts
                ├─── dalvik
                ├─── developers
                ├─── development
                ├─── docs
                ├─── external
                ├─── frameworks
                ├─── hardware
                ├─── libcore
                ├─── libnativehelper
                ├─── Makefile
                ├─── ndk
                ├─── packages
                ├─── pdk
                ├─── prebuilds
                ├─── sdk
                ├─── system
                └─── tools
```

图 A.2　TVOS-C 子目录代码树结构

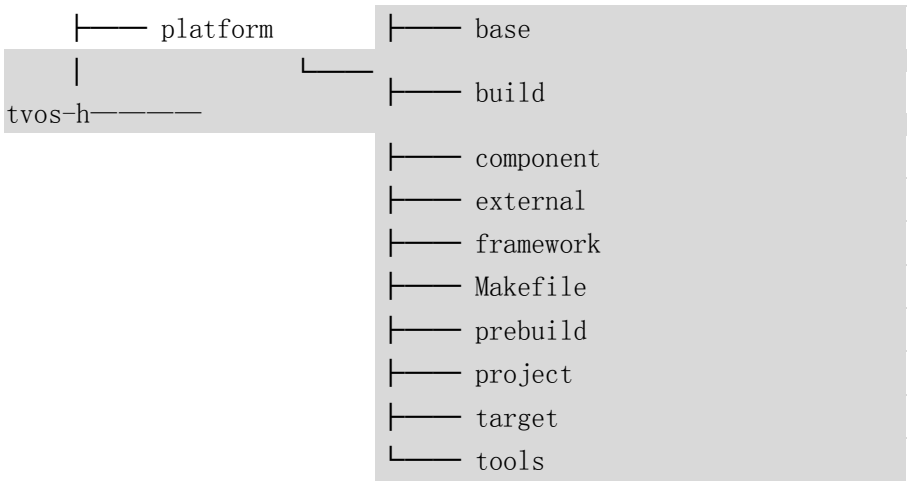


图 A.3　TVOS-H 子目录代码树结构

附 录 B
（资料性附录）
系统功能实现

B.1 系统启动

系统启动流程如图 B.1 所示。

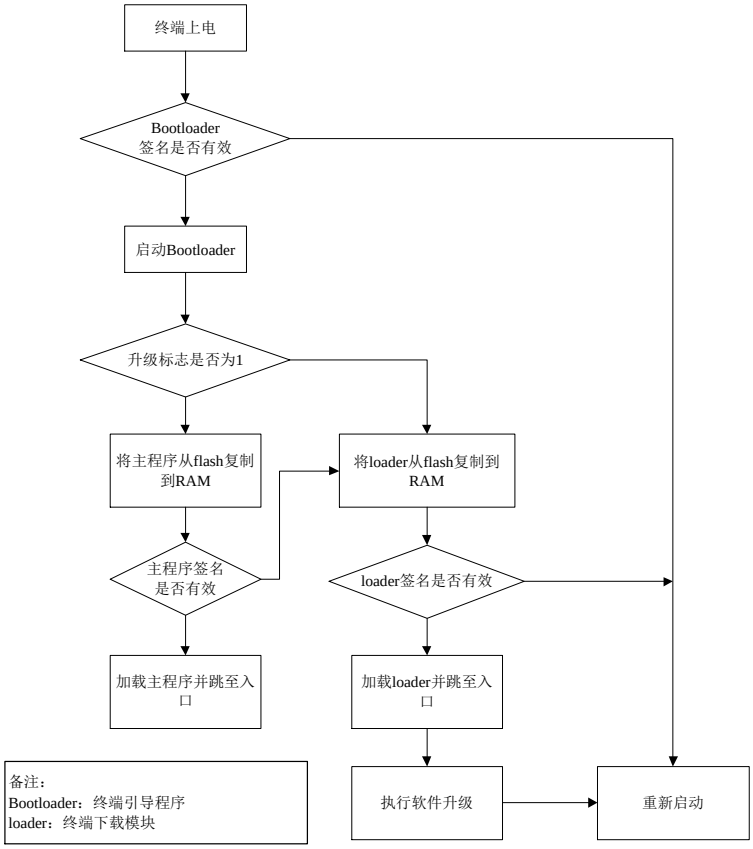


图 B.1 系统启动流程

TVOS 终端在运行前检查终端程序是否符合运营商制定的安全签名规范，保证终端能在安全的环境下正确启动运行程序。终端上电后，先启动 bootloader 引导模块。bootloader 需要进行签名校验，校验通过后根据升级标志判断是否进入 loader 执行系统升级功能。升级有两种模式，OTA 下载（通过 cable）或 IP 下载。所有的升级流必须经过签名后才能下发，进入升级后先下载数据，下载完成后先校验签名才进行擦写和程序烧录（具体擦写位置详见 Flashmap 参考表）。如果不需要升级就校验应用程序签名，校验通过后再启动应用程序。

程序的安全性依靠模块的签名和校验来保证。所有的升级文件，包括系统升级，内置应用升级，第三方应用升级都需要按照运营商的要求签名保证安全，升级软件时必须校验签名，通过以后才能进行升级。运行在终端的程序各分区也要按照运营的要求进行签名，并在程序启动时进行签名校验，如果校验不通过则自动重启终端。

B.2 DTV直播功能实现

B.2.1 基于JAVA的系统功能实现

DTV 基于 JAVA 的应用在接收到频道切换指令或节目播放指令的处理流程如图 B.2 所示。

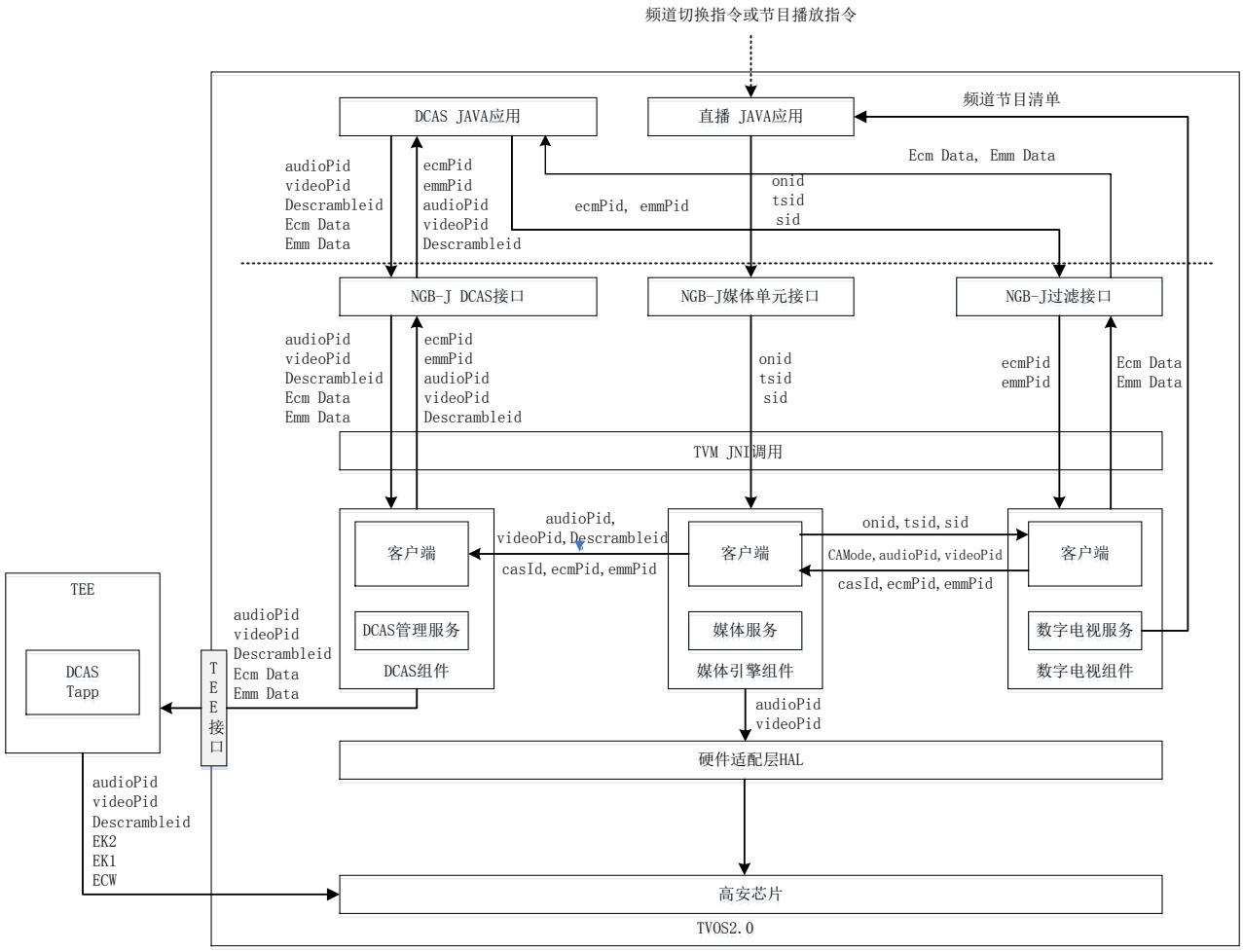


图 B.2 基于 JAVA 的系统处理流程

DTV 直播基于 JAVA 的系统功能实现如下：

a) 用户使用直播应用观看节目，切换到某个频道；直播应用调用 NGB-J 媒体单元的相应接口，并将频道的 dvb 三要素（original network id、transport stream id、service id）传给 NGB-J 媒体单元；

b) NGB-J 媒体单元将 dvb 三要素传递给媒体播放组件；

c) 媒体播放组件根据 dvb 三要素从 DTV 组件获取 freeCAMode 标识，判断频道是否加扰，如果是加扰频道，则获取 casId、ecmPid、emmPid；

d) 如果频道是非加扰的，则媒体播放组件直接调用底层驱动正常播放；如果是加扰的，则媒体播放组件创建播放的 PipeLine，并将 audioPid、videoPid 和 casId、ecmPid、emmPid 一起传给 DCAS 组件；

e) DCAS 组件根据 casId 选择相应的 DCAS 应用，并通过 DCAS API 将 audioPid、videoPid、casId、ecmPid、emmPid 传给 DCAS 应用；

f) DCAS 应用进行相关初始化，并根据 ecmPid 和 emmPid 通过 NGB-J Section 单元调用 DTV 组件获取 ecm Data 和 emm Data；

g) DCAS 应用将 ecm Data、emm Data 通过 TEE API 传给 DCAS 组件；

h) DCAS 组件将 ecm data、emm data 和对应的 audioPid、videoPid、casId 通过 TEE Client API 送到 Secure OS 中的 TA 模块；

i) TA 模块在 Secure OS 中解析 ecm data、emm data 获得 EK2、EK1、ECW 并设置给高安芯片，实现加扰频道的解扰。

B.2.2 基于Web的系统功能实现

DTV 基于 Web 的应用在接收到频道切换指令或节目播放指令的处理流程如图 B.3 所示。

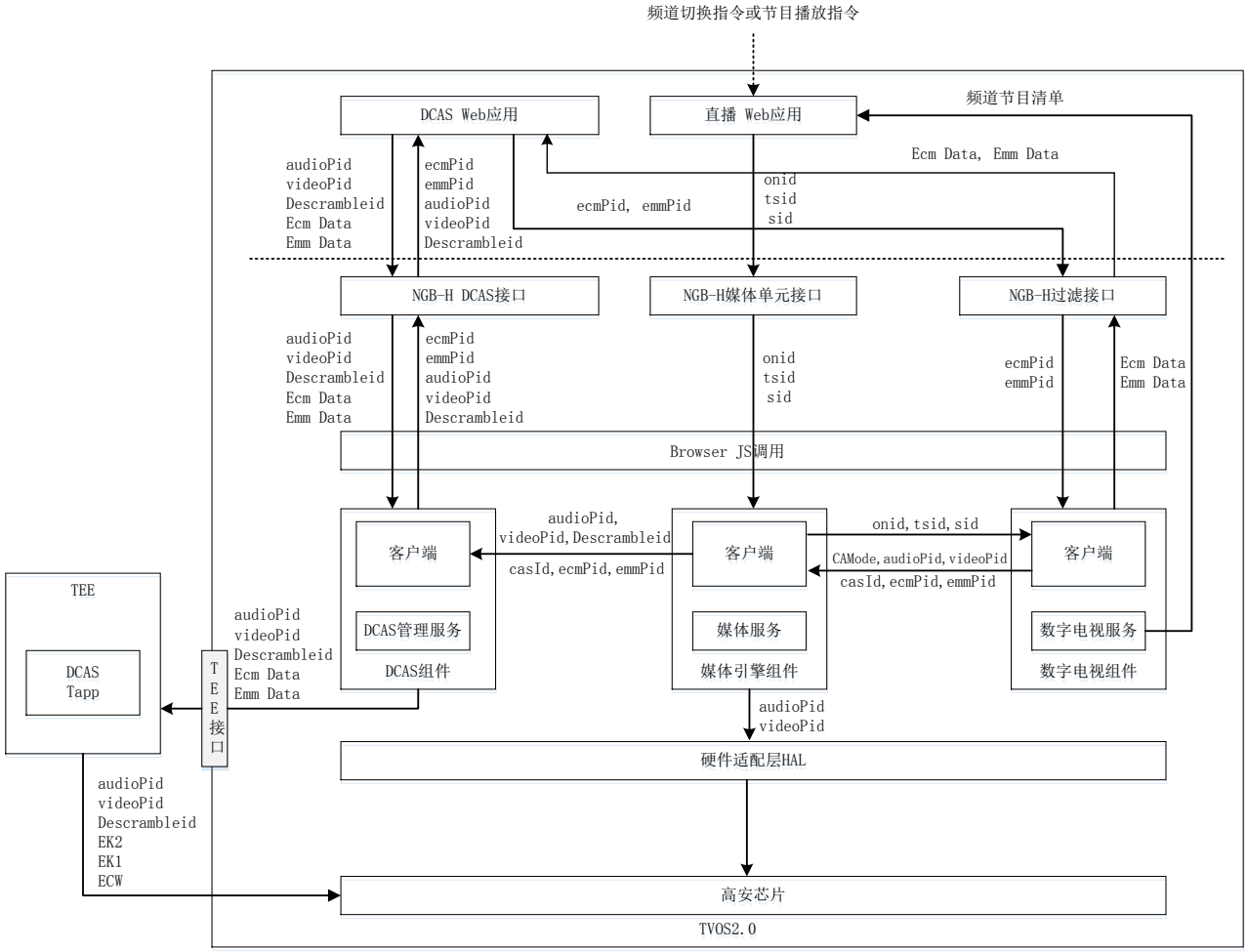


图 B.3 基于 Web 的系统处理流程

DTV 基于 Web 的系统功能实现如下：

a) 用户使用直播应用观看节目，切换到某个频道；直播应用调用 NGB-H 媒体单元的相应接口，并将频道的 dvb 三要素（original network id、transport stream id、service id）传给 NGB-H 媒体单元；

b) NGB-H 媒体单元将 dvb 三要素传递给媒体播放组件；

c) 媒体播放组件根据 dvb 三要素从 DTV 组件获取 freeCAMode 标识，判断频道是否加扰，如果是加扰频道，则获取 casId、ecmPid、emmPid；

d) 如果频道是非加扰的，则媒体播放组件直接调用底层驱动正常播放；如果是加扰的，则媒体播放组件创建播放的 PipeLine，并将 audioPid、videoPid 和 casId、ecmPid、emmPid 一起传给 DCAS 组件；

e) DCAS 组件根据 casId 选择相应的 DCAS 应用，并通过 DCAS API 将 audioPid、videoPid、casId、ecmPid、emmPid 传给 DCAS 应用；
f) DCAS 应用进行相关初始化，并根据 ecmPid 和 emmPid 通过 NGB-J Section 单元调用 DTV 组件获取 ecm Data 和 emm Data；
g) DCAS 应用将 ecm Data、emm Data 通过 TEE API 传给 DCAS 组件；
h) DCAS 组件将 ecm data、emm data 和对应的 audioPid、videoPid、casId 通过 TEE Client API 送到 Secure OS 中的 TA 模块；
i) TA 模块在 Secure OS 中解析 ecm data、emm data 获得 EK2、EK1、ECW 并设置给高安芯片，实现加扰频道的解扰。

B.3 DTV点播功能实现

用户通过JAVA点播应用或者Web网页应用点播节目的处理流程如图B.4所示，包含的具体步骤如下：

a) 如果用户使用 JAVA 点播应用点播节目，点播应用调用 NGB-J Media 单元的相应接口，并将点播节目的 URL(ngbrtsp://..)传给 NGB-J Media 单元；如果用户通过网页点播节目，Web 网页调用 NGB-H JS；
b) NGB-J Media 单元调用 mediaplayer client 接口，将播放节目的 URL 传给 Mediaplayer service 或者 NGB-H 单元调用 mediaplayer client 接口，将播放节目的 URL 传给 Mediaplayer service；
c) Mediaplayer service 根据 URL 类型创建 VOD Player，并将 URL 传给 VOD Player.
d) VOD Player 根据 URL 和服务器进行信令交互，如果是 IPQAM 模式的，和服务器信令交互成功后，从服务器获取该点播节目的频点和 serviceID;如果是 IPTV 模式，通过 IP 网络从服务器获取 TS 包；
e) 如果是 IPQAM 模式，VOD Player 根据频点和 serviceID 信息，调用 DTV 组件相关接口，获取该 serviceID 对应的节目信息，音视频的 PID、Type；如果是 IPTV 模式，VOD Player 根据从服务器获取的 TS 包，解析出所播放节目的音视频 PID、Type；
f) VOD Player 将 videoPid、audioPid、videoType、audioType、demuxID 传给 PipeLine 进行节目播放。

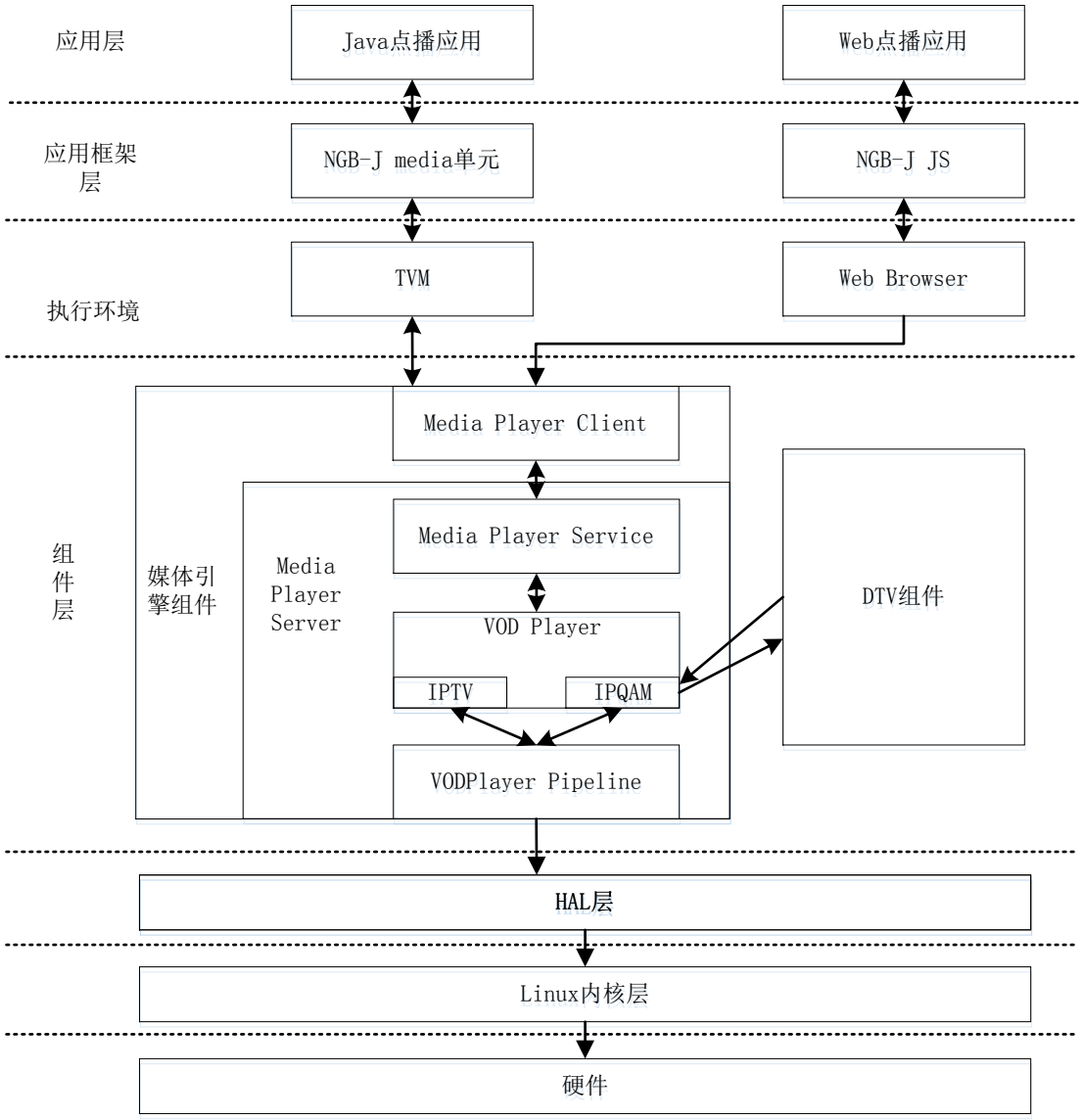


图 B.4　DTV 点播处理流程

B.4　DRM系统功能实现

B.4.1　JAVA平台实现机制

JAVA 平台中，TVOS 系统通过 DRM 应用、DRM JAVA API、DRM 组件、媒体处理组件、DRM TApp 的协同工作，实现 DRM 功能，如图 B.5 所示。

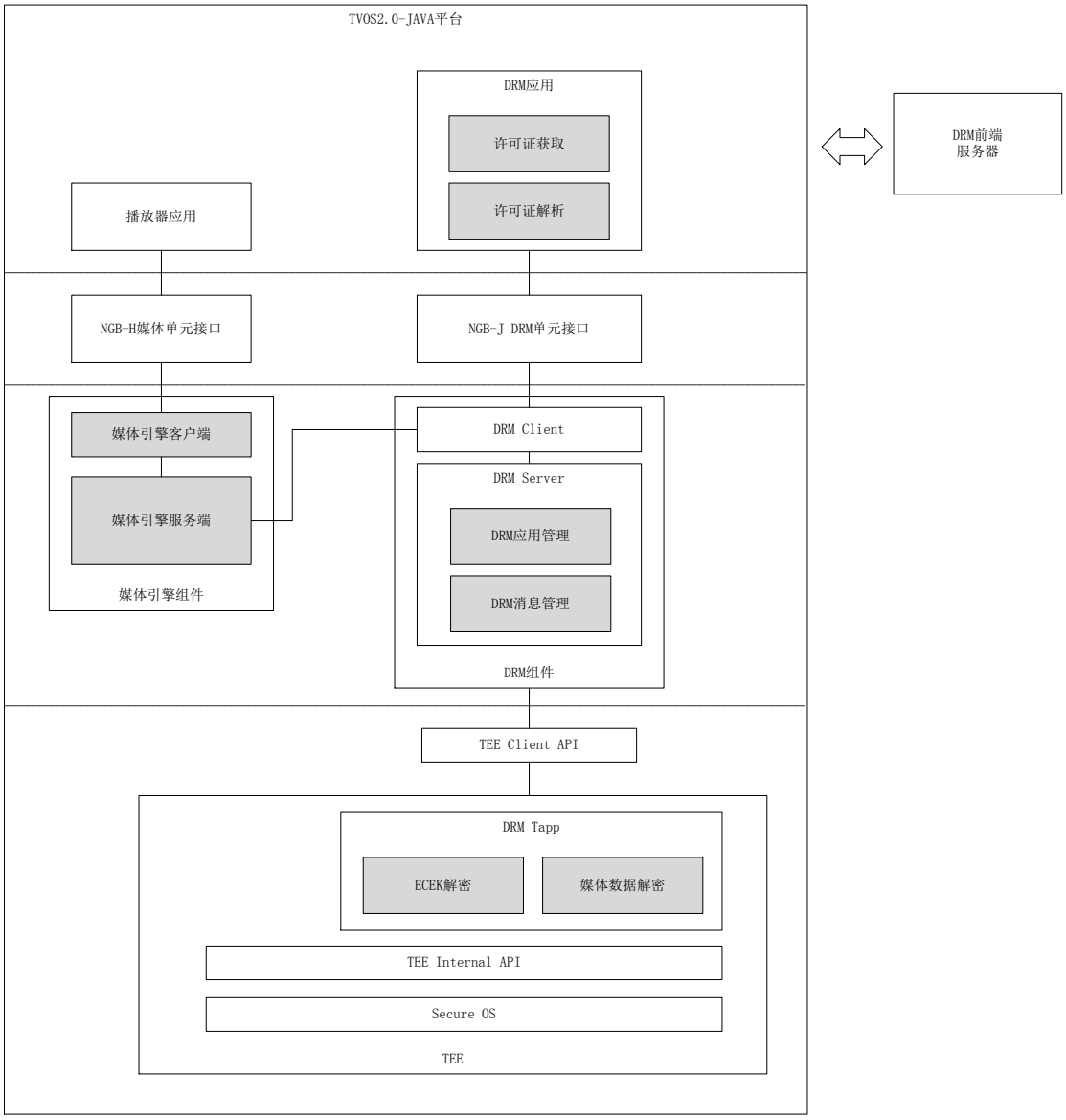


图 B.5　JAVA 平台 DRM 功能实现

具体步骤如下：

a) 播放器应用播放 OTT 节目时，通过 NGB-J 的媒体单元接口，将待播放节目 URL 传给媒体引擎组件；

b) 媒体引擎组件获取待播放媒体数据的播放列表文件和加密媒体数据，其中播放列表文件包括播放地址信息和 DRM 信息，DRM 信息中包含 DRM 应用标识；

c) 媒体引擎组件通过 DRM Client 将 DRM 应用标识发送给 DRM 组件；

d) DRM 组件查询在操作系统中注册的 DRM 应用，获取与 DRM 应用标识对应的 DRM 应用进行加载，并从 DRM 应用获取与之匹配的 DRM TApp 标识；

e) DRM 应用同 DRM 前端服务器交互，获取 DRM 内容许可证并进行解析，判断内容权限并获取加密的内容密钥 ECEK，将授权状态通过 DRM 组件返回媒体引擎组件；

f) 媒体引擎组件将加密的媒体数据存储到 TEE 与 REE 的共享缓存区，并将缓存区地址通过 DRM 组件发送给 DRM 应用；

g) DRM 应用将加密内容内存地址、ECEK 等解密相关信息通过 DRM 组件发送给 DRM TApp；

h) DRM TApp 基于预置的内容密钥解密机制，调用底层 TEE Internal API，解密出内容密钥 ECK，并

使用 CEK 解密共享缓存区中的加密媒体数据；

i) DRM TApp 将解密后的媒体数据存储在安全存储区，将安全存储区的地址通过 DRM 组件发送给媒体引擎组件；

j) 媒体引擎组件将安全存储区地址发送给底层硬件音视频解码器，对解密后的媒体数据进行解码，并通过 HDCP 保护进行输出。

B.4.2 WEB平台实现机制

WEB 平台中，TVOS 系统通过 DRM WEB 应用、DRM JS API、DRM 组件、媒体处理组件、DRM TApp 的协同工作，实现 DRM 功能，如图 B.6 所示。

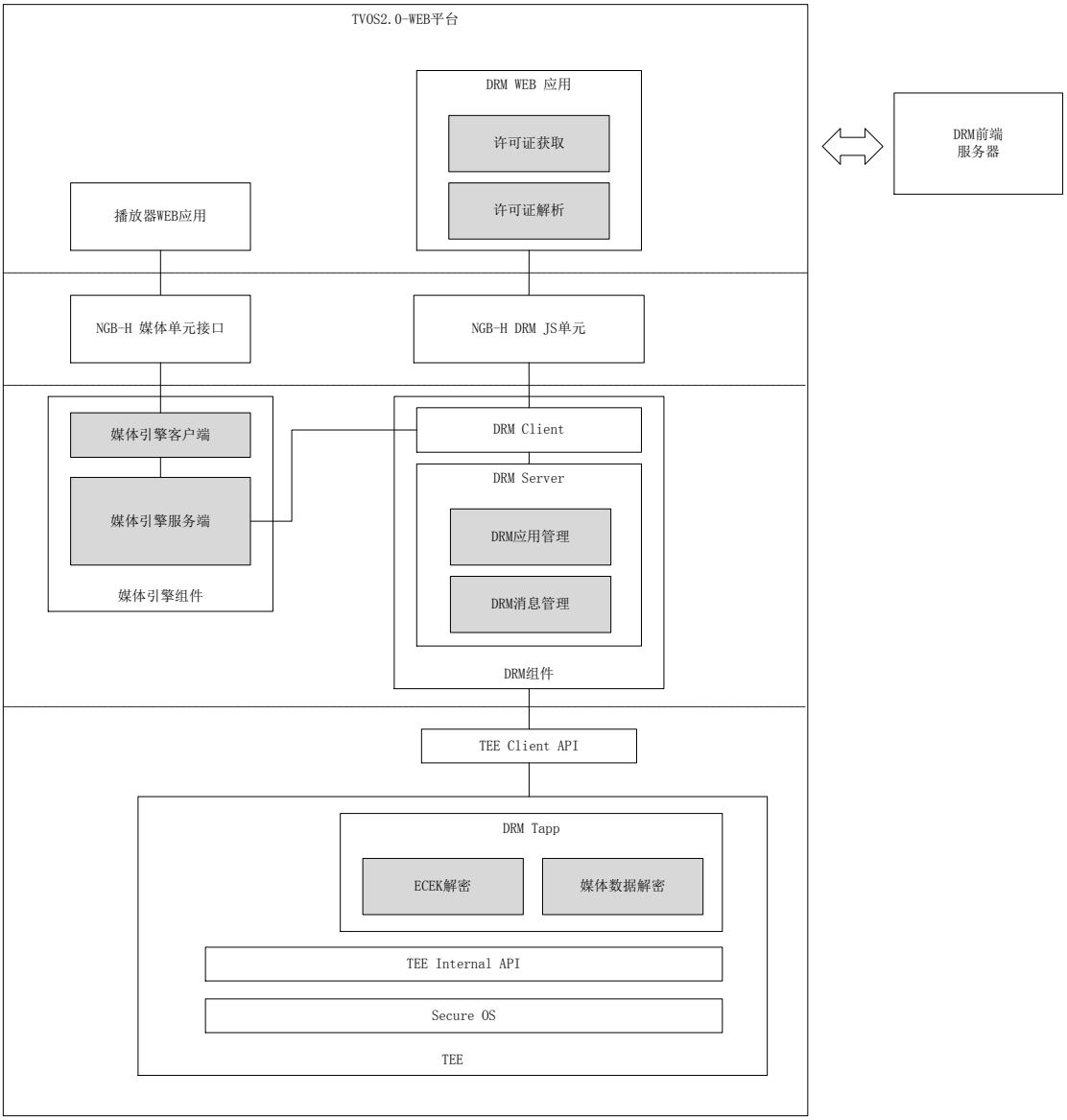


图 B.6 WEB 平台 DRM 功能实现

具体步骤如下：

a) 播放器应用播放 OTT 节目时，通过 NGB-H 的媒体单元接口，将待播放节目 URL 传给媒体引擎组件；

b) 媒体引擎组件获取待播放媒体数据的播放列表文件和加密媒体数据，其中播放列表文件包括播放地址信息和 DRM 信息，DRM 信息中包含 DRM 应用标识；

c) 媒体引擎组件通过 DRM Client 将 DRM 应用标识发送给 DRM 组件；
d) DRM 组件查询在操作系统中注册的 DRM 应用，获取与 DRM 应用标识对应的 DRM 应用进行加载，并从 DRM 应用获取与之匹配的 DRM TApp 标识；
e) DRM 应用同 DRM 前端服务器交互，获取 DRM 内容许可证并进行解析，判断内容权限并获取加密的内容密钥 ECEK，将授权状态通过 DRM 组件返回媒体引擎组件；
f) 媒体引擎组件将加密的媒体数据存储到 TEE 与 REE 的共享缓存区，并将缓存区地址通过 DRM 组件发送给 DRM 应用；
g) DRM 应用将加密内容内存地址、ECEK 等解密相关信息通过 DRM 组件发送给 DRM TApp；
h) DRM TApp 基于预置的内容密钥解密机制，调用底层 TEE Internal API，解密出内容密钥 ECK，并使用 CEK 解密共享缓存区中的加密媒体数据；
i) DRM TApp 将解密后的媒体数据存储在安全存储区，将安全存储区的地址通过 DRM 组件发送给媒体引擎组件；
j) 媒体引擎组件将安全存储区地址发送给底层硬件音视频解码器，对解密后的媒体数据进行解码，并通过 HDCP 保护进行输出。

B.5 安全支付功能实现

TVOS 软件安全提供安全支付功能，通过协调支付应用、应用框架层支付 SDK、支付组件和 Payment TApp 的协同工作，实现支付功能，如图 B.7 所示。

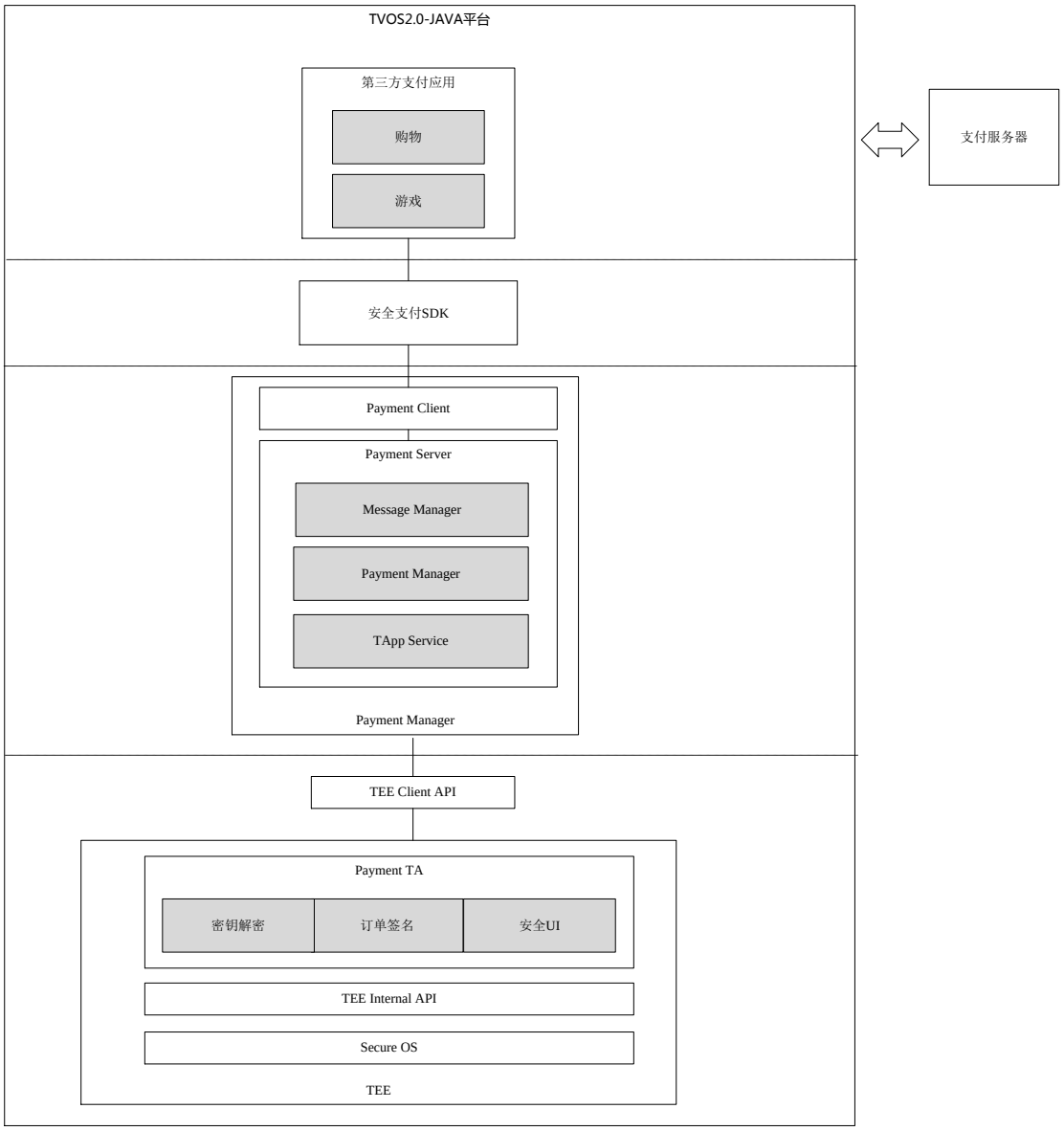


图 B.7　JAVA 平台支付功能实现

安全支付实现方法如下：

a) 用户通过第三方商户应用购买虚拟商品并发起支付请求（点击支付按钮）；
b) 第三方商户应用通过调用支付客户端程序 API 产生一个唯一的支付订单号；
c) 第三方商户应用利用步骤 b）获得的唯一支付订单号向支付后台服发起支付请求；
d) 支付后台服务调用 Payment Manager 组件对唯一订单号进行签名生成操作；
e) 支付 App(TEE)中通过解密获得用于生成签名的安全密钥；
f) 支付 App(TEE)利用步骤 e）获取的安全密钥，对请求订单号生成签名并返回；
g) 支付后台服务利用订单号和步骤 f）获得的签名向 TV 服务器（设备金额支付宝之间，管理 TV 的商户，手机）发起支付二维码请求；
h) TV 服务器进行身份认证后，向支付服务器发起支付二维码请求；
i) 用户通过手机支付宝扫一扫步骤 h）中支付宝服务器返回的二维码，向支付服务器发起支付请求；
j) 支付服务器完成支付流程并返回结果给用户。

B.6 媒体网关功能实现

TVOS 软件提供媒体网关功能，实现数字电视节目在媒体网关上的安全转发，分成服务端及客户端两部分，如图 B.8 所示。

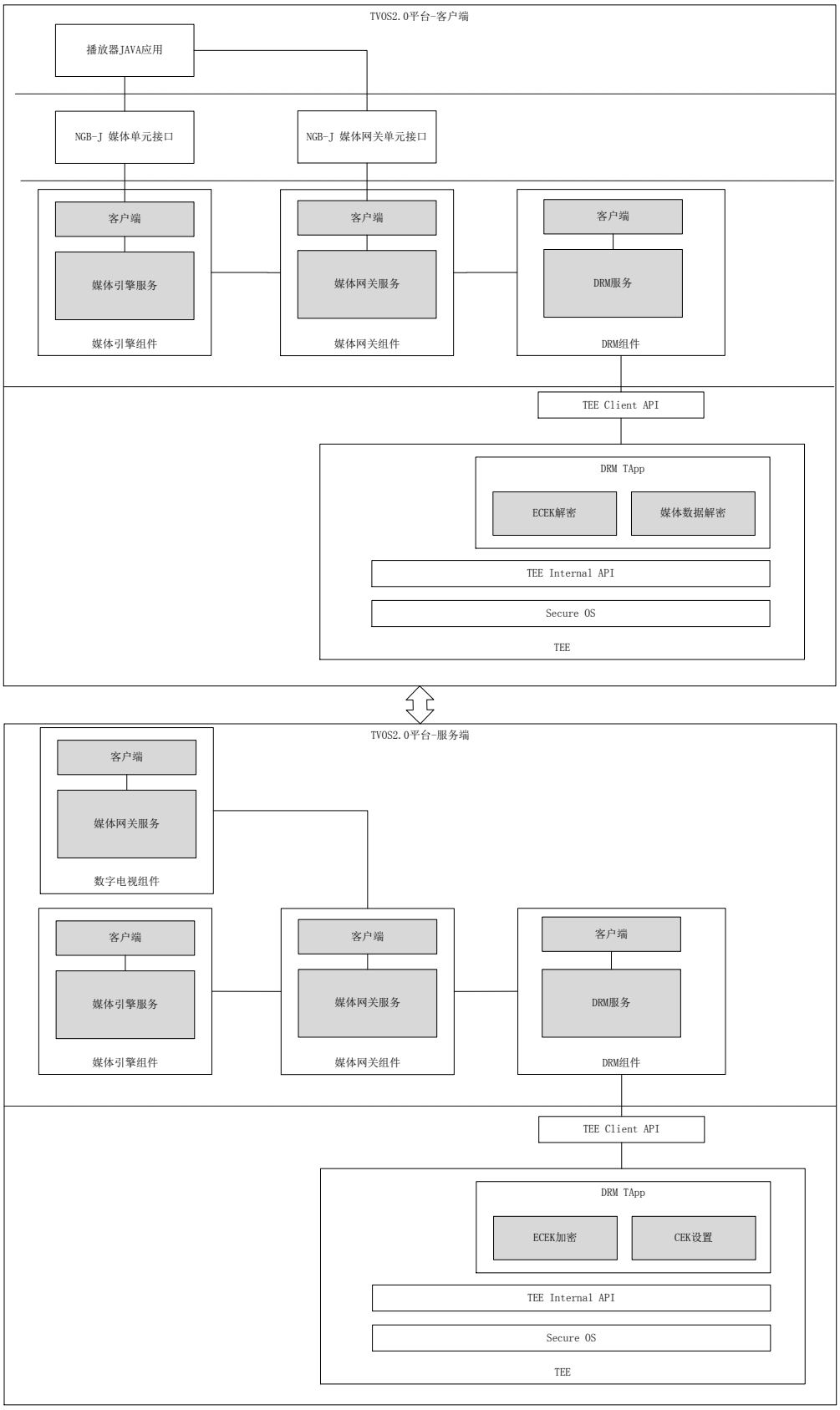


图 B.8　媒体网关功能实现

媒体网关实现方法如下：

a) 确定工作模式。媒体网关客户端上的播放器 JAVA 应用设置媒体网关组件的参数，连接媒体网关服务端，确定二者的工作模式。

b) 认证双方合法性。媒体网关客户端上的媒体网关服务组件和服务端上的媒体网关服务组件调用网关 DRM 接口，交换证书，完成对方证书的校验及合法性认证。双方的媒体网关服务组件访问 DRM 组件，获取证书。媒体网关服务组件经 DRM 接口将证书交换给对方。媒体网关服务组件获取到对方证书后调用 DRM 组件，继而通过 TEE Client API 调用 TEE 中的 TA 完成证书校验。证书校验成功表明媒体网关客户端和服务端均合法，可以开始网关节目播放。

c) 获取节目列表。媒体网关客户端上的播放器 JAVA 应用调用媒体网关组件获取节目列表，媒体网关客户端上的媒体网关组件将获取节目列表请求以 HTTP 协议转发到服务端的媒体网关组件；服务端的媒体网关组件接收到请求后，调用数字电视组件获取节目列表，返回媒体网关客户端上的媒体网关组件；媒体网关客户端上的媒体网关组件将节目列表返回给媒体网关客户端上的播放器 JAVA 应用，在播放器界面上显示。

d) 创建网关播放模块。媒体网关客户端上的播放器 JAVA 应用调用媒体引擎组件创建 OTT Player，设置节目源为本地网关提供的 HTTP 码流服务 URL 地址。

e) 获取节目码流。媒体网关客户端上的媒体引擎组件中的 OTT Player 访问 HTTP 码流服务 URL 地址，向媒体网关客户端上的媒体网关服务组件请求节目码流；媒体网关客户端上的媒体网关服务组件将请求转发到媒体网关服务端上的媒体网关组件；媒体网关服务端上的媒体网关组件解析请求中的节目信息，调用媒体引擎组件创建媒体网关播放器；媒体网关播放器，调用数字电视组件完成锁频，调用硬件 Demux 完成节目码流过滤，调用 DCAS 组件完成节目解扰，创建 TS Sink GStreamer 元件实现码流获取及组播发送；媒体网关服务端上的媒体网关服务组件从本地网卡使用组播读取码流，并将码流返回客户端的媒体引擎组件中的 OTT Player。

f) DRM 加密。节目解扰完成后，媒体网关服务端上的媒体网关组件调用 DRM 组件、继而调用 TEE Client API 接口，通过 TEE 中的 TA 产生内容密钥 CEK，并将内容密钥设置到硬件 Demux 中的加扰器（Scrambler）中，开始节目 DRM 加密；加密采用 ES 层加密，加密后重新复用成 TS。

g) 交互 DRM 内容密钥。在码流播放前，媒体网关客户端需从媒体网关服务端上获取 DRM 内容密钥。媒体网关客户端的媒体网关服务组件调用 DRM 接口 ECEK 获取功能访问媒体网关服务端的数字电视网关服务组件获取 DRM 内容密钥。媒体网关服务端的媒体网关组件调用媒体网关服务端的 DRM 组件从 TEE 中的 TA 获取 ECEK。媒体网关服务端上的 TA 使用客户端证书中的公钥加密内容密钥 CEK 生成 ECEK，依次返回 DRM 组件、媒体网关服务端的媒体网关组件，继而返回给媒体网关客户端的数字电视网关服务组件。客户端的数字电视网关服务组件调用 DRM 组件，将 ECEK 置入 DRM 组件，解密得到 CEK。

   媒体网关客户端上的媒体引擎组件中的 OTT Player 获取到码流后，经 Demux 解复用后传输到解码器中，解码器调用 DRM 解密回调接口，将加密 ES 数据发送到 OTT Player，OTT Player 调用 DRM 组件使用 CEK 解密 ES，完成节目码流的播放。

B.7 智能家居功能实现

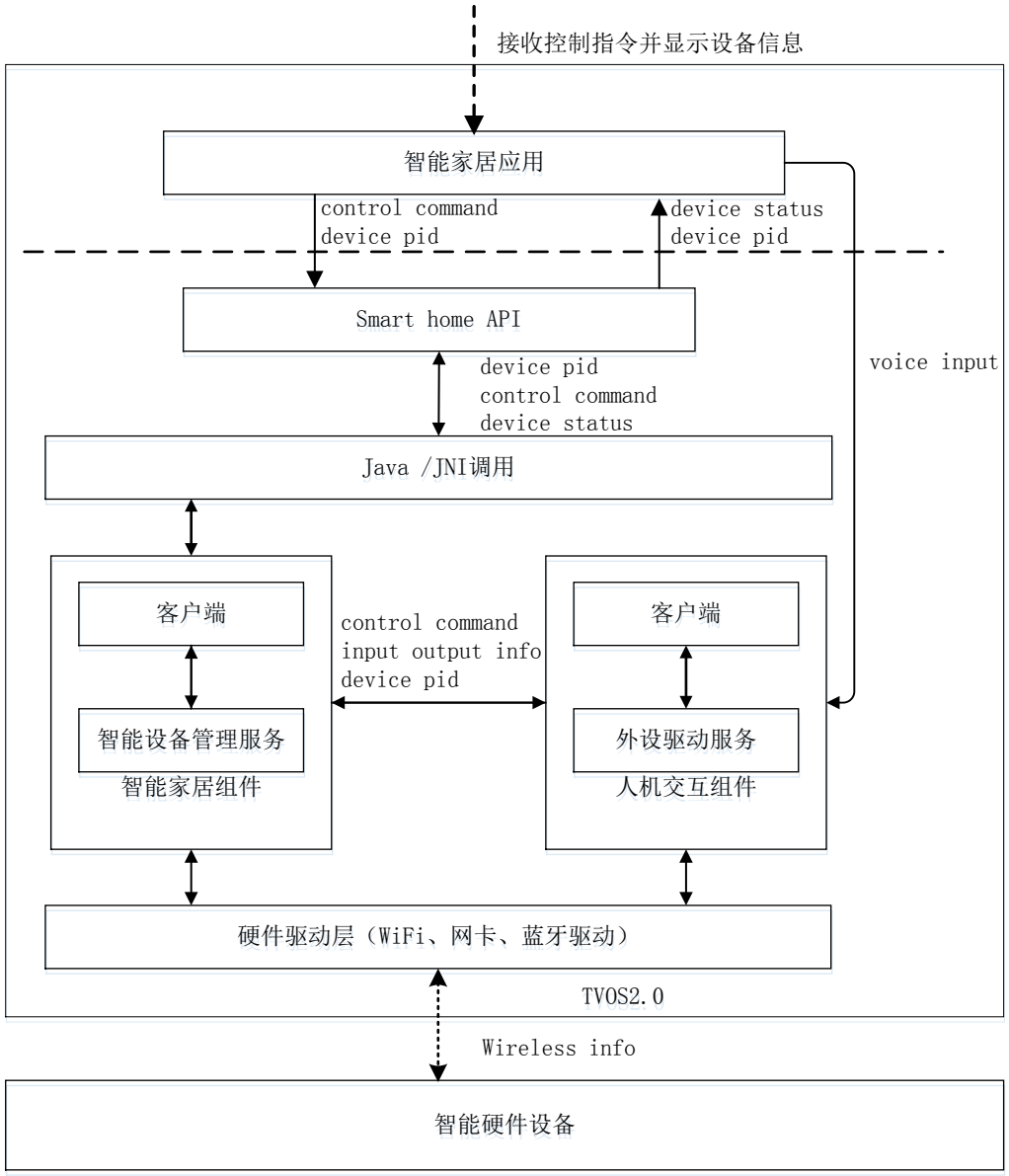


图 B.9　TVOS 系统中的智能家居组件

如图 B.9 所示，智能家居实现对设备的控制以及设备信息获取的流程，包括以下具体步骤：

a) 用户输入控制指令，智能家居应用收到指令后，分析指令，调用相应的智能家居 API 执行指令。如果是语音指令，则直接传递给人机交互组件。

b) 智能家居 API 通过 Java 或 JNI 接口调用智能家居组件。

c) 智能家居组件根据上层的指令，调用智能设备管理服务，选择要操作的对象，翻译上层的指令，并组装数据，发给硬件驱动层。

d) 如果是语音操作指令，人机交互组件分析之后，将相应的操作指令直接传递给智能家居组件。由智能家居组件选择下发到哪一个具体设备去执行操作。

e) 硬件驱动层启动相应的设备驱动服务，连接指定的设备，将接收到的来自智能家居组件的数据，按照标准协议发送给相应的智能设备。

f) 智能设备接收到 TVOS 发送的数据，按照标准进行解析，并执行命令。执行结束之后，发送结果

反馈到 TVOS 的设备驱动层。

g) 设备驱动层接收来自智能设备的数据，传递给智能家居的设备管理服务，设备管理服务根据反馈内容，重新组装数据，并确定将数据上报给智能家居 API，并做相应的设备管理操作，保存设备信息。

h) 智能家居 API 传递来自智能家居的数据信息给上层应用，完成整个工作流程。

B.8 多屏互动功能实现

在 TVOS 设备上，多屏互动组件提供了用于跨屏播放的图片媒体播放器，音乐播放器和视频媒体播放器。基于 UPNP 底层协议，对跨屏设备屏蔽了协议处理过程，流程如图 B.10 所示。

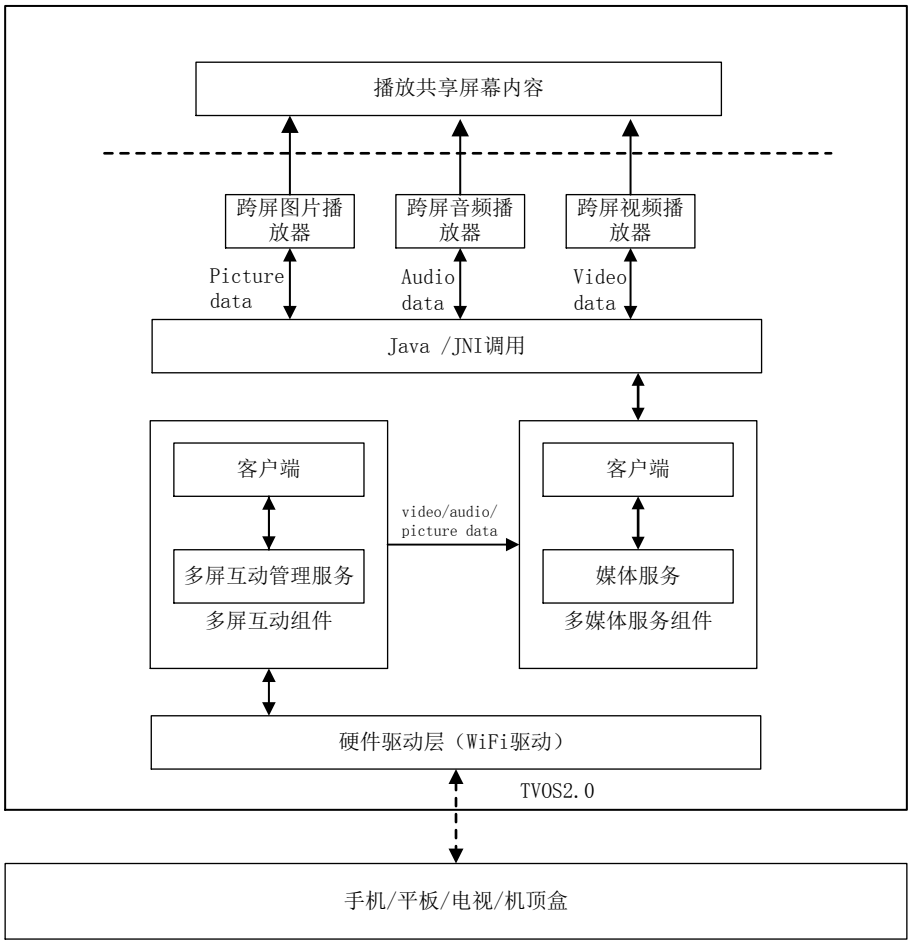


图 B.10 多屏互动实现流程

多屏互动流程的具体步骤如下：

a) TVOS 设备获取自身 IP 地址，首先向 DHCP 服务器发送 DHCP Discover 的消息，如果在指定的时间内，设备没有收到 DHCP Offer 回应消息，设备必须使用 AUTO-IP 完成 IP 地址的获取，当然也可以使用静态配置的 IP 地址；

b) 用户使用其他支持多屏互动的设备，包括手机，平板，电视等，连接到支持 TVOS 系统设备的局域网内，寻找网络上感兴趣的设备，交换一些特定信息或者某项服务的信息，发送多屏互动请求；

c) TVOS 设备发现其他设备后，获取设备或者服务的 UPnP 类型，设备的 UUID 和设备描述的 URL 地址，并得到设备描述的 URL，通过 URL 取回设备描述的信息；

d) TVOS 系统驱动层，接收来自其他设备的 WIFI 数据，通知多屏互动组件有数据过来；

e) 多屏互动管理服务接收来自其他屏幕和服务描述之后，向这些服务发出动作，同时也可以轮询服务的当前状态，将动作发送到设备服务，在动作完成或者失败后，服务返回相应的结果或者错误信息；
f) 发现设备和取得设备描述之后，识别出其他屏幕要共享的内容是图片、音频还是视频，然后调用媒体播放组件，媒体播放组件接收来自多屏互动的多媒体数据，并启动相应的多媒体播放器；
g) 播放器将需要播放的数据，解析并播放多媒体内容。

B.9 终端管控功能实现

B.9.1 终端管控整体流程

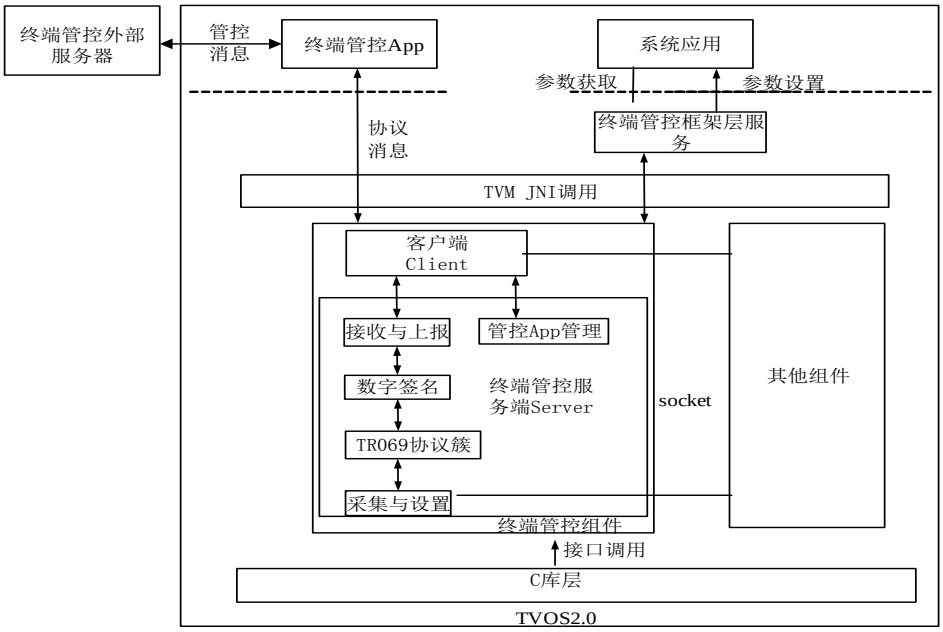


图 B.11 终端管控整体流程

如图 B.11 所示，终端管控的整体流程主要如下：

a) 外部终端管控服务器查看或者设置终端的参数，通过基于 TR069 协议的 http 报文到达终端管控 App；
b) 终端管控 App 解析 http 报文，将 TR069 格式 xml 数据传递给组件接收与上报模块；
c) 接收与上报模块将消息传送给数字签名子模块进行校验处理；
d) 数字签名子模块校验成功则将消息传递给 TR069 协议簇模块；
e) TR069 协议簇模块将消息解析并传递给采集与设值模块；
f) 如果是处理系统底层参数，那么就调用 C 库的接口执行操作。如果是其他组件的参数，那就调用其他组件提供的 client 接口；
g) 其他组件通过调用终端管控组件 client 接口完成相关信息上报；
h) 采集与设置模块将参数数据传送给 TR069 协议簇模块进行处理，然后由数字签名模块进行签名，并由接收与上报模块将消息发送给终端管控 App，终端管控 App 再发送非外部终端管控服务器。

B.9.2 终端管控实现机制

B.9.2.1 终端管理服务器

终端管理服务器主要是用来管理终端的服务器，具有和所有的终端进行安全管控通信的功能，能够提供管理员操作和查看各个终端，实现终端管控功能等服务。

服务器和终端管控通信使用机顶盒的网口通过 http/https 报文承载 TR069 协议进行交互通信。

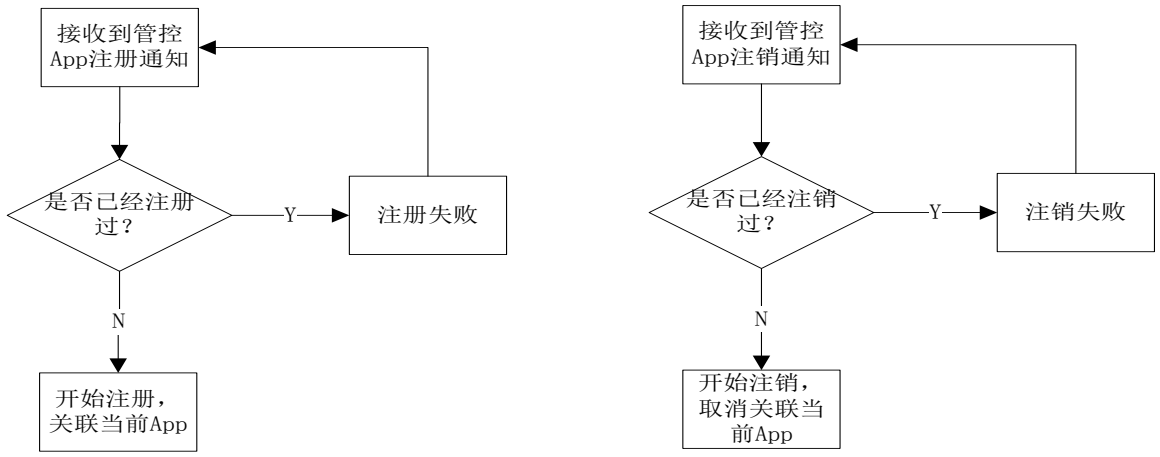
服务器不属于 TVOS 的范畴，在这里不作详细描述。

B.9.2.2 终端管控App

应用层App，主要负责向端管控组件注册和注销；向终端管理服务器执行终端注册、定时心跳消息的触发；定时采集终端参数的触发；处理终端管理服务器的反向连接；处理和终端管理服务器的消息鉴权；中继管理服务器和组件的交互消息；负责将TR069报文组长成http报文和将http报文解析出TR069报文。这里不作详细描述。

B.9.2.3 管控App管理模块

负责对管控App实施注册与注销机制，机制流程如图B.12所示。



a)注册机制　　b)注销机制

图B.12 管控App管理模块流程

B.9.2.4 接收与上报模块

负责与终端管控App进行信息交互，接收来自App的消息，传递给数字签名模块；转发来自数字签名模块的消息，转发给App。

B.9.2.5 数字签名模块

负责实现对终端管控App发来的信息进行数字签名校验，负责实现对上报信息的数字签名。流程如图B.13所示。

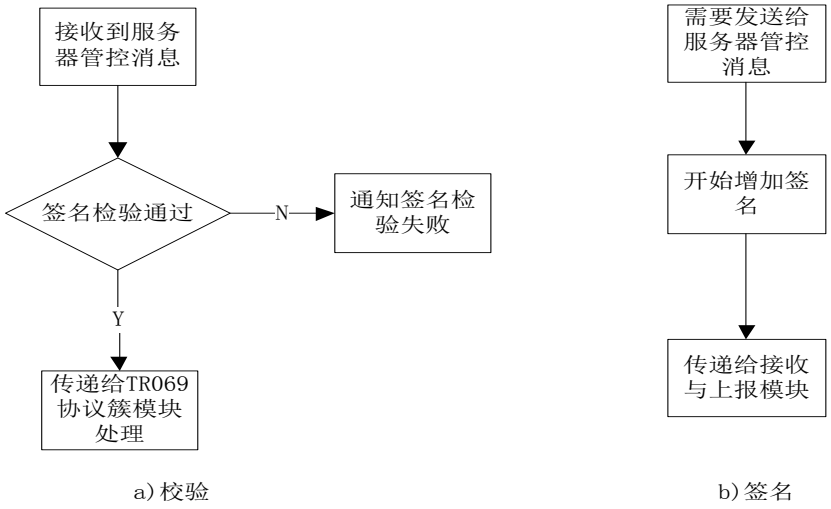


图 B.13　数字签名流程

B.9.2.6　TR069协议簇模块

负责实现TR069协议族协议簇功能，能完成接收数据的协议分析以及上报数据的协议封装。

B.9.2.7　采集与设置模块

负责与其他软件模块协同完成对终端状态参数的查询和设置。

B.10　数据采集功能实现

B.10.1　数据采集总体架构

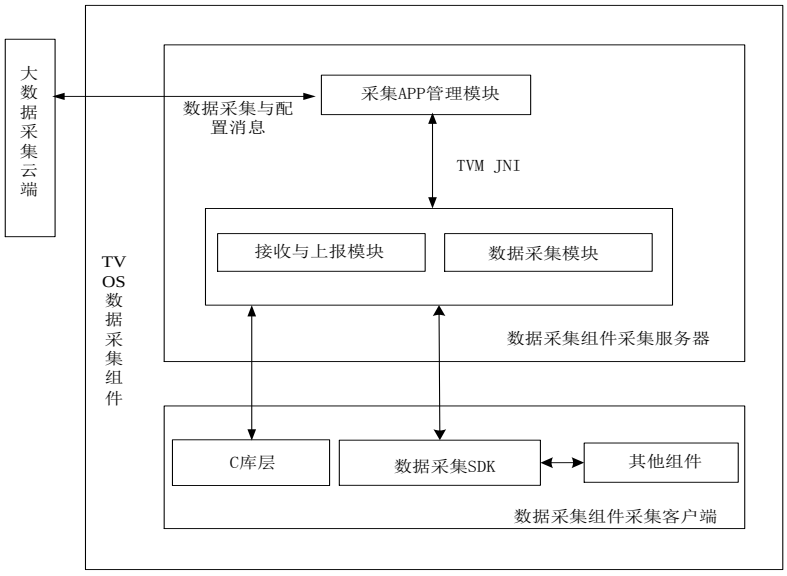


图 B.14　数据采集总体流程

如图 B.14 所示，其中接收与上报模块负责与数据采集 App 进行信息交互。
数据采集模块负责与其他软件模块协同完成对终端业务的数据采集。

采集App管控模块负责对数据采集App实施注册与注销机制。

B.10.2 数据采集实现机制

B.10.2.1 数据平台云端服务器

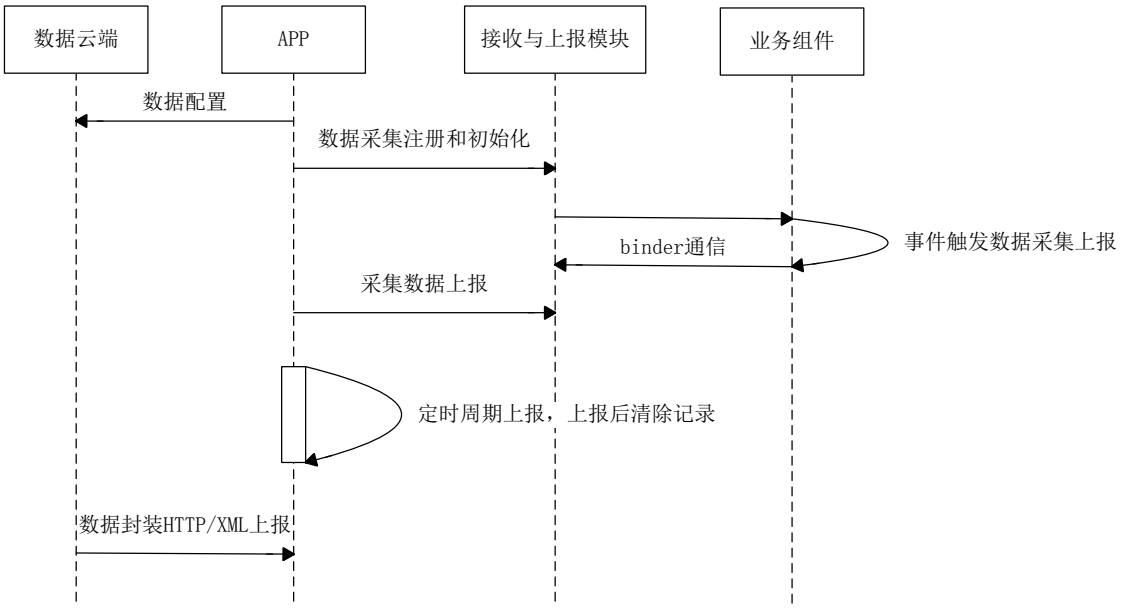
数据平台云端服务器主要是用来汇总和分析终端的数据，能够和所有终端通信，等待终端上报终端采集的数据并进行数据分析。服务器不属于TVOS的范畴，在这里不作详细描述。

B.10.2.2 App管控模块

采集App管控模块负责对数据采集App实施注册与注销机制。负责查询和周期性定时与数据云端数据上报。

B.10.2.3 数据主动上报

系统数据采集主动上报流程如图B.15所示。



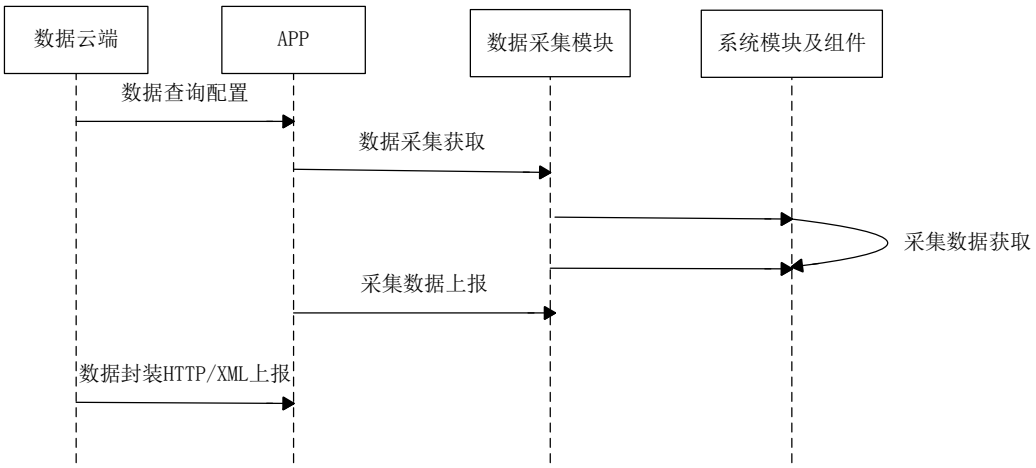
图B.15 系统数据采集流程

数据主动上报流程如下：

a) 数据云端下发数据上报配置信息，App监听后进行数据采集注册和初始化；
b) 业务组件随后由事件触发数据采集；
c) 数据接收与上报模块将数据上报App；
d) App将数据存入内存链表，周期定时上报，上报后清除记录；
e) 上报数据封装格式HTTP/XML。

B.10.2.4 数据采集查询流程

数据采集查询流程如图B.16所示。



图B.16　数据采集查询流程

数据采集查询流程如下：

a)　数据云端下发数据查询配置信息，配置要查询的信息；

b)　App 下发数据查询消息获取数据；

c)　数据采集模块从其他模块或组件获取采集数据；

d)　App 将获取数据封装格式 HTTP/XML 上报。

B.11　应用管理功能实现

B.11.1　应用管理模型的构建与实施

TVOS 应用承载其活动（Activity）及服务（Service）的上下文描述，而应用管理组件为应用的活动（Activity）提供一个应用的最基础的页面，通过 Activity 可让系统准确的控制应用的生命周期；而服务（Service）则提供一个程序执行的基础，不同于 Activity，Service 并不具备显示画面，但也同样具有生命周期。透过 Service 得以实现耗资源的运算或者背景应用等不须画面的工作；而透过 ActivityManager 则可以提供程序执行相关的系统信息。

TVOS 应用管理组件定义了 TVOS 的应用模型，TVOS 的应用框架基于此应用模型构建和实施。构成 TVOS 应用的几个主要部件包括了活动（Activity）、服务（Service）、消息机制（Broadcast & Intent）、内容提供器（Content Provider）等，这些部件组成了 TVOS 应用的概念空间和特性空间。

ActivitiyThread 是应用程序概念空间中的重要概念，是应用模型和应用框架实现的基础。基于 ActivityThread 提供的 IActivityThread 接口，应用管理组件的服务端可以将 Activity 状态的变化传递到客户端，从而完成应用的生命周期等管理。

TVOS 应用模型与应用框架如图 B.17 所示。

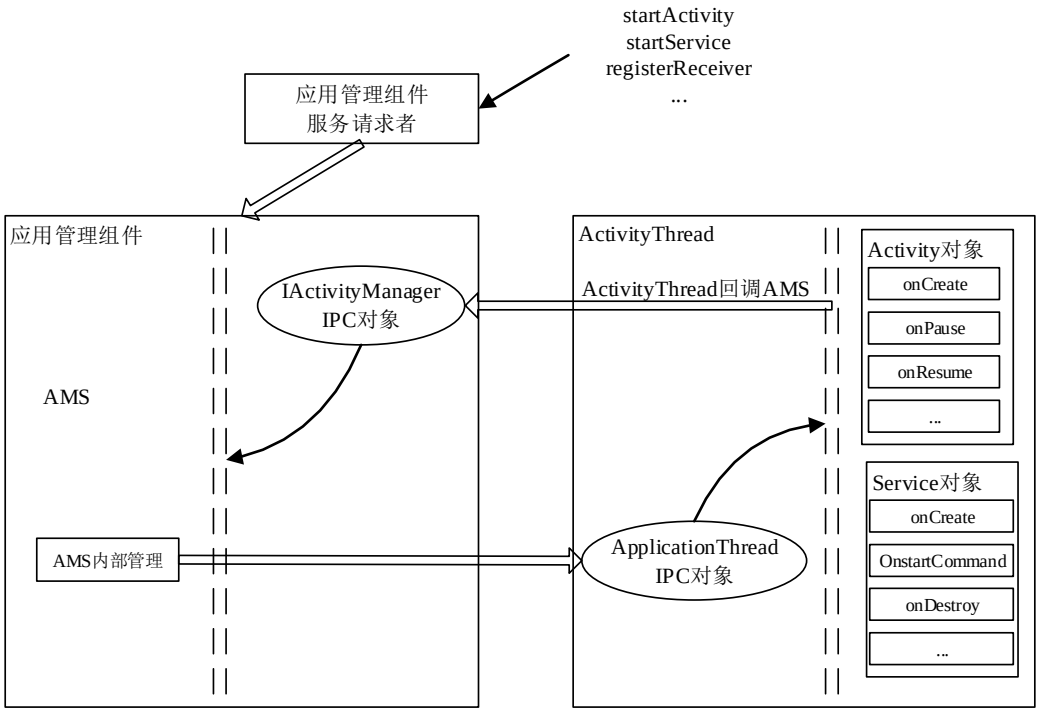


图 B.17　TVOS 应用模型与应用框架

B.11.2　应用管理机制

TVOS 应用程序通常由一个或多个基本部件组成，分别为：Activity、Service、Content Provider 以及 Broadcast Receiver。应用管理组件是这四大组件的控制中心，负责它们的开启，关闭以及查询管理操作。应用管理组件寄存于系统服务进程中，当系统启动时，将创建一个线程来循环处理客户的请求。

a)　Activity 的调度机制

Activity 是提供应用程序与用户进行交互的组件。每个 Activity 都提供了一个可用于显示用户界面的窗口。TVOS 应用程序通常由多个 Activity 组成。每个 Activity 都可以启动其他 Activity 来实现用户的操作，当启动了一个新的 Activity 后，旧的 Activity 将会停止，同时新的 Activity 会被系统压到一个称为回退栈（Back Stack）的栈中，并获取用户的焦点。这个 Back Stack 遵循“后进先出”的原则，因此当用户操作完当前 Activity 并按下“返回”键后，当前 Activity 将会被从栈中弹出并销毁，之前的 Activity 将重新恢复。

当一个 Activity 由于有新的 Activity 启动而被停止时，它将会通过 Activity 生命周期中的回调方法获取到这个状态改变消息。一个 Activity 可能会因为状态的改变而收到多个回调方法，无论是系统创建 Activity、停止 Activity、恢复 Activity、还是销毁 Activity，都会接收到一个回调方法。利用这些回调方法，可以在 Activity 状态改变时进行相应的操作；当 Activity 恢复时，可以重新获取资源以及恢复被中断的操作等。这些状态之间的转换也就形成了 Activity 的生命周期。

b)　Broadcast 通信机制

消息广播管理模块使用了 Broadcast 通信机制，实现对软件模块间的广播消息通信管理，可以降低消息发送方和接受方的耦合度，提高系统的可扩展性和可维护性。该机制基于消息发布和订阅的事件模型，即广播发送者负责发布消息，广播接收者需要先订阅消息，才能接收到消息。

应用管理组件是广播机制中的注册中心，广播接收者将自己注册到应用管理组件中，并指定接收的广播类型。当广播发送者发送一个广播时，这个广播先发送到应用管理组件，然后应用管理组件根据这个广

播的类型找到相应的广播接收者，最后将这个广播发送给他们处理。

广播接受者的注册方式可分为静态注册和动态注册，同等情况下，动态注册的广播接收者比静态注册的广播接收者优先收到广播。

广播的发送方式可分为有序和无序两种。广播接收者注册广播时，可以指定他们的优先级。当应用管理组件接收到有序广播时，它将把这个有序广播发送给符合条件的、优先级较高的广播接收者处理，然后发送给符合条件的、优先级较低的广播接收者处理；而当应用管理组件接收到无序广播时，他就会忽略接收者的优先级，并行地把该无序广播发送给所有符合条件的广播接收者处理。

广播的发送和接受都可以通过权限限制的方式，保证通信的安全，可以根据需要进行配置。

c) Service 机制

Service 机制是 TVOS 系统中用于处理与用户界面无关的业务逻辑的运行机制。

Service 不直接与用户交互，所涉及的业务逻辑一般都是计算型的，在后台运行。例如音乐播放器服务，当用户退出媒体应用用户界面，但希望音乐依然可以继续播放，此时 Service 将保证音乐继续播放。

Service 拥有一部分 Activity 所不具备的特性。一是后台运行，开机自动启动，在后台运行不会有过多对话框来影响用户体验。二是不被 Activity 生命周期所限制，Activity 处于完全活跃的周期是 onResume()与 onPause()之间，如果这周期之外发生了事件，Activity 便无法响应处理。由于 Service 同样也是主线程里执行的，于是也会因为生命周期回调过程中执行耗时操作而触发应用程序无响应，因此应将耗时操作尽可能放在独立线程中执行。

Service 是构建 TVOS 服务的根本，TVOS 组件层本身就是由大量 Service 构成，对 Service 的管理是由应用调度管理模块和应用生命周期管理模块来完成。

d) 软件模块间的数据共享机制

应用管理组件负责对管理软件模块间的数据共享进行管理，软件模块间具体的数据共享通信是通过数据交换管理组件提供的数据更新通知机制来完成的。

e) 进程管理机制

默认情况下，一个应用程序运行在同一个进程中。如果有特殊需求，也可以通过设置资源配置文件来改变默认行为，将 Activity、Content Provider、Service 以及 Broadcast Receiver 生成具有唯一标识的独立进程，不同的应用程序可以通过进程标识共享同一个上述进程。系统可以在内存缺少的情况下，强制停止一些对用户当前操作影响不大的进程。被停止的进程当再次收到组件调用请求的时候会被应用管理组件重新启动。

B.11.3 TVOS应用程序的启动过程和原理

Intent 是 TVOS 中对一个操作的描述，具有多种用途，既可用于启动应用程序的 Activity 实例，也可用于发送一个特定的广播等。由于应用程序需要跨进程调用系统服务（启动其他进程中的组件）的功能，因此 Intent 在实现内部被设计成一个可以被序列化，并用于进程间通信时传递各种数据的载体。

当需要启动一个应用程序的 Activity 时，发起者首先需要构造一个 Intent 请求，其中包含要启动的 Activity 名称。调用者无需使用被调用者的头文件定义，只需要知道名称，便可以通过这种松耦合的方式发出请求。当应用管理组件收到请求后，应用管理组件开始通过用户 ID 确保调用者有权执行这一请求操作。

接下来通过应用安装组件查询 Intent 的相关信息，确定要启动的 Activity 存在于哪个已安装的程序包中，以及其他相关信息。有了足够的信息后，应用管理组件开始生成 ActivityRecord 实例。ActivityRecord 需要和特定的 Task 相关联，应用管理组件根据启动模式等参数判断是否需要启动新的 Task，如果不需要则寻找出需要关联的现有 Task。此时 ActivityRecord 被置于 Activity 栈顶端。

在系统默认的情况下，隶属于同一个应用程序包的应用程序组件，在运行时共享同一个进程。因此应用管理组件首先判断需要启动的 Activity 对应的进程是否已经存在。如果进程不存在，应用管理组件需

要通过控制孵化器产生的种子进程，创建用于执行此 Activity 的应用程序进程。

B.12 窗口管理功能实现

B.12.1 窗口的创建和组织管理方式

应用程序启动后，窗口管理服务首先需要计算其窗口位置大小，并为其创建窗口对象。为了能正确地显示窗口内容，窗口管理服务需要协调新添加的窗口与其他窗口之间的关系，实现窗口的组织和管理。对窗口的组织管理主要包含两方面的内容，一是系统中多个应用程序同时运行时，各个应用程序窗口之间的组织和管理；二是应用程序窗口与系统中的特殊窗口（壁纸、输入法等）之间的组织和管理。

a) 应用程序窗口创建

窗口管理服务作为服务端接收来自应用程序的请求，应用程序发起创建窗口指令后，窗口管理服务在服务端创建与客户端对应的窗口对象，称之为服务端窗口对象，它主要负责完成与客户端窗口对象的交互通信。与此同时，窗口管理服务将为服务端窗口对象创建窗口 surface，并使其共享于客户端窗口对象。而窗口显示的本质是将窗口内容绘制于其拥有的窗口 surface 上，然后将窗口 surface 的内容发送至 surface 合成渲染进程进行合成渲染，最终展现在显示设备上。

由于 TVOS 采用共享内存（显存）的方式绘图，因此客户端窗口 surface 和服务端窗口 surface 共享相同内存（显存）空间，在窗口管理服务创建窗口 surface，同时向 surface 合成渲染进程请求共享内存（显存）空间，新创建的共享内存（显存）区域由窗口 surface 管理，客户端将需要绘制的窗口内容填充至窗口 surface，然后以进程间通信的方式传送至 surface 合成渲染进程进行合成渲染。

如果应用程序端向窗口管理服务发出改变窗口大小或者显示隐藏窗口等共有指令，则服务端窗口对象通过改变其窗口 surface 的大小或显示隐藏后，通知应用程序端重绘，而此时应用程序端将再次触发 suface 合成渲染进程重新将窗口 surface 的内容渲染至显示设备显示。

除上述情况之外，服务端窗口 surface 还用于绘制窗口自身动画效果和窗口之间切换动画效果，窗口创建示意图如图 B.18 所示。

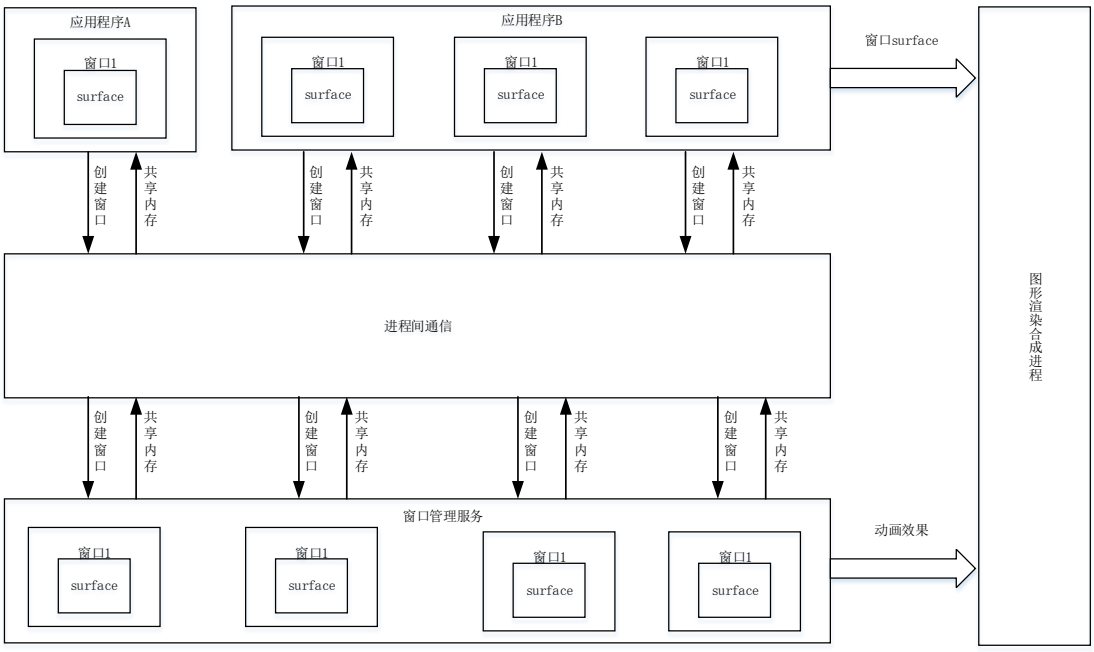


图 B.18 TVOS 窗口创建简要示意图

b) 应用程序窗口管理机制

应用程序窗口管理机制主要体现在对显示设备显示区域的划分和窗口层叠次序的计算。显示设备显示区域划分好后，就可以确定应用程序显示区域大小，窗口层叠次序计算完成后就可以通知 surface 合成渲染进程按正确的叠放次序显示所有窗口。

B.12.2 显示设备划分及应用程序窗口大小计算

为保证准确显示应用程序窗口，窗口管理服务需对显示设备进行划分并计算出应用程序窗口的大小。以电视屏幕为例，情景一：应用程序非全屏显示并不弹出输入法窗口。此场景窗口管理服务需从屏幕区域裁剪状态栏区域和导航栏区域，裁剪后的区域称之为应用程序内容区域，从应用程序内容区域再裁剪四个边衬区域，{content-left, content-right, content-top, content-bottom}， 最后剩余区域是窗口内容显示区域，如图 B.19 所示。

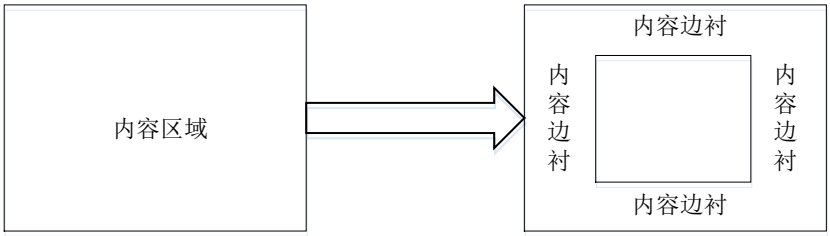


图 B.19 显示设备区域划分-1

情景二：应用程序非全屏显示并弹出输入法窗口。窗口管理服务需重新计算应用程序窗口内容显示区域，称之为应用程序可见区域，应用程序可见区域和应用程序内容区域在不弹出输入法窗口时大小一致，而在应用程序弹出输入法窗口后，应用程序内容区域被输入法窗口部分遮盖，窗口管理服务需从输入法服务取得输入法窗口的所占区域大小，并从应用程序内容区域裁剪输入法窗口区域，最后余下区域被定义为应用程序可见区域，同理从应用程序可见区域裁剪四个边衬区域，{visible-left, visible-right, visible-bottom, visible-top}，最后剩余区域是窗口内容显示区域，如图 B.20 所示。

图 B.20 显示设备区域划分-2

据上所述，两种情景都需裁剪出边衬区域，它的主要作用是当屏幕显示内容超出应用程序内容区域或应用程序可见区域范围时，需要绘制出滚动条。边衬区域数量设计成四个主要是为了适应横竖屏切换的场景，边衬区域的宽度设计成滚动条的宽度，而滚动条的宽度，状态栏和导航栏所占区域范围，均可在资源配置文件中设定默认固定值，窗口管理服务获取相应值后进行计算处理，给它们预留出对应绘图区域。

B.12.3 窗口叠放次序计算

TVOS 是多任务操作系统，系统中可以同时运行多个应用，而每个应用程序都至少拥有一个窗口，窗口管理服务负责管理所有应用程序的窗口。为了让这些窗口正确有序的显示，窗口管理服务需对它们的层叠顺序进行管理，实现此功能的关键就在于 z-order 排序，z-order 排序的本质是计算窗口 surface 叠放次

序，它的原则是顶层窗口的 z-order 值最大，底层窗口的 z-order 值最小。

计算窗口 z-order 值依赖窗口堆栈，窗口类型以及窗口是否为子窗口。z-order 计算完成后会传送到 surface 合成渲染进程，它会根据窗口 surface 的 z-order 渲染出正确的窗口叠放顺序。 窗口堆栈管理所有服务端窗口对象，在计算 z-order 之前，所有应用程序的服务端窗口对象在窗口堆栈中的位置已经被计算确定，该项工作由应用管理服务完成，不过这里的窗口对象是指基础窗口对象，可以把基础窗口对象理解为根窗口，所有子窗口都以它为父窗口或祖先窗口。

窗口类型是指 TVOS 所支持的所有窗口种类，主要有状态栏，搜索栏，系统警告窗口，系统错误窗口，系统输入法窗口，系统输入法对话框（位于输入法窗口之上），重叠窗口，位于锁屏窗口之上的壁纸窗口，短暂显示的通知窗口等。

在引入计算 z-order 算法之前，先设想触发窗口管理服务调整 z-order 的主要场景：

a) 弹出输入法窗口；
b) 弹出输入法对话框；
c) 添加普通应用程序窗口；
d) 添加普通应用程序窗口，并且该窗口需要显示壁纸；
e) 添加壁纸窗口；
f) 焦点窗口更改；
g) 窗口可见性改变。

上述场景被应用程序触发后，窗口堆栈需作调整（以下窗口均为服务端窗口对象）：

a) 将输入法置于弹出输入法窗口的窗口之上；
b) 将输入法对话框置于输入法窗口之上；
c) 如果被添加窗口是子窗口，则在窗口堆栈中查找其父窗口位置并根据子窗口类型确定置于父窗口之上或之下，否则使用应用管理服务确定的堆栈顺序；
d) 使用应用管理服务确定的应用窗口堆栈顺序，并把壁纸窗口置于该窗口之下；
e) 将壁纸窗口置于需要显示壁纸的窗口之下；
f) 重新计算焦点，将当前焦点窗口置于窗口堆栈中应用程序窗口的最顶层，而某些系统窗口仍在其上，如状态栏；
g) 将不可见窗口调整至应用程序窗口最底层。

窗口堆栈调整完成，触发窗口 z-order 计算，即计算窗口 surface 叠放顺序（以下窗口对象均指服务端窗口对象）：

a) 根据窗口类型为窗口分配固定窗口类型值 windowtype；
b) 定义乘法因子常量 multiplier；
c) 定义偏移值常量 offset；
d) 计算同一类型窗口 baselayer 值，baselayer = windowtype * multiplier + offset；
e) 确定窗口是否为子窗口以及子窗口类型；
f) 计算窗口堆栈中窗口 z-order 值 Z，Z = baselayer + n * multiplier + 窗口令牌 z-order 调整值，n 为窗口在窗口堆栈中的由底层到顶层的位置序号，窗口令牌 z-order 调整值由应用管理服务在创建 window token（关于其定义请参考安全性设计）时指定。如果该窗口为子窗口，则子窗口找到父窗口位置后再根据其类型决定位于父窗口之上或之下，简要结构如图 B.21 所示。

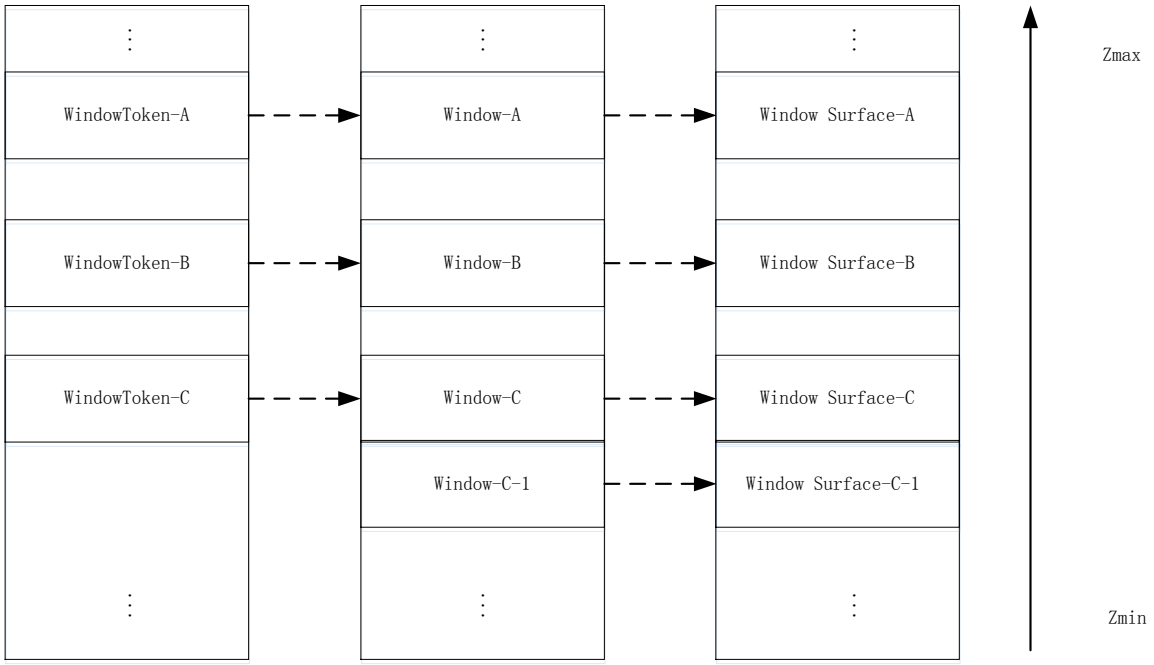


图 B.21　TVOS 应用程序窗口管理框架

B.12.4　安全性设计

窗口管理服务引入窗口安全机制。应用程序启动后应用管理服务创建 window token 用来唯一标识应用顶层窗口，应用管理服务会把 window token 传递给应用程序、窗口管理服务和输入法管理服务。窗口管理安全性主要体现在以下几方面：

a) 阻止非法程序操控正常程序绘制伪装窗口；窗口管理服务要求应用程序在添加或删除其窗口时必须提供 window token，非法程序试图向其他应用程序添加窗口时会因 window token 不匹配而被窗口管理服务拒绝操作。

b) 阻止除 launcher 以外的应用程序与壁纸窗口发生交互；窗口管理服务只允许 launcher 程序与壁纸窗口发生交互，其他非法应用程序如果试图向壁纸窗口发送指令，也会因为它所持有的 window token 不被窗口管理服务接受而失败。

c) 阻止非法程序操控其他正常程序的输入法窗口；在请求输入法管理服务对输入法做相应动作时，必须指定当前应用程序的 window token，如果操控到其他应用程序弹出的输入法窗口，则会被输入法管理服务拒绝。

B.12.5　应用程序窗口与特殊窗口之间的组织管理

特殊窗口主要包含状态栏、壁纸、输入法窗口。这些窗口的组织管理不同于普通应用程序窗口的组织管理，具体如下：

状态栏主要用来显示一些系统图标（如：wifi 状态、系统事件等）和应用的通知图标（如：下载、闹钟）。状态栏窗口的高度在系统的配置文件中定义，它的 Z-order 位于所有应用程序窗口之上。只有当最上层应用程序窗口是全屏窗口时，状态栏窗口被隐藏，其他情况下它都应显示在屏幕的最上方。

壁纸主要用来提高 TVOS 的用户体验，可以被设置为各种图片。如果当前显示应用程序被设置有透明属性和显示壁纸属性时，窗口管理服务应该将壁纸窗口紧贴当前应用程序窗口，并放置于它的下方。通过这样的方式，实现在透明或者半透明应用程序窗口中显示壁纸内容，美化应用程序。

输入法窗口主要用来显示输入所需要的各种模拟键盘，用户可以通过输入法窗口向取得焦点的应用程序窗口输入字母或者文字。窗口管理服务应该时刻关注是否有窗口需要使用输入法，如果有应用程序窗口需要使用输入法，窗口管理服务就会调整输入法窗口的位置，将其置于当前需要使用输入法的应用程序窗口之上。

当某应用程序需要同时显示壁纸、输入法、状态栏时，他们之间的位置如图 B.22 所示。

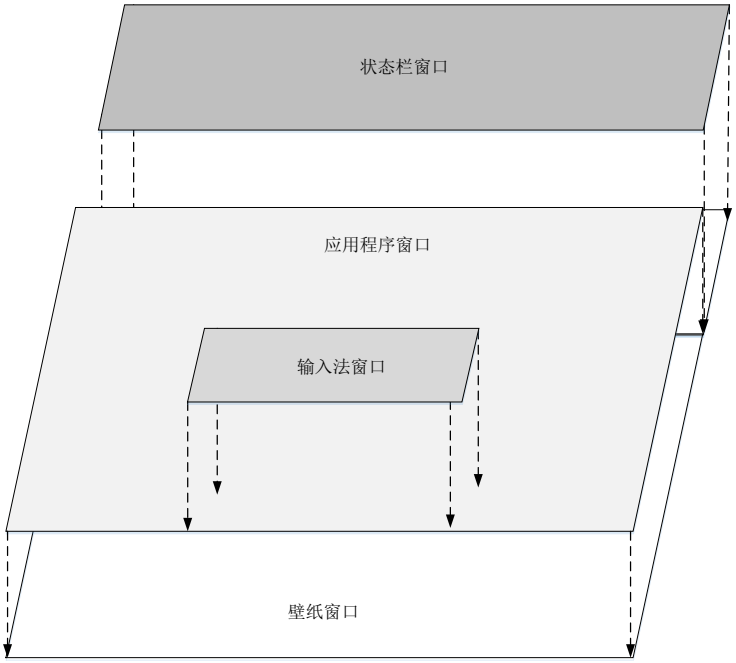


图 B.22　特殊窗口的组织管理方式

B.12.6　窗口焦点管理和输入事件转发

在 TVOS 系统中，当有输入事件产生时，系统会根据输入事件类型和信息，以及当前系统中的应用程序状态，决定将该输入事件发送到哪个 TVOS 程序，然后由对应的 TVOS 程序响应该输入事件，完成其应有的功能。其工作流程如图 B.23 所示。

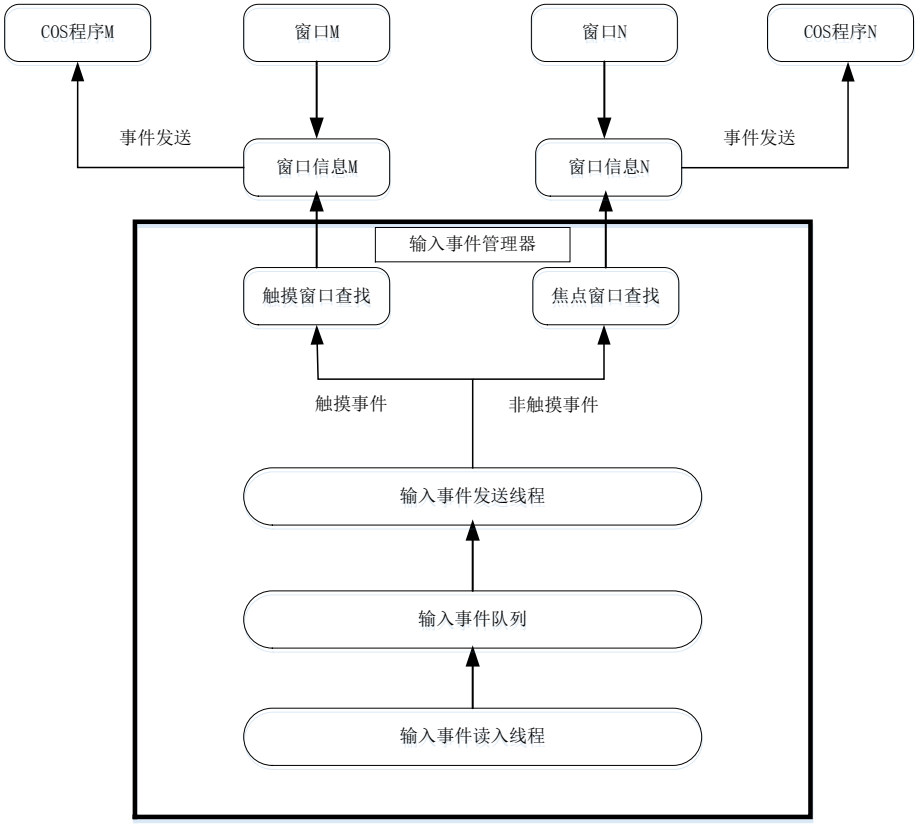


图 B.23　输入事件转发流程

如图 B.23 所示，窗口管理服务会启动和控制系统的输入事件管理器，而输入事件管理器负责输入事件的采集和发送。它主要分为两个工作线程，读取线程负责循环读取输入事件，当有输入事件产生时，由输入事件读取线程把输入事件放入到输入事件队列中，然后由输入事件发送线程将输入事件发送到应用程序中。为了保证输入事件能被准确地发送到对应的 TVOS 程序，输入事件管理器包含窗口管理服务中所有窗口的信息描述，根据这些窗口信息描述和输入事件本身的信息，决定输入事件应该被发送到哪个窗口。

而窗口管理服务负责更新和维护系统输入事件管理器中的窗口信息列表，每个窗口信息包含对应窗口的状态、大小、位置，是否为焦点窗口以及输入事件传输通道等信息。每当系统中的窗口信息发生变动，如添加删除窗口，调整窗口位置和状态等操作时，窗口管理服务会重新计算窗口信息，并将它们更新到输入事件管理器中。

当输入管理器接收到输入事件，如系统按键所产生的输入事件，首先根据指定的系统窗口管理策略决定是否将该事件发送到应用程序端，如果由系统响应该输入事件，如：导视键，电源键被按下，则将该输入事件交与系统响应；如果决定将输入事件发送到 TVOS 应用程序响应，则通过输入事件管理器中存储的窗口信息查找目标窗口，并将该输入事件通过窗口信息中的输入事件传输通道发送到对应的应用程序。

B.13　应用安装功能实现

B.13.1　应用安装组件的架构与接口策略

应用安装组件中相关的部件的接口模式，遵循 TVOS 框架中基于 Binder IPC 架构的 Proxy-Stub 模式，其接口模式如图 B.24 所示。

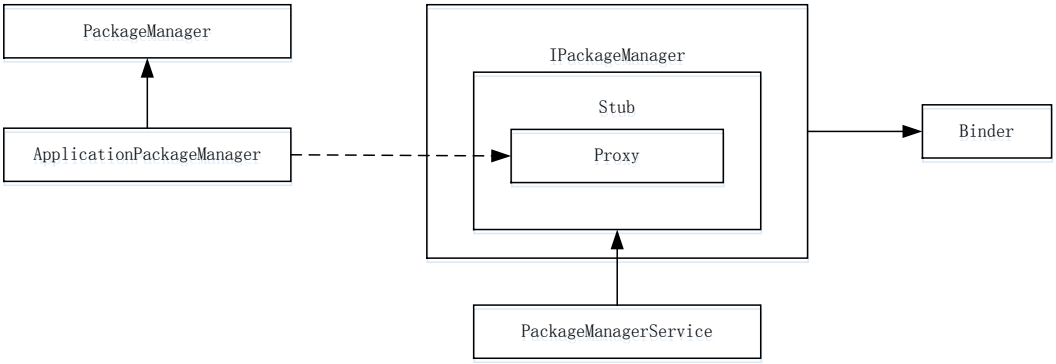


图 B.24　应用安装组件的接口模式

IPackageManager 接口类中定义了许多业务函数，出于安全方面的考虑，接口提供的只是一个子集，该子集被封装在抽象类 PackageManager 中。应用程序一般通过 Context 的 getPackageManager 函数返回一个类型为 PackageManager 的对象，该对象的实际类型是 PackageManager 的子类 ApplicationPackageManager。

B.13.2　应用安装组件的功能实现原则

B.13.2.1　安全

应用安装组件是 TVOS 安全防护的第一道门槛。应用安装组件在应用程序生命周期的不同阶段，会有不同的系统安全保障，以确保应用程序的完整性、来源和运行行为的可控性。体现在应用安装组件的设计中，需要能够满足以下几点需求：

a) 应用程序包的完整性进行检查，以确保应用程序包没有被其他人篡改过，被篡改过的应用程序包，应用安装组件拒绝安装；
b) 应用安装组件对应用程序的来源进行审查，对于没有经过应用商店审核过的应用程序，拒绝安装，应用安装组件只会接受来自于应用商店的安装请求；
c) 应用安装组件对应用程序的权限进行审查，如果应用程序想要获取系统级的访问权限，必须具备相应的签名，如果签名信息不匹配，应用安装组件必须吊销应用程序的相关权限，确保系统的安全性。

B.13.2.2　应用安装组件的单元考量

首先，应用安装组件功能明确，作为系统级的核心组件，在框架中具备以一个系统服务的形式提供功能；

其次，考虑到与应用管理组件、窗口管理组件等其他框架组件之间的关系，应用安装组件可遵循其他框架核心组件的模式，在 TVOS 的系统服务中启动和管理。

B.13.2.3　应用安装组件接口考量

在框架 API 中需要考虑应用程序的安装请求、应用信息查询、应用卸载等接口开发。

B.13.3　应用安装组件的启动过程

在应用安装组件启动时的主要工作是扫描 TVOS 系统中几个目标文件夹中的应用程序，从而建立合适的数据结构以管理诸如安装包信息、四大组件信息、权限信息等各种信息。其工作流程大体分为三个阶段：

a) 扫描目标文件夹之前的准备工作：系统 XML 文件扫描和解析，构建相关数据结构；

b) 扫描目标文件夹：扫描系统中的应用程序，完成 APK 的物理机构到数据结构的转化；
c) 扫描之后的工作：保存信息到文件中。

B.13.4 应用安装流程

要安装的应用程序包，首先应用商店将应用程序下载至设备端,然后通过应用安装组件的接口，向应用安装组件发起安装请求，应用安装组件在接收到安装请求后，执行流程如图 B.25 所示。其中处理流程描述如下：

a) 通过 verification 请求，让验证程序首先扫描需要安装的目标应用程序，这个请求是为了留给杀毒软件用的；
b) 确定安装位置：应用程序在 XML 中设置的安装点默认为 AUTO，在具体对应时倾向内部存储空间；
c) 检查应用程序包最小需求空间大小；
d) 检查应用程序包的完整性，以及应用商店签名是否完整；
e) 审核应用程序包的权限申请；
f) 为应用程序分配 UID；
g) 安装完成后，向 TVOS 系统广播有新的应用程序添加成功。

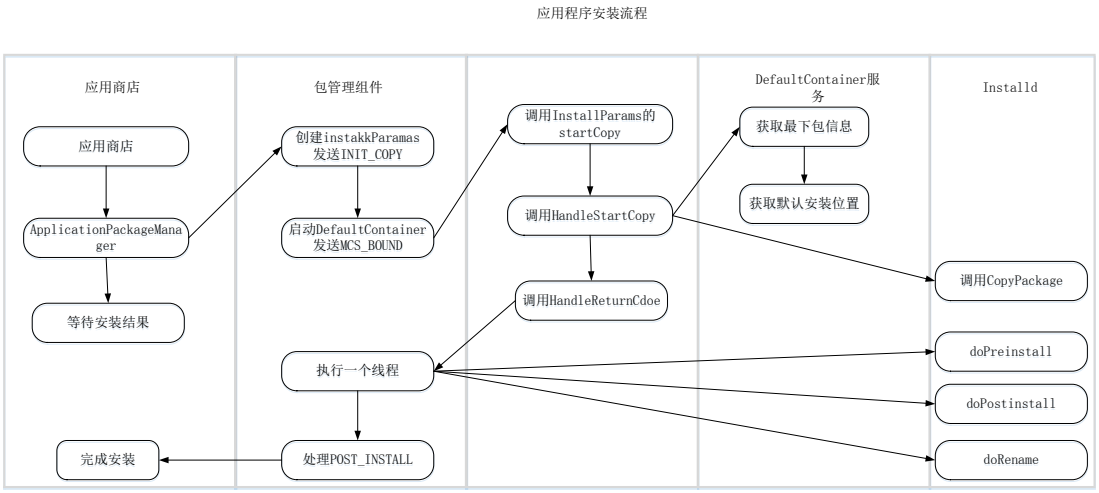


图 B.25 应用安装程序包

B.13.5 应用程序的验证

B.13.5.1 验证规则

TVOS 只允许通过统一的应用商店下载应用，不允许通过载体的拷贝进行分发。TVOS 采用数字签名技术来保证应用的来源合法性和数据完整性，通过应用自签名加上应用商店签名的方式确保合法应用能够正确下载与安装，未知或不可靠来源的应用将不能安装。

B.13.5.2 应用自签名

应用自签名是通过数字签名来标识应用程序的开发者和在应用程序之间建立信任关系，该数字签名由应用开发者完成或者由发布应用的公司统一签名，该签名并不需要权威的数字证书签名机构认证，它只是用来让应用程序包自我认证。

自签名过程如图 B.26 所示。

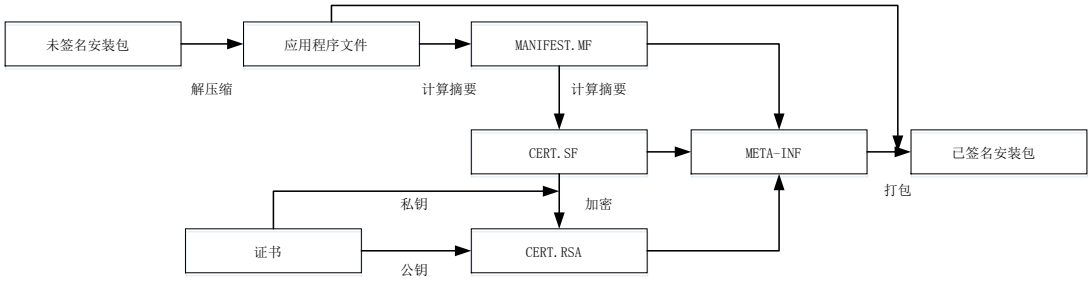


图 B.26　应用自签名过程

B.13.5.3　应用商店签名

为保证 TVOS 应用的安全性，在应用上架之前除进行必要的功能、性能、兼容性等测试之外，还需要对安全性进行检测，只有通过检测的应用才允许发布到应用商店中，并通过终端下载安装。

应用商店签名可确保此应用来源于合法应用商店，可以标识应用程序的安全性和合法性。

应用商店签名过程如图 B.27 所示。

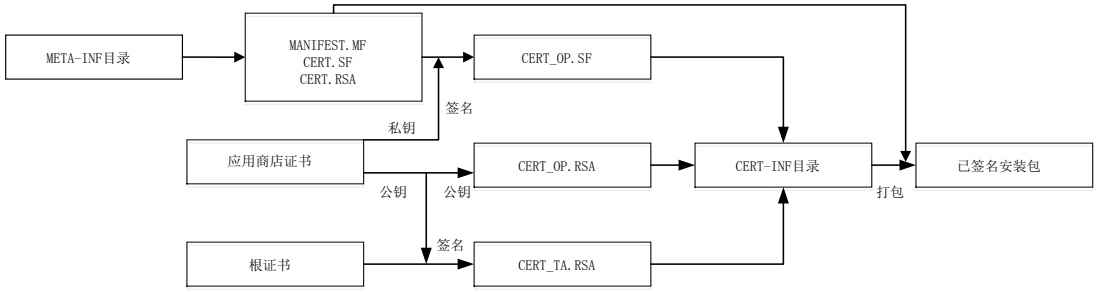


图 B.27　应用商店签名过程

TVOS 应用在执行安装过程之前必须通过签名校验。应用签名校验过程如图 B.28 所示。

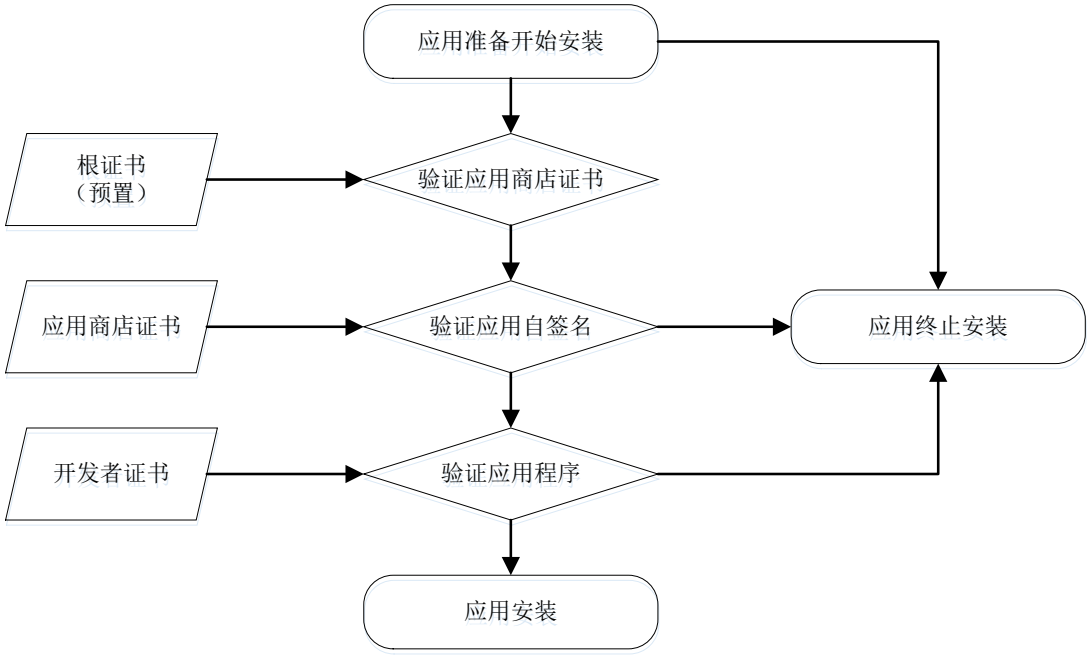


图 B.28　应用安装校验过程

验证不通过则终止应用在 TVOS 系统的安装。

中 华 人 民 共 和 国
广播电影电视行业标准
智能电视操作系统
第 1 部分：功能与架构
GY/T 303.1－2016
*
国家新闻出版广电总局广播电视规划院出版发行
责任编辑：王佳梅
查询网址：www.abp2003.cn
北京复兴门外大街二号
联系电话：（010）86093424　　86092923
邮政编码：100866