|  |  |  |  |
| --- | --- | --- | --- |
| **全智能相机平台软件设计方案书** | | | |
| 文件编号： |  | 当前版本： | V1.0 |
| 页数： |  | 发布日期： |  |
| 文件状态：初版■ 评审通过□ 正式发布□ | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 作者 | 彭元华、裴广涛 | | |
| 评审人员 |  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| 审核人员 |  | | |
| 批准 |  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修订详细记录** | | | | |
| **版本** | **修订内容详述** | **编制/修订人** | **批准人** | **修订日期** |
| V1.0 | 新建文档 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[目录 3](#_Toc18787)

[第1章 概述 5](#_Toc11718)

[1.1 目的 5](#_Toc1480)

[1.2 适用范围 5](#_Toc8428)

[1.3 参考资料 5](#_Toc3573)

[1.4 约定 5](#_Toc21420)

[第2章 总体设计 6](#_Toc17279)

[2.1 系统分析 6](#_Toc15058)

[2.1.1 业务系统分析 6](#_Toc9060)

[2.1.2 设计目标 6](#_Toc21817)

*[2.2](#_Toc14215)* [整体系统架构 6](#_Toc14215)

[2.3 软件系统架构 7](#_Toc14824)

[2.4 处理流程设计 8](#_Toc17621)

[2.4.1识别结果处理时序 8](#_Toc1038)

[2.4.2外部IO事件时序 9](#_Toc9171)

[2.4.3外部接入协议命令时序 9](#_Toc29673)

[第3章 模块设计 10](#_Toc24167)

[3.1 多媒体模块设计 10](#_Toc5085)

[3.1.1配置参数管理设计 10](#_Toc17687)

[3.1.2视频模块设计 10](#_Toc27462)

[3.1.3音频模块设计 11](#_Toc32534)

[3.1.4算法模块设计 12](#_Toc21699)

[3.2 协议&业务层模块设计 12](#_Toc29648)

[3.2.1接入协议层设计 13](#_Toc31770)

[3.2.2业务层设计 14](#_Toc8670)

[3.3 VZSDK设计 15](#_Toc26585)

[3.3.1架构设计 15](#_Toc22148)

[3.3.2接口设计 15](#_Toc21856)

[第4章 通信协议设计 18](#_Toc4643)

[4.1 内部星形结构消息 18](#_Toc8710)

[4.2 WebServer标准透传协议 18](#_Toc2379)

[4.3 TcpServer标准透传协议 19](#_Toc4708)

[第5章 文件结构 23](#_Toc28496)

[第6章 维护设计 24](#_Toc4375)

[附录A 附录 25](#_Toc24206)

[附录B 图、表目录 26](#_Toc2167)

# 概述

## 目的

随着公司产品线的不断丰富以及业务逻辑的多样化，当前的软件整体架构在支撑各个产品过程中日显捉襟见肘，新产品的开发缺少统一的软件架构、模块间耦合严重、代码\人力复用率低、开发效率低下，无法支撑后续产品的可持续发展。

为了解决上述问题，我们需要设计一个架构统一的、可扩展的软件平台，支撑各个产品线业务需求，提升代码\人力的复用率，达到快速、高质量支撑各个产品线的目的。

## 适用范围

本文档用于开发内部，用于指导软件开发。

## 参考资料

## 约定

*【列出本文档中所用到的专门术语、缩写、规范等约定】*

# 总体设计

## 系统分析

### **2.1.1 业务系统分析**

当前我司的产品都是基于实时视频的智能安防设备，各产品在视频输入\输出\处理、识别结果管理\推送、对外接入协义等主要功能以及主线处理流程上一致，差异只在算法模块和产品业务模块。

**主要的共同点**：

1）系统构成：嵌入式设备搭载Linux系统(HuaWei Linux、LiteOS、标准Linux)。

2）业务流程：采集视频->算法(编码、AI识别)->推给客户(实时预览、识别结果)。

3）回放流程：采集视频->算法(编码、AI识别)->设备存储(识别结果、视频)->客户查询(结果、视频回放)。

4）控制流程：用户配置->协议转发->模块处理。

**主要的差异点**：

1）硬件不同，因为算力和外设接入能力，产品有特殊要求。

2）算法不同，人脸设备和车牌设备以及智慧盒子需要集成不同算法，适配不同业务流程。

3）用户展示不同，需要配对算法输出结果和对算法进行特殊配置。

当前各产品中这些差异点没有充分的剥离，模块间耦合大，个别模块主线处理流程不统一，导致开发\维护成本高，并且无法在一个平台上兼容多种业务模型。

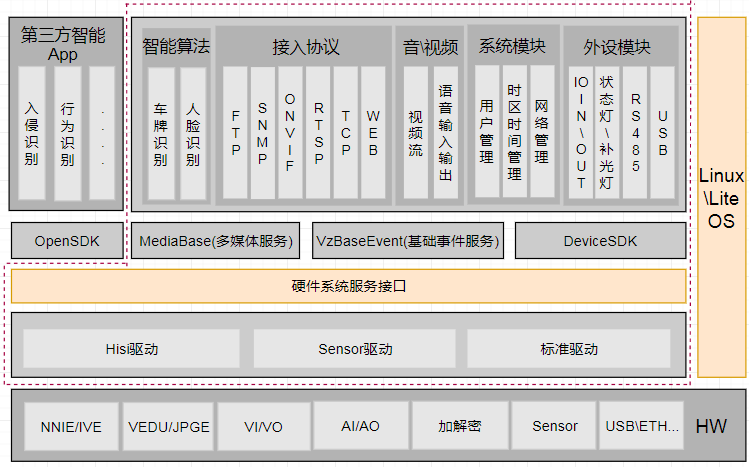
### **2.1.2 设计目标**

定义一个结构化的、通用的智能相机平台，可同时支撑多种业务模型，具体目标如下：

1. 统一各种业务模型的主线处理流程。
2. 将具体业务相关功能充分剥离，提升代码的复用率。
3. 减少应用软件、算法、硬件之间的耦合；减少应用软件模块间的耦合。
4. 同时可以支持多种智能应用业务。
5. 支持设备端\服务器端的客户二次开发。
6. 支持单进程运行，支持任意模块拆分或合并为多个进程运行。

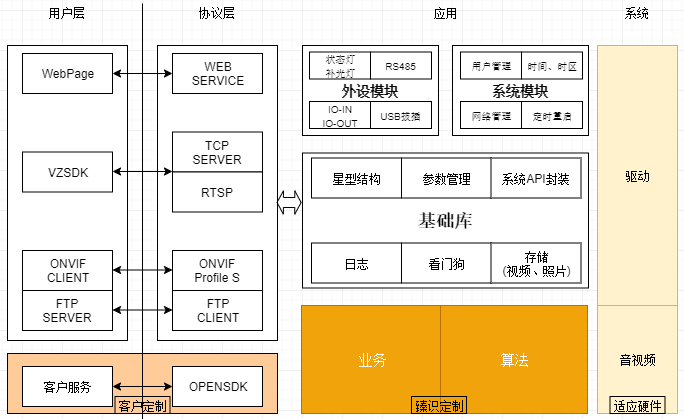
## 整体系统架构

根据设计目标，要求系统自身可加载多种预置的智能算法业务，同时用户也可以在设备终端基于OpenSDK自定义智能App，或者在PC端通过多种协议接入使用设备端的服务，如，音\视频服务、系统服务、外设服务、运维服务(日志、配置参数、存储)等。对系统功能、服务拆解如下图所示：



## 软件系统架构

软件系统架构设计如下图所示：



整体划分为4层：系统服务层、应用层、接入协议层、用户层。

* **接入协议层**：用户和相机的交互通道，对外提供的服务都经过此层传输，解析成内部协议转给应用层处理。

**协议模块(Web\Tcp\RTSP\ONVIF\FTP)**：不关心命令内容，仅做协议解析、转发，然后通过星型结构消息直接透传到后端应用模块进行处理。

**OPENSDK**：设备端客户二次开发工具包，支持客户定制协议，HTTP推送、TCP协议等。

* **应用层**：此层细分为业务相关和业务无关模块，各模块之间通过DP消息通信，降低模块间的耦合。

1）业务无关模块：

**系统模块**：系统配置相关功能，提供应用整体运行环境，包括网络环境配置、时区时间管理配置、设备定制重启、设备硬件信息管理、设备名称管理、登录用户管理。

**外设管理模块**：用于统一设备对外设控制的方式，包括：状态灯、补光灯、IO-IN\OUT、RS485输入\输出、USB拔插信号。

**基础服务模块**：MediaBase，多媒体服务，主要包括：图片、视频以及数据存储&查询，音频输入和输出，实时视频流RTSP服务；VzBaseEvent，基础事件服务，主要包括：进程\线程管理、进程\线程间通信、配置参数管理、日志服务、看门狗、网络通信服务、系统服务封装等。

2）业务相关模块：

算法+业务：产品定义相关，产品独有特性。可能是车牌识别，可能是人脸抓拍，可能是人脸抓拍+人脸识别等；根据不同的算法结果需要对其进行相应的应用层处理，比如调用存储，白名单比对，外设联动等应用业务处理。

* **系统层：**屏蔽系统差异，为应用层提供统一的开发接口。

## 处理流程设计

### 2.4.1识别结果处理时序



### 2.4.2外部IO事件时序



### 2.4.3外部接入协议命令时序



# 模块设计

## 多媒体模块设计

**当前存在的问题**：

* 和算法模块耦合严重，多媒体中存在较多的算法相关业务逻辑；
* 在车牌、人脸项目中业务处理流程不统一；
* 兼容硬件差异、海思sdk版本较麻烦。

**设计目标**：实现模块的平台化，在不同产品、平台上能够做到快速移植、开发。多媒体业务分为音频、视频、算法，各部分设计要求如下：

* 音频：音频业务功能固定，新产品迁移只需关注音频质量与海思sdk的兼容。
* 视频：视频业务功能固定，和算法完全解耦，多媒体模块只提供基础的接口，如结果图片生成、叠加OSD等，新产品迁移只需关注视频配置的不同与海思sdk的兼容。
* 算法：算法业务功能固定，新产品迁移只需关注海思sdk的兼容。

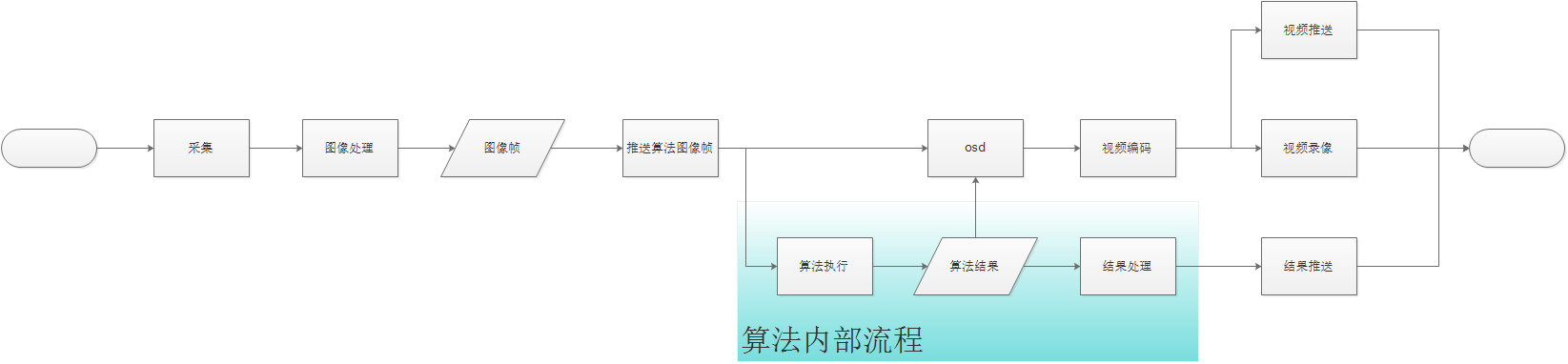
### **3.1.1配置参数管理设计**

由于不同的硬件差异或者海思sdk版本的更新，在以前的实现中，我们需要通过修改代码来实现音频、视频参数的适配、升级，较麻烦。

而通常海思音频、视频接口，很少修改函数名和形参，而传入海思接口的参数，一般分为两类，基本类型与指针。所以我们可以通过加载配置文件的参数、序列化数据，传入到hisi接口。从而兼容不同版本的sdk。这种设计方法，对以后的平台化工作极为方便，研发人员只需要修改配置参数就可实现兼容，做到新产品的代码快速迁移。

### **3.1.2视频模块设计**

视频处理逻辑包括：视频编码、图像配置、结果配置、视频推送。处理流程如下：



如上图所示，多媒体模块与算法模块完全解耦，固化视频的处理流程，仅包含vpss、encode、OSD叠加等，不再有任何与算法相关逻辑(包括结果处理和视频帧策略)。多媒体与算法的数据交互，采用的是数据透传。

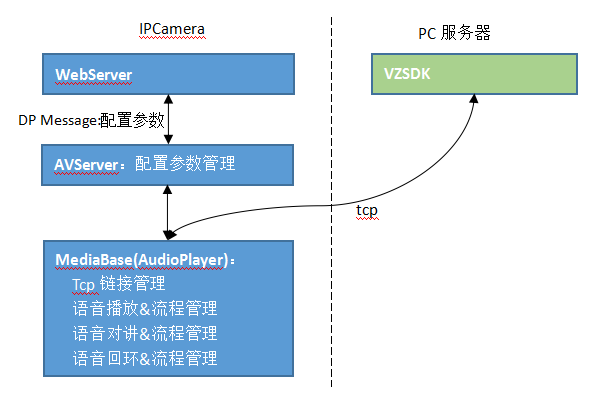
**实时结果**：osd直接叠加到图像上，由算法实现osd的内容，多媒体将内容叠加到图像。

**事件结果**：结果Osd的叠加采用的是扩展区域的叠加，叠加的内容同样由算法实现。

**阻塞与非阻塞**：推送图像帧到算法，这是个同步的过程。算法可以修改图像帧的内容，往图像帧拷贝当前的算法帧。如果算法不拷贝，就是非阻塞，拷贝图像就是阻塞。

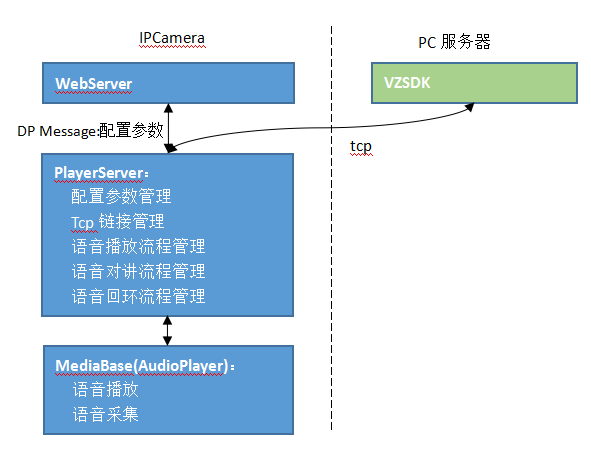
**结果推送**：多媒体不做转换，直接推送结果

### **3.1.3音频模块设计**



当前的音频播放、采集整体框架如上图所示，语音的输入输出、业务流程控制都在MediaBase中实现，导致基础音频服务接口和业务耦合较大，并且稳定性较差：

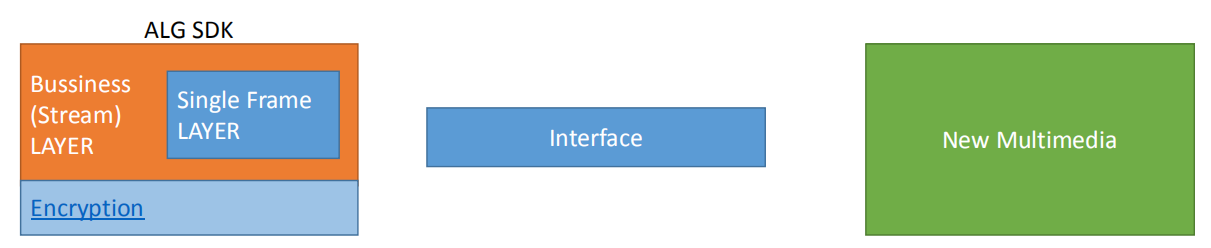
1. Tcp链接管理单独实现，未做到代码的复用，问题多，链接稳定性差。
2. 控制通道和数据通道分离，连接建立过程繁琐。
3. 音频采集、播放作为基础服务完全和业务耦合，扩展性差。
4. 内部流程状态管理实现复杂、参数配置管理混乱，导致可扩展性、可维护性差。



针对当前的问题，优化后的架构如上图所示：

1. 将Tcp连接管理、语音对讲、播放等和业务相关流程控制移到应用层PlayerServer模块实现。
2. Tcp连接管理、数据收发使用基础库的接口。
3. 控制通道和数据通道复用，简化连接建立过程。
4. MediaBase中仅提供基础的音频播放、采集服务，完全和业务解耦，并且语音播放和采集独立，方便业务扩展。

### **3.1.4算法模块设计**



**ALG SDK**（完全由算法开发）：

* 基于传统ALG接口创建（init, process, getResult, setParam）。
* 所有的结果传递，参数传递，均用结构体实现。
* 此模块可向外销售。

**Interface**（可以由算法开发，也可以由多媒体开发）：

* 脱胎于传统多媒体部分。
* 提供回调函数 recall\_GiveAndGetFrame，可使New Multimedia模块向ALG SDK传递图像帧。
* 可调用New Multimedia提供的编码模块，向外透传结果及编码好的图片。
* Interface对所有传入参数做解析，对传出结果做解析。参数及结果均由json实现。
* 此模块可以控制阻塞与非阻塞模式， 通过在回调recall\_GiveAndGetFrame中，拷贝当前帧/结果帧实现。
* 此模块可以控制结果是否编解码，通过回调recall\_EncodeAndResult中，是否编码结果帧实现。
* 此模块可以控制叠加OSD方式， 通过回调recall\_GiveAndGetFrame中，叠加OSD方式实现。

## 协议&业务层模块设计

整体目标：将各模块中产品特有的业务流程剥离，固化通用的功能、业务流程，提升代码的复用率；并为产品特有的需求提供一种可扩展的加载方式，方便功能定制。

总体结构如下图所示：



### **3.2.1接入协议层设计**

**当前存在问题**：

1. 和业务耦合，协议模块中包含业务相关的处理逻辑，产品需求扩展繁琐。
2. 对业务层事件(消息)预定义，导致无用消息触发过多，消耗系统资源。

**设计目标**：和业务解耦，将具体业务流程剥离，方便各产品需求的扩展。

我们将具体业务流程从各协议模块剥离，协议模块仅实现基础的协议管理、数据转发，不关心命令的具体内容，将协议解析并通过星型结构消息直接透传到后端应用模块进行处理。这样协议模块的功能固化下来，后端增加新的模块、实现新功能时，协议模块不需要修改，仅用户层和应用层模块协商具体协议即可，方便各产品需求的扩展。

* 私有协议(TCP服务、WEB服务(HTTP，WebSocket))。

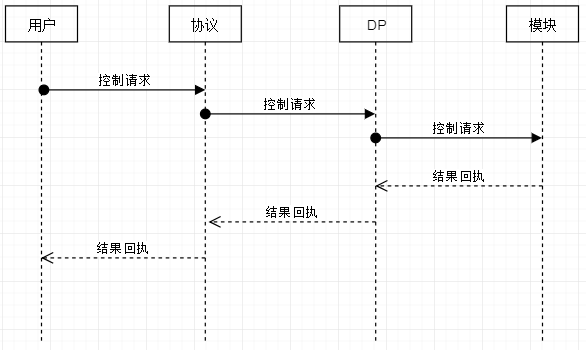
模块内部设计分为本地处理协议和透明转发协议。

本地处理协议：包括：登录、退出、获取硬件信息、检查超时时间这些不需要登录就可以访问的接口。

* 通用协议(ONVIF ProfileS、FTP)。

此外，各协议层监听的DP消息，采用定制的方式进行注册，例如，当客户端连接到TcpServer时，客户端根据业务需求主动注册对应的DP消息，TcpServer只作为消息收发的代理，当客户端断开时TcpServer取消对该消息的注册。和之前预定义的方式相比，可以减少大量的无效DP消息。

上下行消息转发流程如下图所示：



### **3.2.2业务层设计**

产品业务层，拆分为产品通用业务服务、产品差异化业务服务。差异化业务模块以接口方式对通用业务模块提供服务。

以BusinessServer为例，BusinessServer.cpp中只处理各产品通用的业务，剥离差异化的部分。各个产品定制、差异化的业务通过BusinessInterface对外提供统一的接口：外设管理(屏显、语音等)、脱机管理、识别结果处理等，各个产品实现class BusinessXxx，实现对外接口，如，BusinessCar、BusinessFace等。

BusinessServer作为整个模块DP消息的入口，对于通用的消息，处理后再转发到各个子模块处理；子模块定制的消息，直接转发到子模块处理。

Virtual int Init();

Virtual int DealDpMsg(DpMessage::Ptr dp\_msg, Json::Value& resp\_json);

以车牌一体机为例，类关系如下。



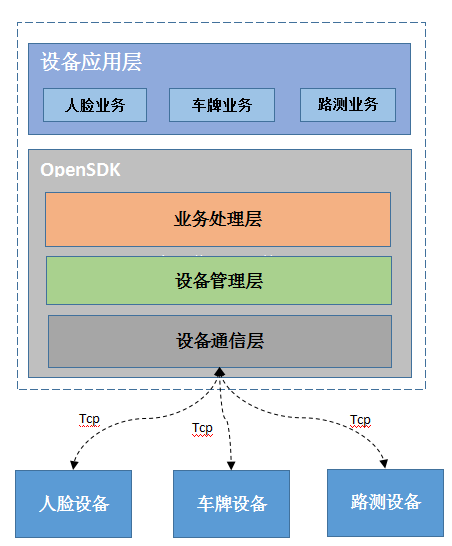
## VZSDK设计

### 3.3.1架构设计

**当前存在的问题**：当前SDK中，已经考虑了不同类型设备的兼容，但是与嵌入式设备通迅层和业务层代码耦合在一起的，不利于不同类型设备的扩展。

**设计目标**：架构清晰，减少模块间的耦合，兼容不同类型的相机，方便后续各产品业务的扩展。

基于上述目标，我们把与设备通讯层、业务处理层分开，设备通讯层只负责传输数据，上层业务层负责处理各种业务。通过分层，减少代码的耦合，使扩展维护更加容易。



上图是整个SDK的主要功能架构图，当然在这个图上面的功能并不完整，除了与设备和SDK交互通讯外，视频连接部份并不包含。由于有些业务在视频部份处理，比如报警流使用之前的视频播放库，提供直接传入窗口句柄播放视频、解码回调、报警流回调、原始数据回调的接口供上层业务处理，比如录像、绘制等功能。

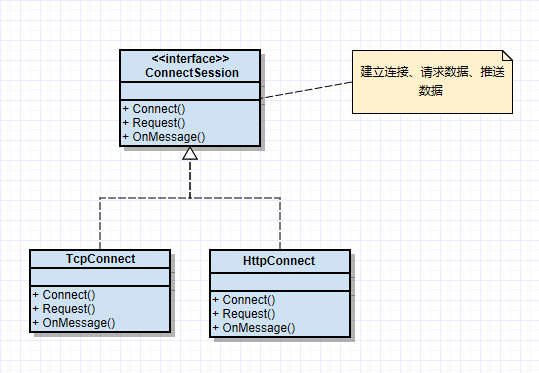
**设备通信层**：只处理发送和接收二进制数据，接收到请求透传到设备管理层，然后传到设备层进行业务处理。tcp通讯层负责发送tcp命令和返回结果数据，http通讯接口发送get、post请求和返回结果数据。

**设备管理层**：管理设备、执行登录、断线重连、心跳等。

**业务处理层**：屏蔽内部通信协议, 客户只需要通过不同API的调用，拿到返回值使用. 整个业务层需要分为所有产品线的公共业务和产品线上的私有业务。

### **3.3.2接口设计**

下面主要描述设备通讯层的接口设计，通讯层只负责发送和推送数据，数据为二进制数据，不关心数据的格式和内容。这样就可以达到对任意设备做到通用的目的。



接口主要包括三个部份:

1. Connect连接设备，建立好连接后，就可以发送请求和推送数据。

|  |  |
| --- | --- |
| 接口描述 | 连接设备 |
| 接口 | int Connect(const char\* device\_ip, int port, const char \*user\_name, const char \*password, int timeout, int type ); |
| 参数说明 | [IN]device\_ip 设备的IP地址  [IN]port 设备端口  [IN]user\_name 访问设备所需用户名  [IN]password 访问设备所需密码  [IN]timeout超时时间(单位毫秒)  [IN]type登录类型 0:tcp登录、1: http登录 |
| 返回值 | 返回值类型（int)，返回值为连接句柄，-1表示失败 |

1. Request请求分为两类，如果是http请求需要带参数get/post参数，tcp请求不需要，这里需要在接口中增加请求类型的参数，上层来区分。

|  |  |
| --- | --- |
| 接口描述 | 请求数据 |
| 接口 | int Request(const void \*data, int data\_size, int timeout, int type, std::string& reponse\_data); |
| 参数说明 | [IN]data 发送的数据  [IN]data\_size 数据长度  [IN]timeout超时时间(单位毫秒)  [IN]type登录类型 0:tcp请求、1: http get请求 2: http post请求  [out]reponse\_data返回数据(string可以保存二进制数据) |
| 返回值 | 返回值类型（int)，返回值为0代表成功，-1表示失败 |

1. OnMessage用于主动推送数据，主要是通过tcp连接来推送。

|  |  |
| --- | --- |
| 接口描述 | 推送数据 |
| 接口 | void OnMessage(const void \*data, int data\_size,); |
| 参数说明 | [out]data 发送的数据  [out]data\_size 数据长度 |
| 返回值 | 无 |

# **通信协议设计**

为方便各模块内部以及外部的协议对接，针对智能相机相关通信协议，做一个统一化的规定，后期所有协议按照以下格式定义进行开发。

首先，所有内部星形结构消息，请求以及回复采用统一格式，统一字段。

其次，外部协议（Tcp/Web）基于内部模块间的协议做一层统一格式的封装，解析之后直接透传到各模块，外部请求消息中需要携带相关内部消息的属性信息。

## 4.1 内部星形结构消息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 协议功能 | 内部模块之间交互使用 | | | | |
| Request | {  "type" : "do\_something",  "body" : {  // 透传内容  }  } | | | | |
| 字段解释 | 字段名称 | 字段含义 | 取值范围 | 是否必须 | 备注 |
| type | 消息类型 | String(0,32] | 是 |  |
| body | 消息内容 |  | 否 | Get类消息可没有此字段 |
| Response | {  "type" : "do\_something",  "state" : 200,  "err\_msg" : "All Done",  "body" : {  // 透传内容  }  } | | | | |
| 字段解释 | 字段名称 | 字段含义 | 取值范围 | 是否必须 | 备注 |
| type | 消息类型 | string(0,32] | 是 |  |
| state | 回复状态 | int[0, 1000] | 是 |  |
| err\_msg | 错误信息 | string(0,32] | 是 |  |
| body | 消息内容 |  | 否 | Set类消息可没有此字段 |

## 4.2 WebServer标准透传协议

统一请求消息：request.php

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 协议功能 | 网页请求协议消息 | | | | | | | | |
| Request | {  "type" : "do\_something",  "module" : "BUS\_SYS\_REQUEST",  "userinfo" : "asdgasdzvxcgad2",  "blockflag" : 1,  "body" : {  // 透传内容  }  } | | | | | | | | |
| 字段解释 | 字段名称 | 字段含义 | | | 取值范围 | | 是否必须 | 备注 | |
| type | 消息类型 | | | string(0,32] | | 是 |  | |
| module | 处理模块对应的星形结构消息 | | | string(0,32] | | 是 |  | |
| userinfo | “用户名：密码”使用固定字符串加密后的字符串 | | |  | | 否 | 短连接时需要此字段认证用户，长连接登录一次即可 | |
| blockflag | 内部转发时是否阻塞 | | | int[0, 1] | | 是 | 0:非阻塞的dpmsg;1:阻塞的dpreq | |
| body | 消息内容 | | |  | | 否 | 部分get类消息可不用此字段 | |
| Response | {  "type" : "do\_something",  "state" : 200,  "err\_msg" : "All Done",  "body" : {  // 透传内容  }  } | | | | | | | | |
| 字段解释 | 字段名称 | | 字段含义 | 取值范围 | | 是否必须 | | | 备注 |
| type | | 消息类型 | string(0,32] | | 是 | | |  |
| state | | 回复状态 | int[0, 1000] | | 是 | | |  |
| err\_msg | | 错误信息 | string(0,32] | | 是 | | |  |
| body | | 消息内容 |  | | 否 | | | 部分设置类消息可没有此字段 |

## 4.3 TcpServer标准透传协议

统一请求消息：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能 |  | | | | | |
| Request | {  "type" : "do\_something",  "id" : 123456,  "module" : "BUS\_SYS\_REQUEST",  "userinfo" : "asdgasdzvxcgad2",  "blockflag" : 1,  "body" : {  *// 透传内容*  }  } | | | | | |
| 字段解释 | 字段名称 | 字段含义 | 取值范围 | 是否必须 | 备注 | |
| type | 消息类型 | string(0,32] | 是 |  | |
| id | 当前消息ID | int | 是 | 用于客户端校验当前回复和当前请求是否匹配 | |
| module | 处理模块对应的星形结构消息 | string(0,32] | 是 |  | |
| userinfo | “用户名：密码”使用用户密码加密后的字符串 | String(0, 128] | 否 | 加密时第一次登录必须要有此字段，其他情况不用此字段 | |
| blockflag | 内部转发时是否阻塞 | int[0, 1] |  | 0:非阻塞的dpmsg;1:阻塞的dpreq | |
| body | 消息内容 |  | 否 | 部分get类消息不用此字段 | |
| Response | {  "type" : "do\_something",  "state" : 200,  "err\_msg" : "All Done",  "body" : {  // 透传内容  }  } | | | | | |
| 字段解释 | 字段名称 | 字段含义 | 取值范围 | 是否必须 | | 备注 |
| type | 消息类型 | string(0,32] | 是 | |  |
| state | 回复状态 | int[0, 1000] | 是 | |  |
| err\_msg | 错误信息 | string(0,32] | 是 | |  |
| body | 消息内容 |  | 否 | | 部分set类消息不需要 |

统一请求消息：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能 |  | | | | | |
| Request | {  "type" : "do\_something",  "id" : 123456,  "module" : "BUS\_SYS\_REQUEST",  "userinfo" : "asdgasdzvxcgad2",  "blockflag" : 1,  "body" : {  *// 透传内容*  }  } | | | | | |
| 字段解释 | 字段名称 | 字段含义 | 取值范围 | 是否必须 | 备注 | |
| type | 消息类型 | string(0,32] | 是 |  | |
| id | 当前消息ID | int | 是 | 用于客户端校验当前回复和当前请求是否匹配 | |
| module | 处理模块对应的星形结构消息 | string(0,32] | 是 |  | |
| userinfo | “用户名：密码”使用用户密码加密后的字符串 | String(0, 128] | 否 | 加密时第一次登录必须要有此字段，其他情况不用此字段 | |
| blockflag | 内部转发时是否阻塞 | int[0, 1] |  | 0:非阻塞的dpmsg;1:阻塞的dpreq | |
| body | 消息内容 |  | 否 | 部分get类消息不用此字段 | |
| Response | {  "type" : "do\_something",  "state" : 200,  "err\_msg" : "All Done",  "body" : {  // 透传内容  }  } | | | | | |
| 字段解释 | 字段名称 | 字段含义 | 取值范围 | 是否必须 | | 备注 |
| type | 消息类型 | string(0,32] | 是 | |  |
| state | 回复状态 | int[0, 1000] | 是 | |  |
| err\_msg | 错误信息 | string(0,32] | 是 | |  |
| body | 消息内容 |  | 否 | | 部分set类消息不需要 |

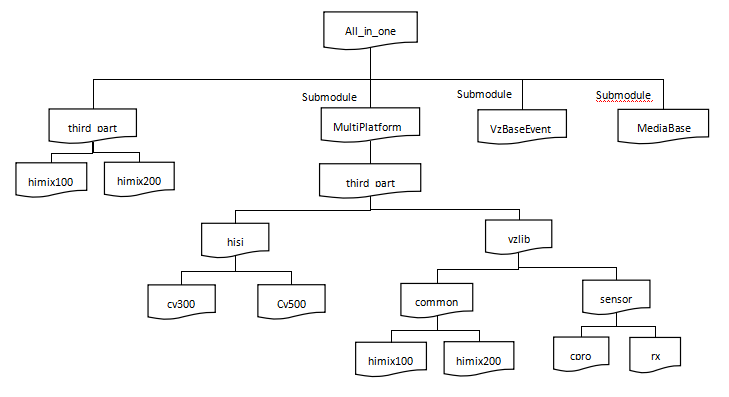
**其他模块协议详见：**

http://note.youdao.com/groupshare/?token=D03248788CCD42AF9EF6F4B63486B8EA&gid=94147904

# 文件结构

**文件\编译结构定义如下**：

* VzEventBase、MediaBase、MultiPlatform(AvSDK)作为submodule，所有的模块随着应用一起编译为静态库，禁止存在多级submodule。
* 公共的库（.so\.a）采用就近原则、唯一存放，如，hisi的库、osd.a、libsns\_vzsensor.so等只有AvSDK使用，就放到Multi\_Platform third\_part目录下，而libVzDeviceSDK.so等有多个部分使用库就放到应用third\_part\library目录下，禁止存在重复文件。
* 各个submodule中的子模块要尽量做到可独立编译（在app整体框架下）、运行。
* 基础服务模块（VzEventBase、MediaBase、AvSDK等）要做到和产品独立，只对外提供通用或用户可定制的服务。例如，xxx\_face.cpp是针对人脸业务，但其中又包含车牌业务。
* 各个产品要做到分支共用，原则上禁止拉分支，小的功能差异尽量做到可配置或者通过宏区分；大的差异并且独立，通过独立文件的方式扩展。



**文件结构&编码规范详见：**

http://note.youdao.com/groupshare/?token=B1B603015F6847B99D92BB0BF61B4F39&gid=94147904

# 维护设计

*【说明为方便维护工作的设施，如维护模块等。说明为了系统维护的方便而在程序内部设计中增加的一些专门用于系统的检查与维护的检测点和专用模块。如debug 信息，错误记录，关机记录等模块。*

*可从如下方面介绍：a, 内容描述； b, 资源需求；c, 设计流程**】*

1. 附录
2. 图、表目录