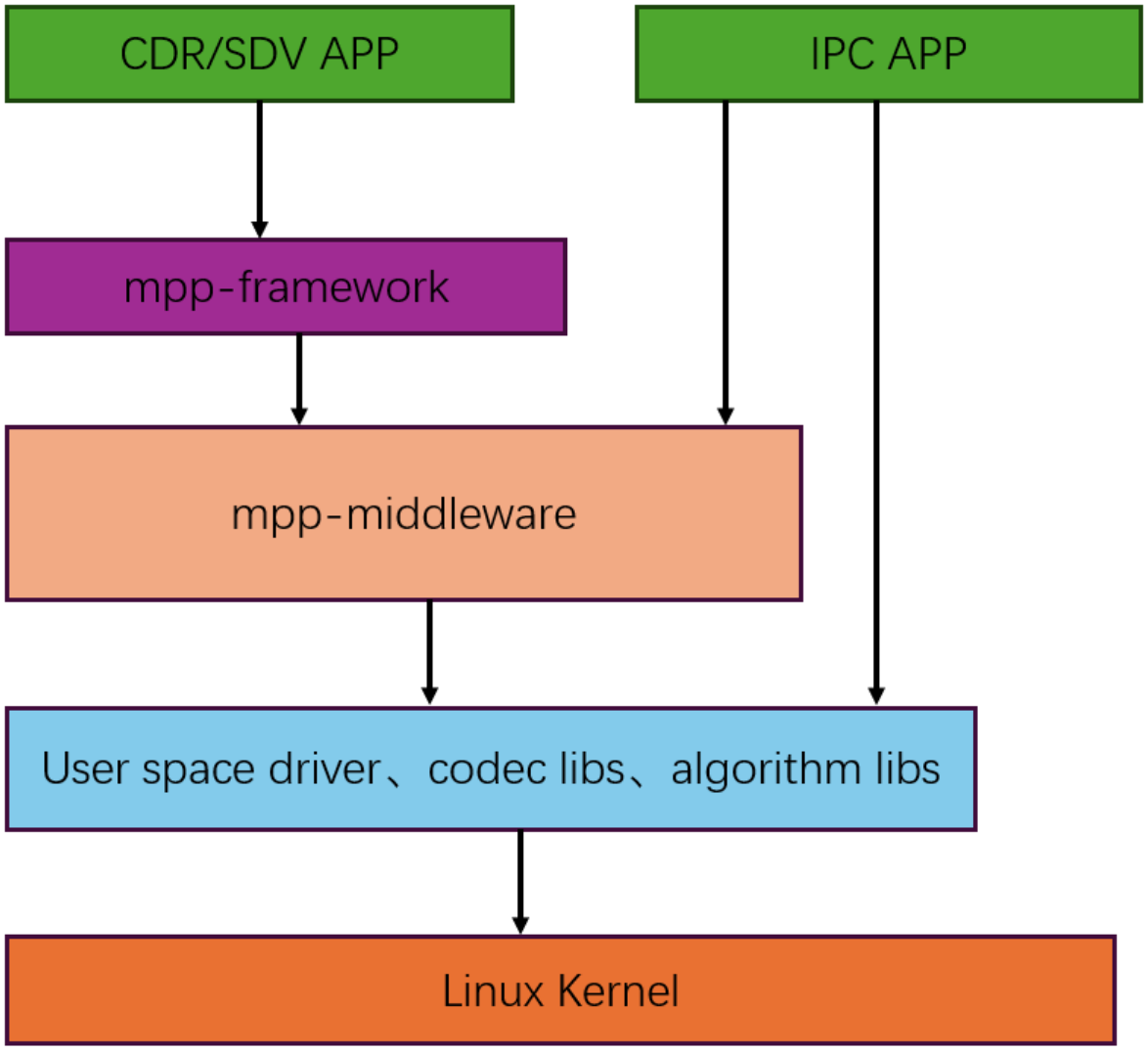
全志MPP平台基础

MPP平台是为监控领域Linux系统设计的多媒体框架，对音视频采集、编解码、输出，音视频智能分析，提供不同层次的整套API接口，满足不同客户的业务 开发需求。

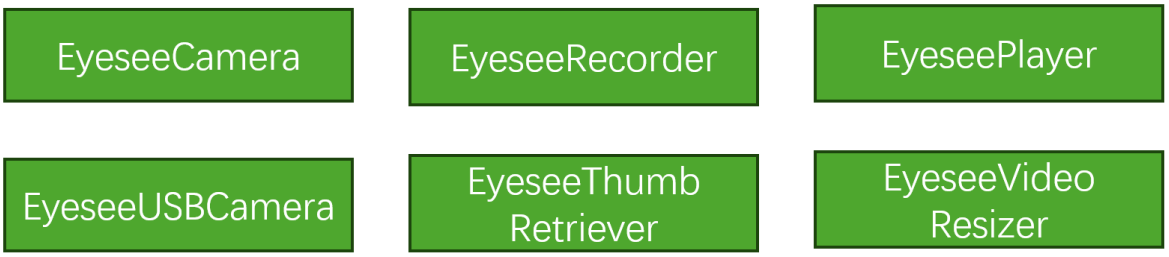
framework：集成度较高，模块数量少，接口粗密度

middleware：集成度较低，组件数量多，接口细密度



> mpp-framework是往下调用mpp-middleware层的各个组件
>
> mpp-middleware是往下调用用户态的驱动、编解码库、算法库等

1.MPP平台-框架



EyeseeCamera：摄像头图像采集

EyeseeRecorder：录制视频和音频文件

EyeseePlayer：播放视频

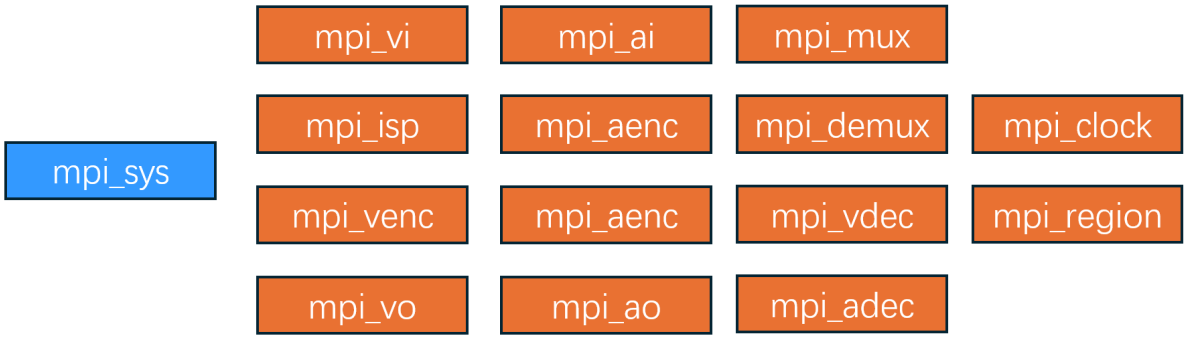
EyeseeUSBCamera：采集USB摄像头图像

EyeseeThumbRetriever：视频缩略图

EyeseeVideoResizer：视频重编码

2.MPP平台-组件

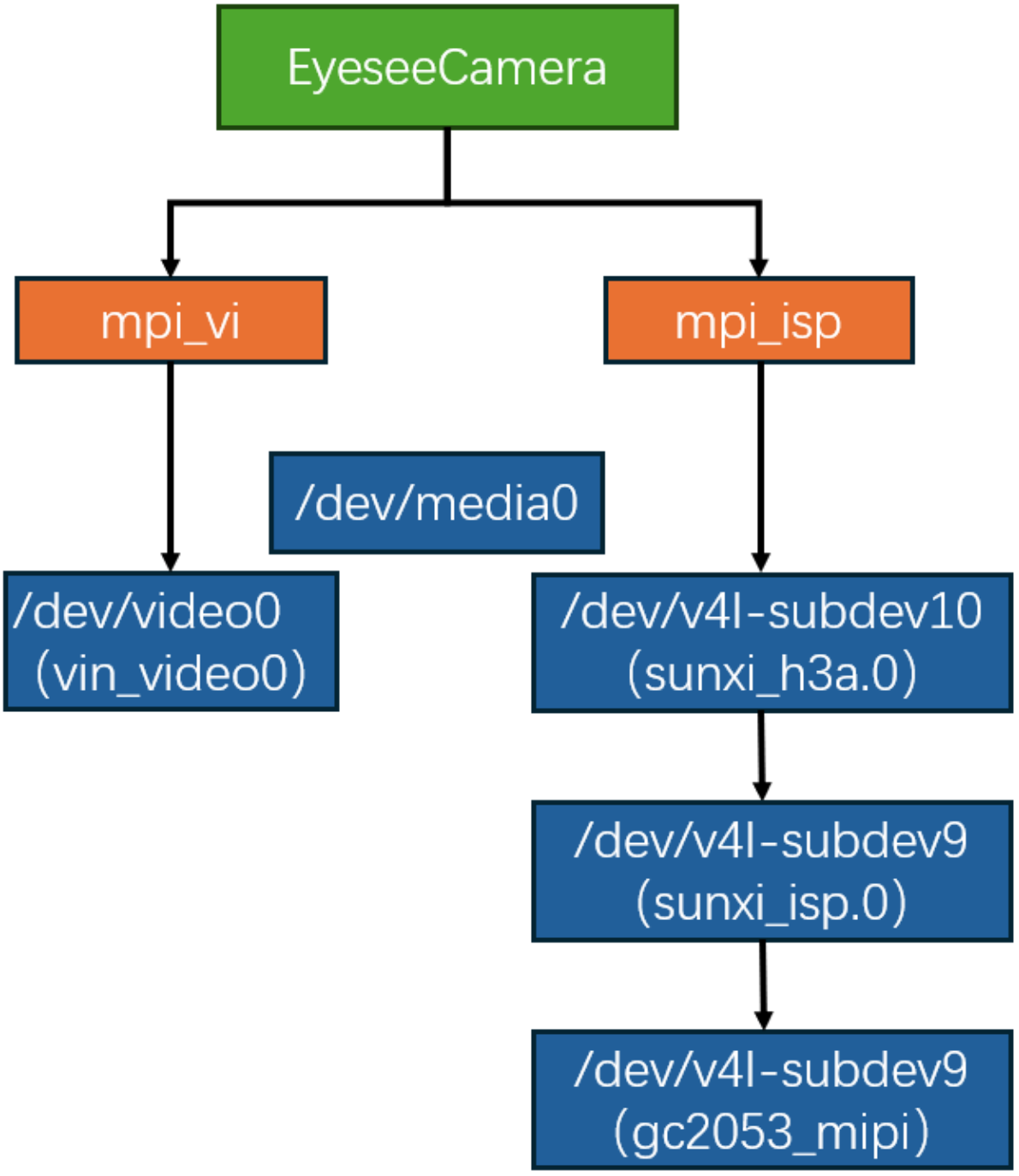
mpp-midedleware中的组件展示：



3.框架与组件的关系

对于Framework其实就是调用midedleware中的各个组件而成，下面以Framework中的摄像头图像采集为例：

Framework —> midedleware —>userspace_v4l2Driver



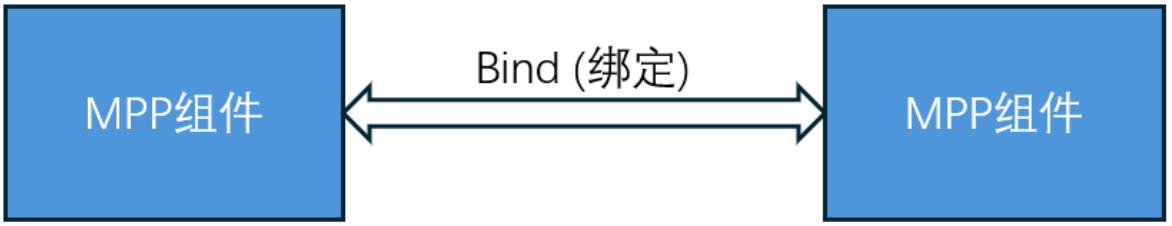
mpi_vi组件：使用摄像头驱动接口video0

mpi_isp组件：使用调整图像效果的驱动文件（由内核的V4L2驱动生成的）

4.MPP组件

4.1 架构原则

全志MPP平台底层基于OpenMAX 2L协议开发，由于OpenMAX对于多媒体编解码器定义了一个标准化的媒体组件接口，但是在安防领域中有一套通用的API,所以全志的MPP平台实际是底层的核心组件层的API重新封装，封装成安防领域中常用的API。



mpi层中的绑定关系对应的是核心组件中的隧道连接。

MPP组件的代码上层为MPP_API接口，下层为OpenMAX接口。

MPP_API接口

MPP_API：

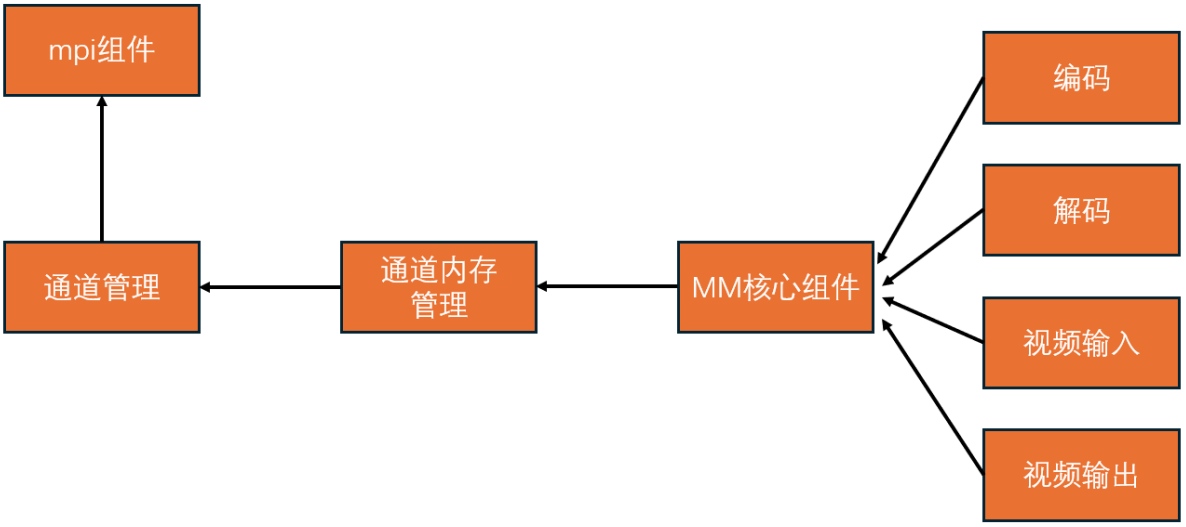
```
AW_MPI_VENC_CreateChn
AW_MPI_VENC_DestroyChn
AW_MPI_VENC_ResetChn
AW_MPI_VENC_StartRecvPic
AW_MPI_VENC_StopRecvPic
AW_MPI_VENC_RegisterCallback
AW_MPI_VENC_SetChnAttr
AW_MPI_VENC_GetChnAttr
AW_MPI_VENC_GetStream
AW_MPI_VENC_ReleaseStream
AW_MPI_VENC_SendFrame
```

Component接口

Component API:

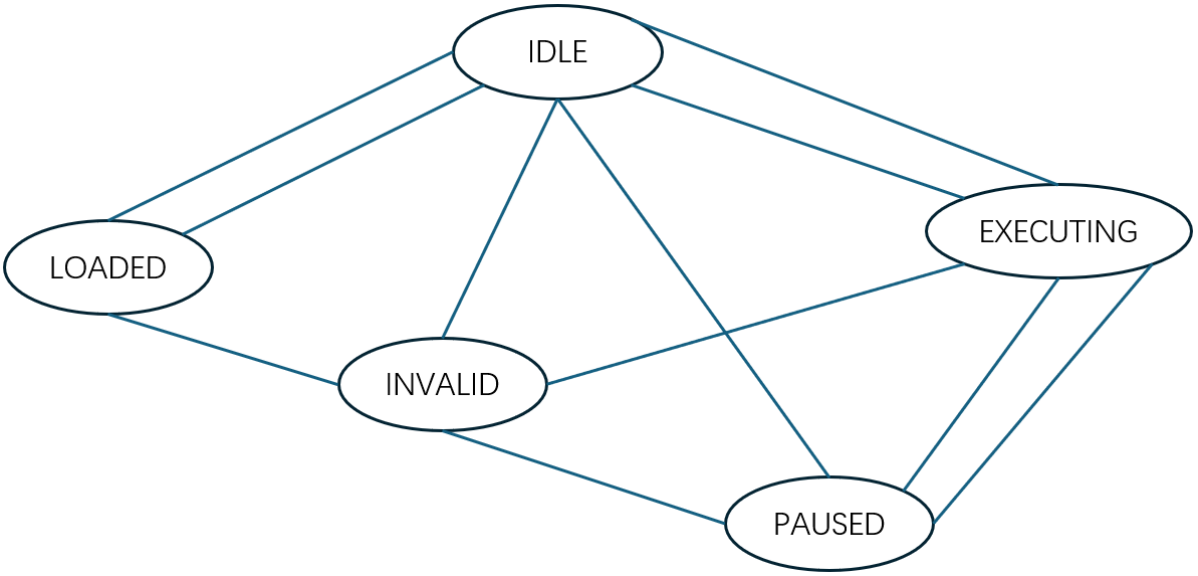
```
typedef struct MM_COMPONENTTYPE
{
        void* pComponentPrivate;
        void* pApplicationPrivate;
        ERRORTYPE (*SendCommand)();
        ERRORTYPE (*GetConfig)();
        ERRORTYPE (*SetConfig)();
        ERRORTYPE (*GetState)();
        ERRORTYPE (*ComponentTunnelRequest)();
        ERRORTYPE (*EmptyThisBuffer)();
        ERRORTYPE (*FillThisBuffer)();
        ERRORTYPE (*SetCallbacks)();
        ERRORTYPE (*ComponentDeInit)();
} MM_COMPONENTTYPE;
```

4.2 代码层次



mpi组件往下到MM核心组件分化出不同的功能，对于数据传输和状态转化都是一致。

4.3 状态转换



每一个组件内部都是有状态：

LOADED：初始创建时

IDLE：初始化完成后

EXECUTING：运行状态

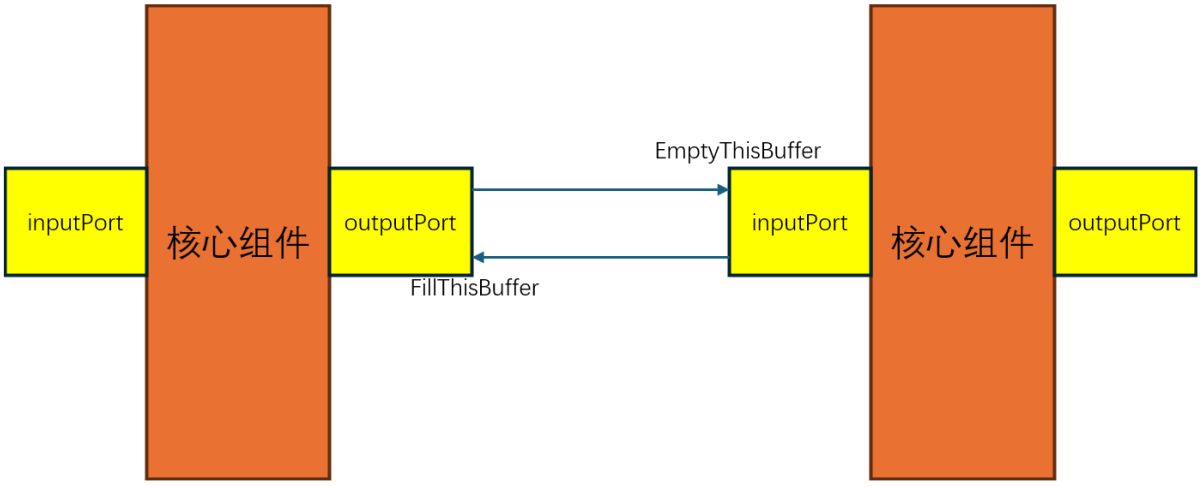
PAUSED：暂停状态

INVALID：运行错误状态

> 如果需要对组件进行绑定，最好在IDLE状态下进绑定

5.组件绑定原理

组件绑定，也称为建立隧道链接。两个组件建立隧道链接后，彼此可以直接传输数据或者做其他控制操作，而不需主控模块的干预。



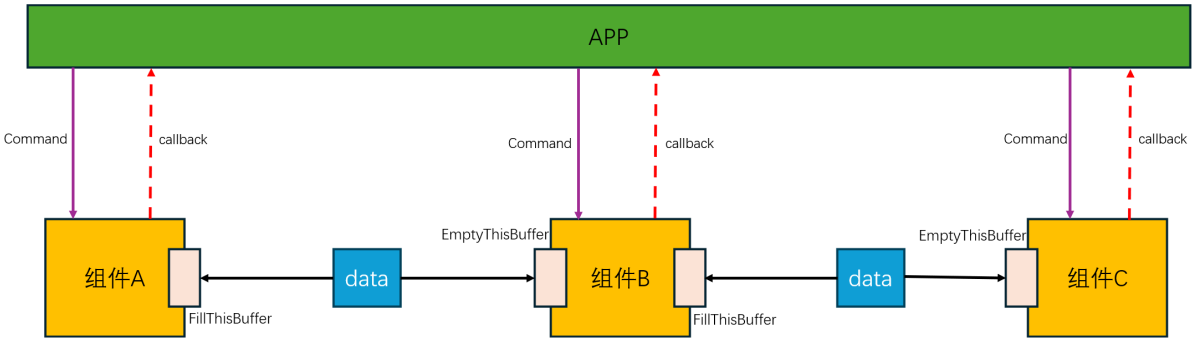
端口的输入/输出性质，在component初始化时已经确定。所以给定Port端口号后，组件知道自己的端口是输入有效数据还是输出有效数据。

隧道链接的原理是将双方组件的MM_COMPONENTTYPE*指针设置给对方，从而彼此组件内部可以直接调用对方接口。实现所谓的数据自动传输：

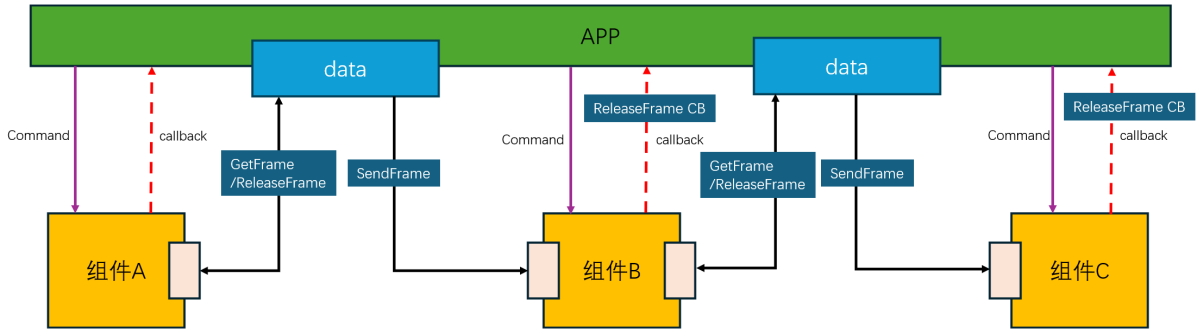
```
CompA ->B::EmptyThisBuffer() ->CompB
CompA <- A::FillThisBuffer() <- CompB
```

6.绑定与非绑定的数据处理

绑定模式（Tunnel模式）



非绑定模式（Non-Tunnel模式）



7.组件开发流程