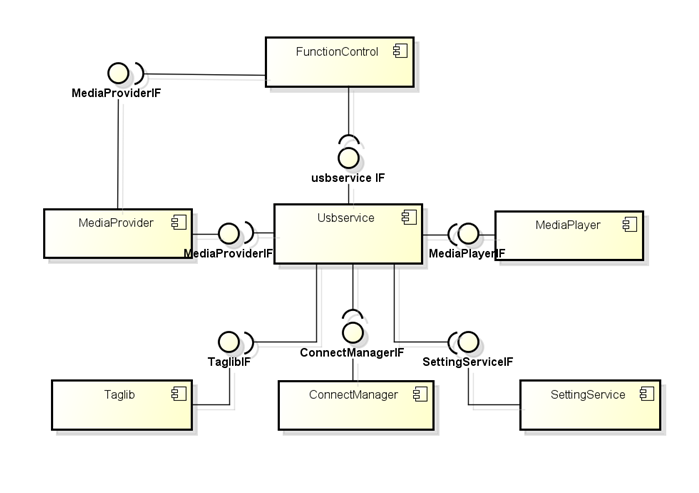
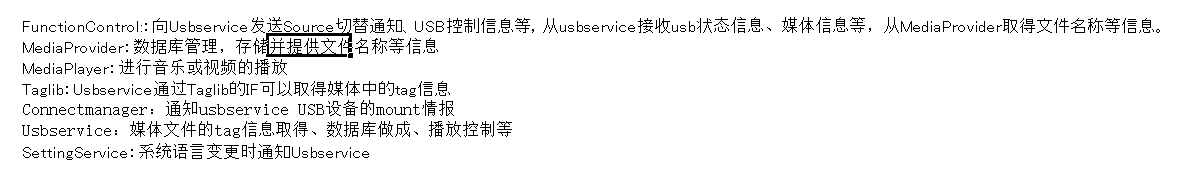
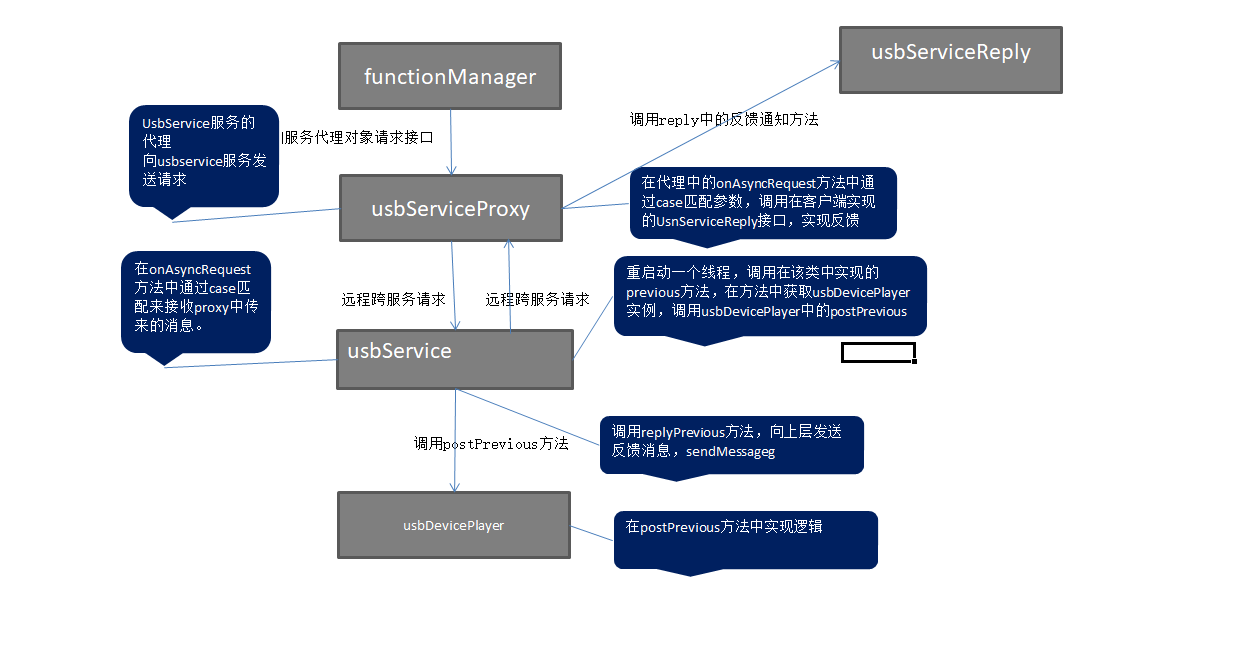
## 1 Usb与各个模块之间的调用关系





在usbservice中，上层与functionManager交互，下层与MediaService服务、DevicePlayerService服务、ConnectService服务交互。



## 2 Usbservice与各个模块交互的方式

### Usbservice与functionManager模块交互

向usbservice发送数据，和接受数据

在usbservice中，通过proxy与functionManager交互。

Proxy作为usbservice的代理接口，在proxy中的方法中sendMessage，该message会被GMUsbserviceBase中的onAsyncResponse方法中的case 接收。

而在UsbserviceBase中从serviceBasea中继承过来的sendAsyncRequest方法发送的消息，会被proxy中的onAsyncResponse方法中的case接收。

在usbService调用底层的逻辑处理请求前会将replyer应答反馈给上层。具体是，在Usbservice中发送sendMessage请求，在代理的onAsyncResponse方法中，以case匹配形式对应方法，实现将结果反馈给functionManager层。

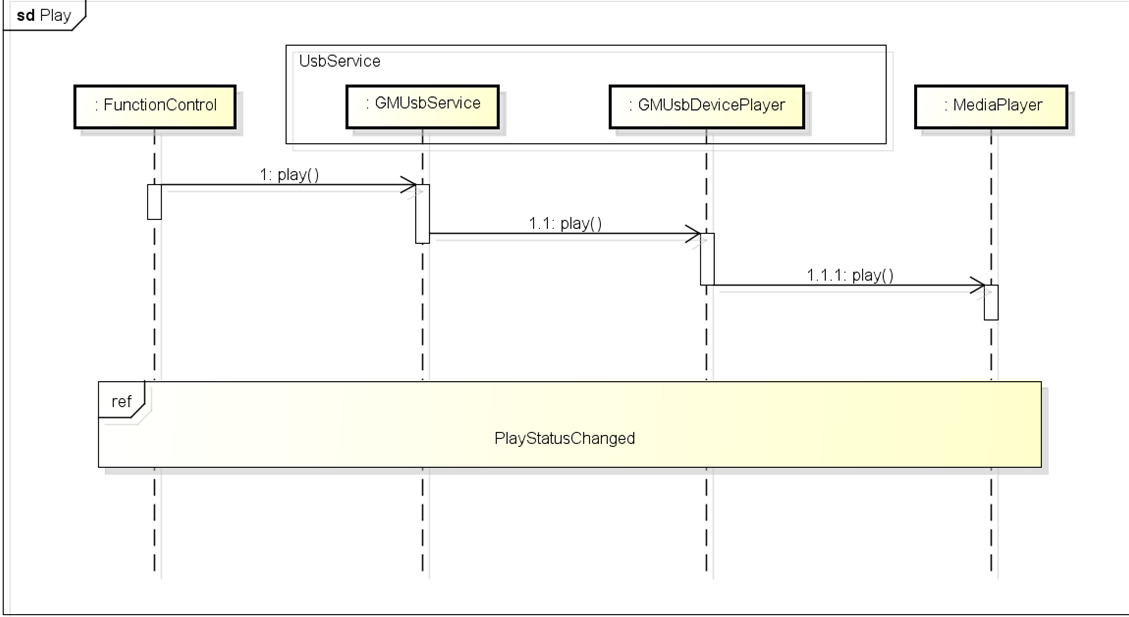
### UsbService与DevicePlayer模块的交互

在USbService中实现了plantform层的DevicePlayer的服务代理类NMMPClassicPlayerDelegation。在usbService中接受到message请求在方法onAsyncResponse中被case匹配处理。处理时，调用DevicePlayer中的方法，实现usbService服务与DevicePlayer服务的交互。

## 3 核心方法实现

### Play

#### 时序



#### 涉及的模块

FunctionManager

GMUsbService

MediaPlayer

#### 实现

functionManager 调用proxy代理中play方法

|  |
| --- |
| uint64\_t GMUsbProxy::play()  {  USBLOGD\_FUNCALL; //打log  //定义调用id，静态，unsigned long 型  static uint64\_t call\_id = 1;  unsigned int code = USBMethod\_Play; //方法id,通过枚举类型定义  //准备要发送的数据包  android::Parcel data;  prepareAsyncData(data);  // request service  //向usbservice发送异步请求,通过返回值判断请求是否成功。  if (BS\_NO\_CONNECTION == sendAsyncRequest(code, data)) {  return 0; // invalid call id '0' to indicate error  }  //调用完该方法后，返回call\_id+1  return call\_id++;  } |

proxy代理服务中的play请求，在usbService的onAsyncRequest方法中被接收，通过case匹配请求，选择对应的case进行处理。

|  |
| --- |
| int GMUsbServiceBase::onAsyncRequest(SenderId &id, unsigned int code, const android::Parcel &data) {  case USBMethod\_Play:  {  GMTaskDes des(Runnable\_Pri\_N, 0, "USBMethod\_Play");  (m\_pMainThread->looper()).postRunnable(GMTaskBind(des,this, &GMUsbServiceBase::play, id));  break;  } |