# WebRTC手记之WebRtcVideoEngine2模块

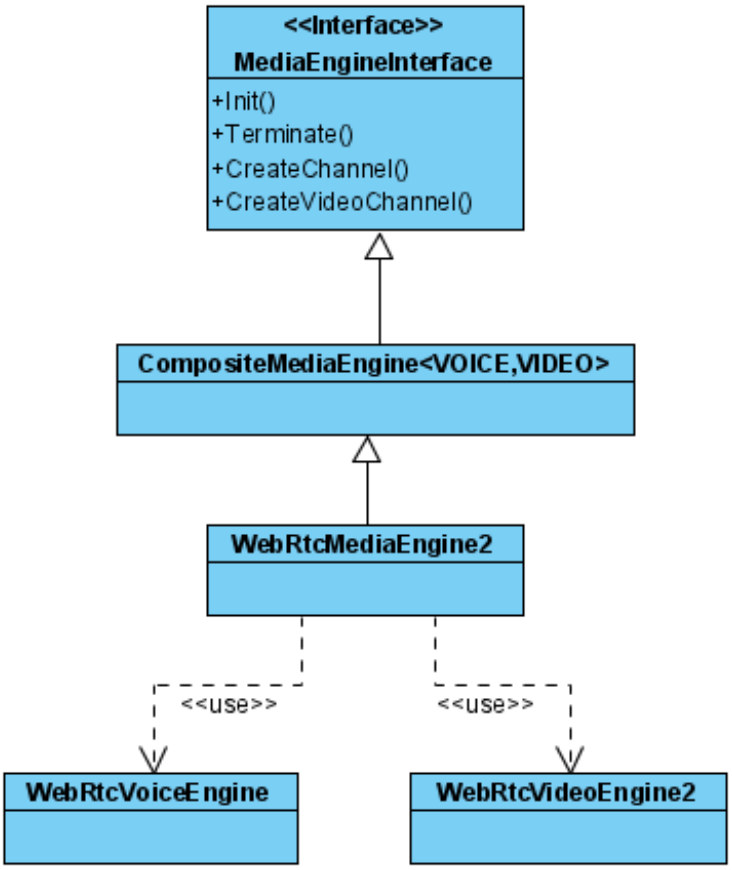




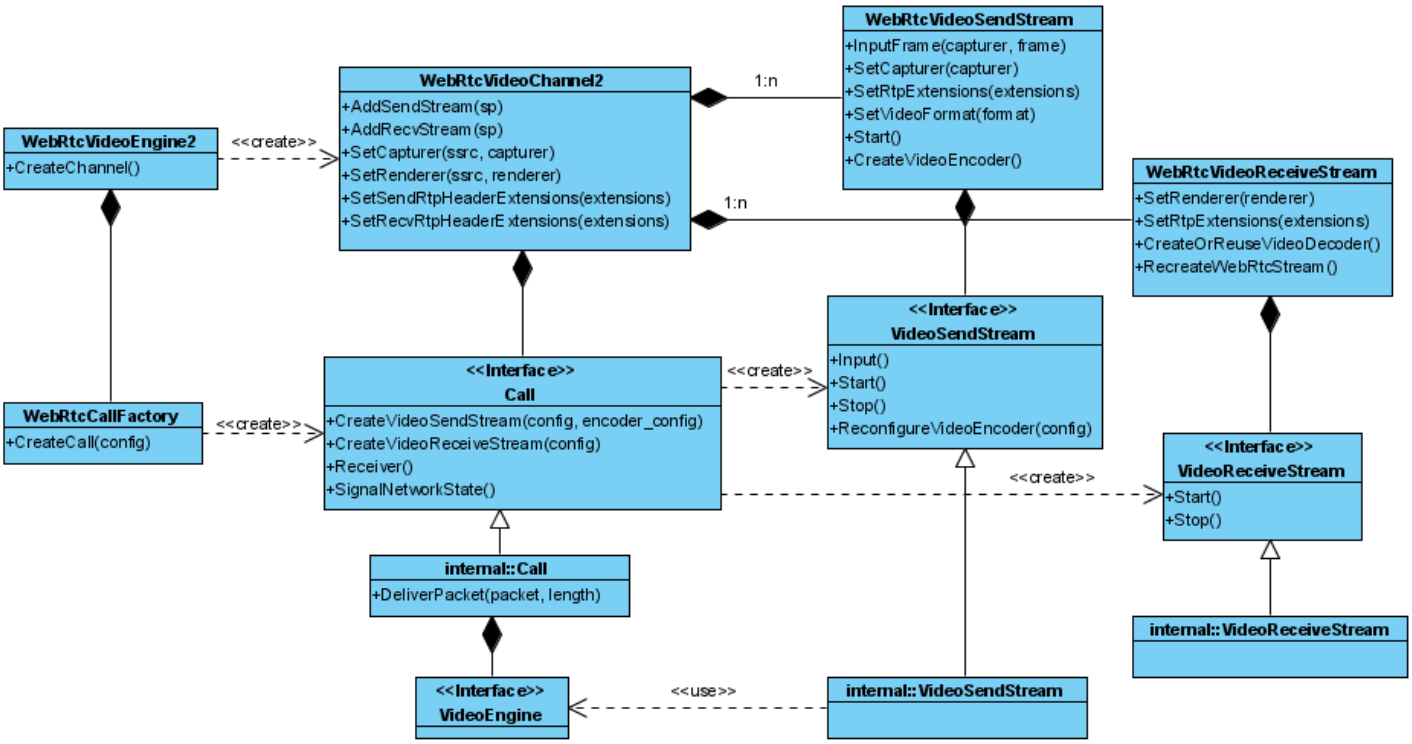
终于讲到视频数据的编码发送模块了，不容易。总体来说也看了不少时间WebRTC的源码了，最大的感触就是各个模块在开发的时候非常独立，每个模块都定义了自己的一套接口，最后串起来的时候添加各种适配对象来转接。这给我们这些刚开始源码阅读的人带来非常大的苦恼，不过WebRTC的模块内的结构设计还是很不错的，不然我也没有看下去的动力。

注意命名，WebRtcVideoEngine2带了个2字，不用想，这肯定是个升级版本的VideoEngine，还有个WebRtcVideoEngine类。从目前我的理解来看，WebRtcVideoEngine2比WebRtcVideoEngine改进之处在于将视频流一分为二：发送流（WebRtcVideoSendStream）和接收流（WebRtcVideoReceiveStream），从而结构上更合理，源码更清晰。这个部分等下会细说。在介绍WebRtcVideoEngine2之前，先简单地分析一下WebRTC的Media Engine结构，说实话，我真不会表达Engine是个怎样的概念，但既然这样命名，核心模块肯定是错不了的。结构很简单：

- **MediaEngineInterface**：抽象Media Engine的逻辑接口，负责创建用于视频传输的VideoMediaChannel、用于音频传输的VoiceMediaChannel、注册音频数据处理接口等。
- **CompositeMediaEngine**：实现MediaEngineInterface接口，本身也是个模板类，两个模板参数分别是视频Engine和音频Engine。其派生类WebRtcMediaEngine依赖的模板参数是WebRtcVoiceEngine和WebRtcVideoEngine，而用于Chromium的WebRtcMediaEngine2则依赖WebRtcVoiceEngine和WebRtcVideoEngine2。




WebRtcVideoEngine2主要作用在于创建视频channel对象WebRtcVideoChannel2。结构如下：

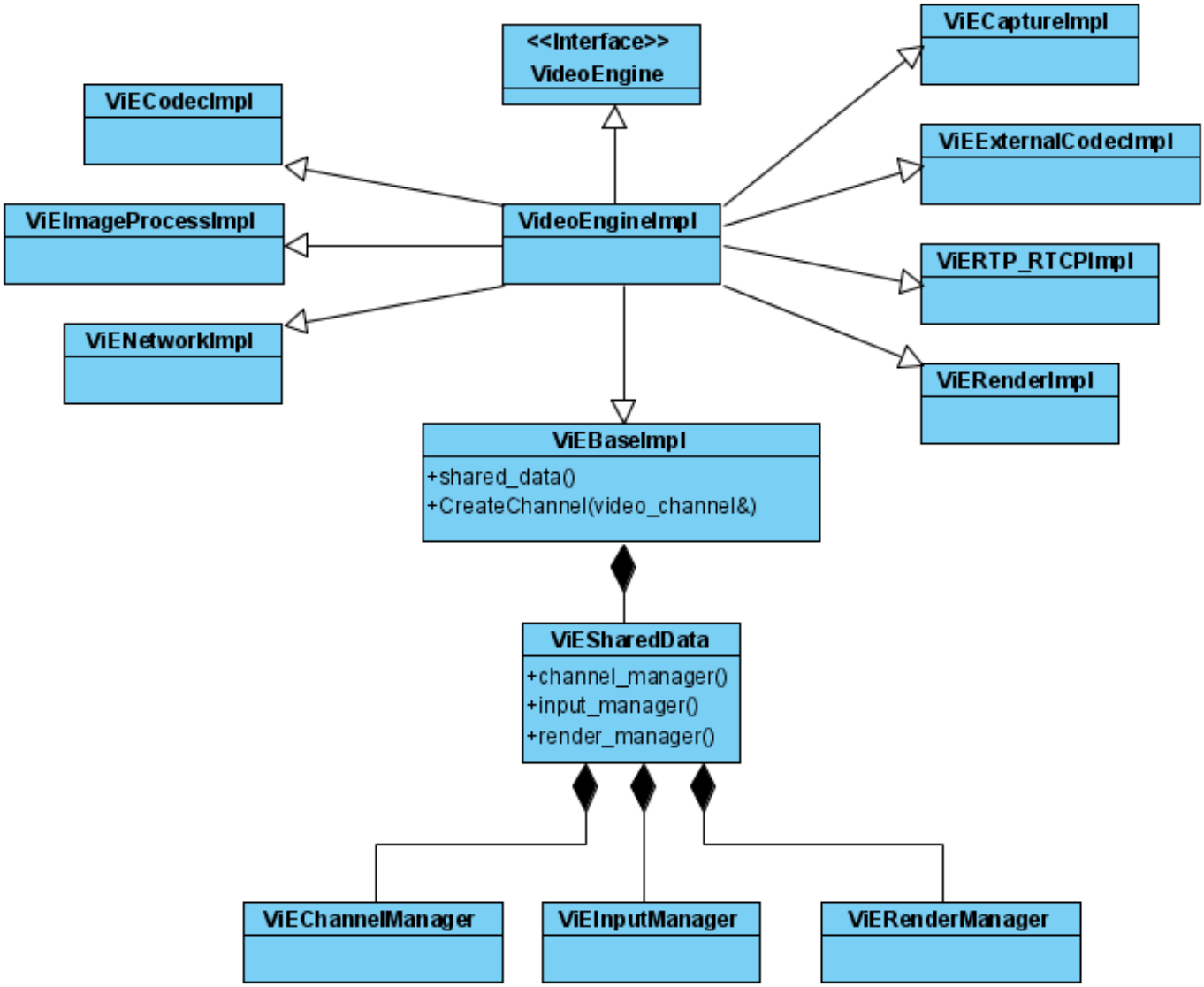



当调用WebRtcVideoChannel2的AddSendStream方法时，会创建一个WebRtcVideoSendStream对象，同样，调用AddRecvStream成员方法，会创建一个WebRtcVideoReceiveStream对象。

当外部调用WebRtcVideoChannel2的SetCapturer方法时，会转给WebRtcVideoSendStream来响应，WebRtcVideoSendStream内部将InputFrame成员方法挂接VideoCapturer的SignalVideoFrame信号来接收视频采集器传输过来的视频帧数据。

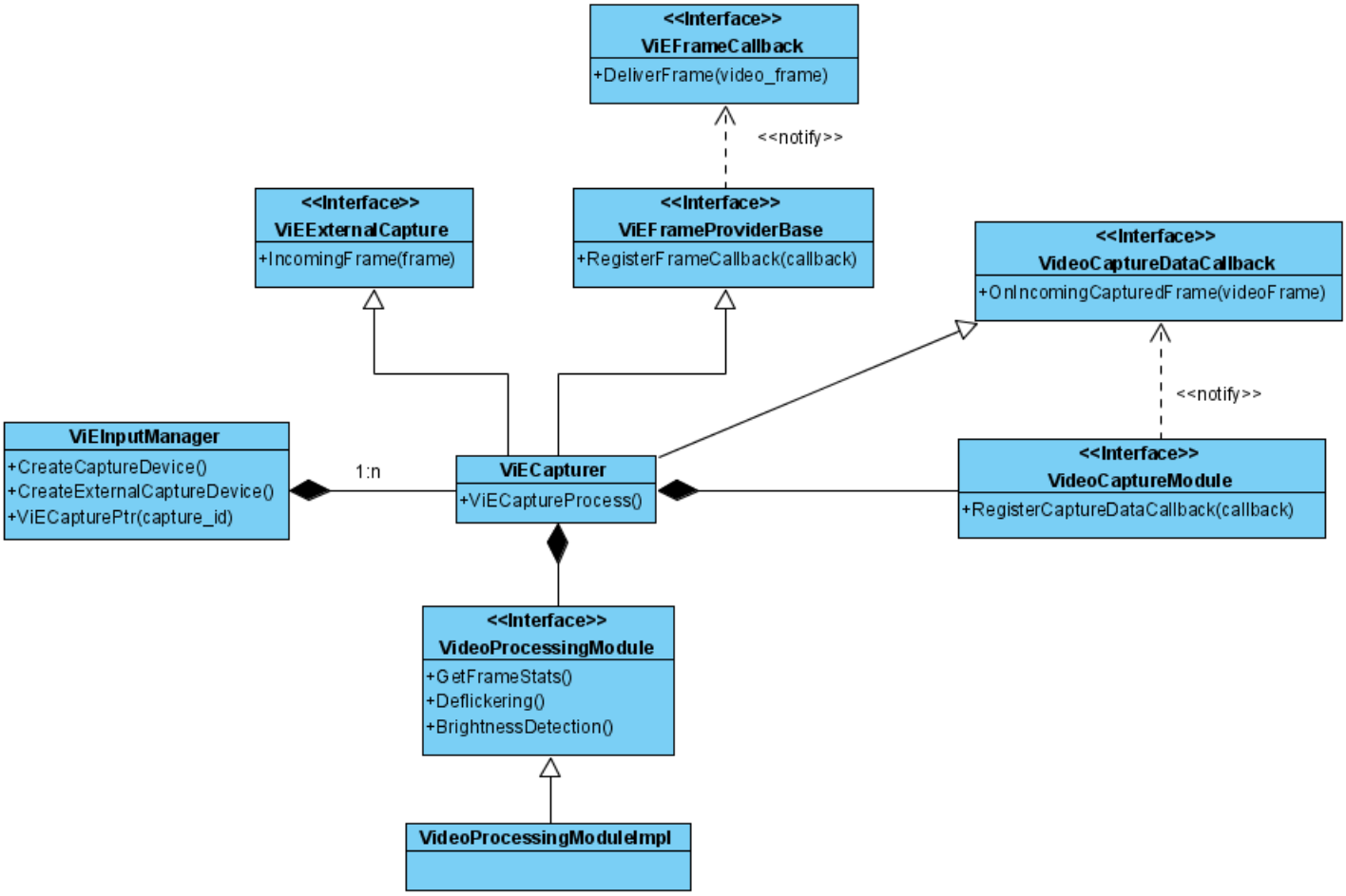
WebRtcVideoChannel2的AddSendStream和SetCapturer的调用时机这里暂时不考虑，这些涉及到网络连接，等每个节点的内容分析完后，再探讨整个流程。

如图所示，WebRtcVideoSendStream和WebRtcVideoReceiveStream也只是个包装类，内部依赖Call接口创建对应的VideoSendStream接口实现类和VideoReceiveStream接口实现类。在internal命名空间内，分别有一个Call类、VideoSendStream类、VideoReceiveStream类来实现这三个接口，Call类创建关键的VideoEngine对象来管理视频数据发送过程中的一系列处理逻辑。从代码结构上看，VideoEngine是一个相对独立的模块，它封装视频数据采集后的处理、编码等逻辑，下面仔细分析一下VideoEngine的结构：



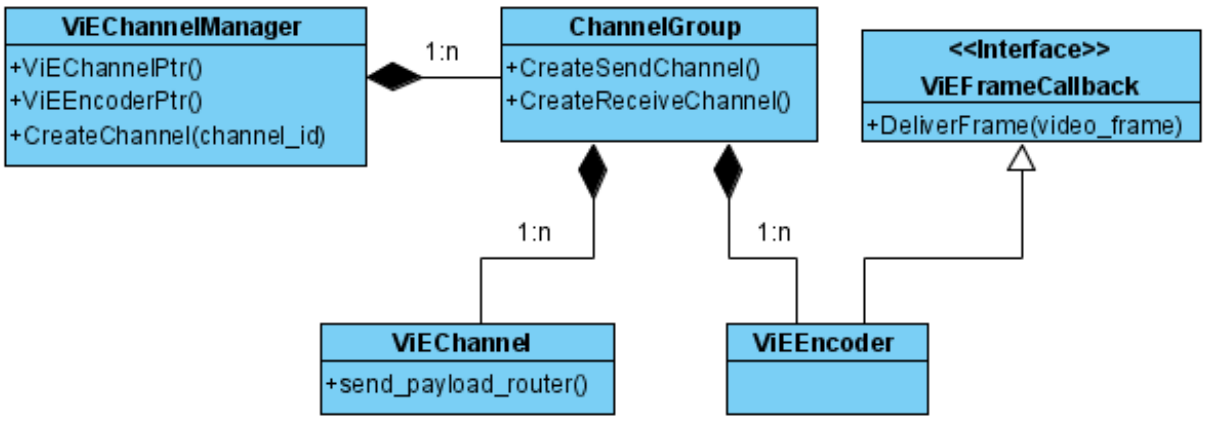

VideoEngine模块里有ViEBase、ViECodec、ViECapture、ViEImageProcess、ViENetwork、ViERender、ViERTP_RTCP、ViEExternalCodec接口，注意，这些都是功能性的接口，它们相应的实现分别对应于上图中的XXXImpl类，VideoEngineImpl类从所有的XXXImpl接口派生，因此外部有了VideoEngine接口，都可以通过强转的方式获取ViEBase、ViECapture等之类的接口（根据VideoEngine强转成相应的接口的逻辑封装在目标接口的GetInterface静态方法中），外界可以通过这些接口来完成视频数据做相应的设置，而这些设置最终都反映到一个名叫ViESharedData的类对象里。该对象由ViEBaseImpl创建并在各接口的实现之间共享，XXXImpl可以通过ViEBaseImpl的shared_data方法来访问，用于共享的数据有三类：ViEInputManager、ViEChannelManager和ViERenderManager。下面分别介绍一下这关键的三个对象。

- **ViEInputManager**：封装了视频采集/输入逻辑（哈哈，又是一套视频输入逻辑），结构：



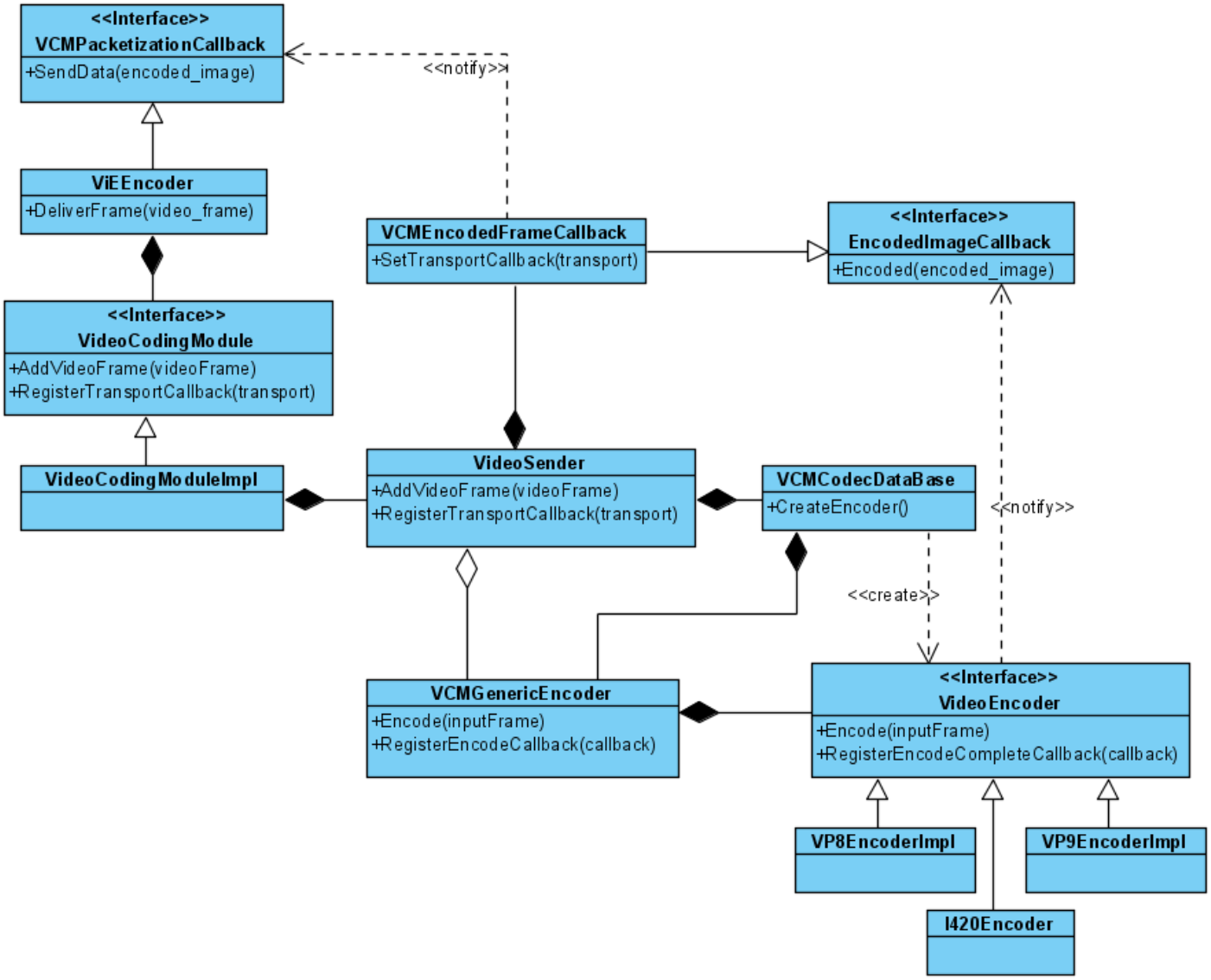

ViEInputManager为每个通道分配一个ViECapturer对象来做为视频源，ViECapturer可以传入也可以自己创建一个VideoCaptureModule视频采集模块，并通过VideoCaptureDataCallback接口从其接收数据，也可以直接通过ViEExternalCapture接口接收从外部直接传入的视频帧数据（调用ViEExternalCapture接口的IncomingFrame方法）。VideoSendStream就是通过ViEInputManager创建一个ViEExternalCapture对象来传入外界传来的视频帧数据（通过WebRtcVideoSendStream的InputFrame传来）。这里要注意，ViEInputManager为创建的ViECapturer对象分配一个capture_id，外界可以通过这个capture_id来操作其对应的ViECapturer。视频源传入逻辑已经明了，接下来分析一下视频是怎么传出去的。无论通过哪种视频数据接收方法，ViECapturer都不会立即将数据传递出去，因为它内部需要对这些视频数据做相关的处理。数据处理必然耗时，如果采用同步的方式，必将阻塞视频传入的流程。因此，在创建ViECapturer的时候，会启动一采集线程，该线程调用ViECaptureProcess处理函数，在该处理函数里，先调用VideoProcessingModule对视频数据进行处理（灯光加亮、去闪烁），如果在ViEImageProcessImpl里注册了ViEEffectFilter处理对象，这里也会调用该对象来处理视频帧数据，最后通过DeliverFrame方法分发到注册进来的所有ViEFrameCallback接口。

- **ViEChannelManager**：封装了视频编码和传输逻辑，这块结构比较复杂，总体如下：



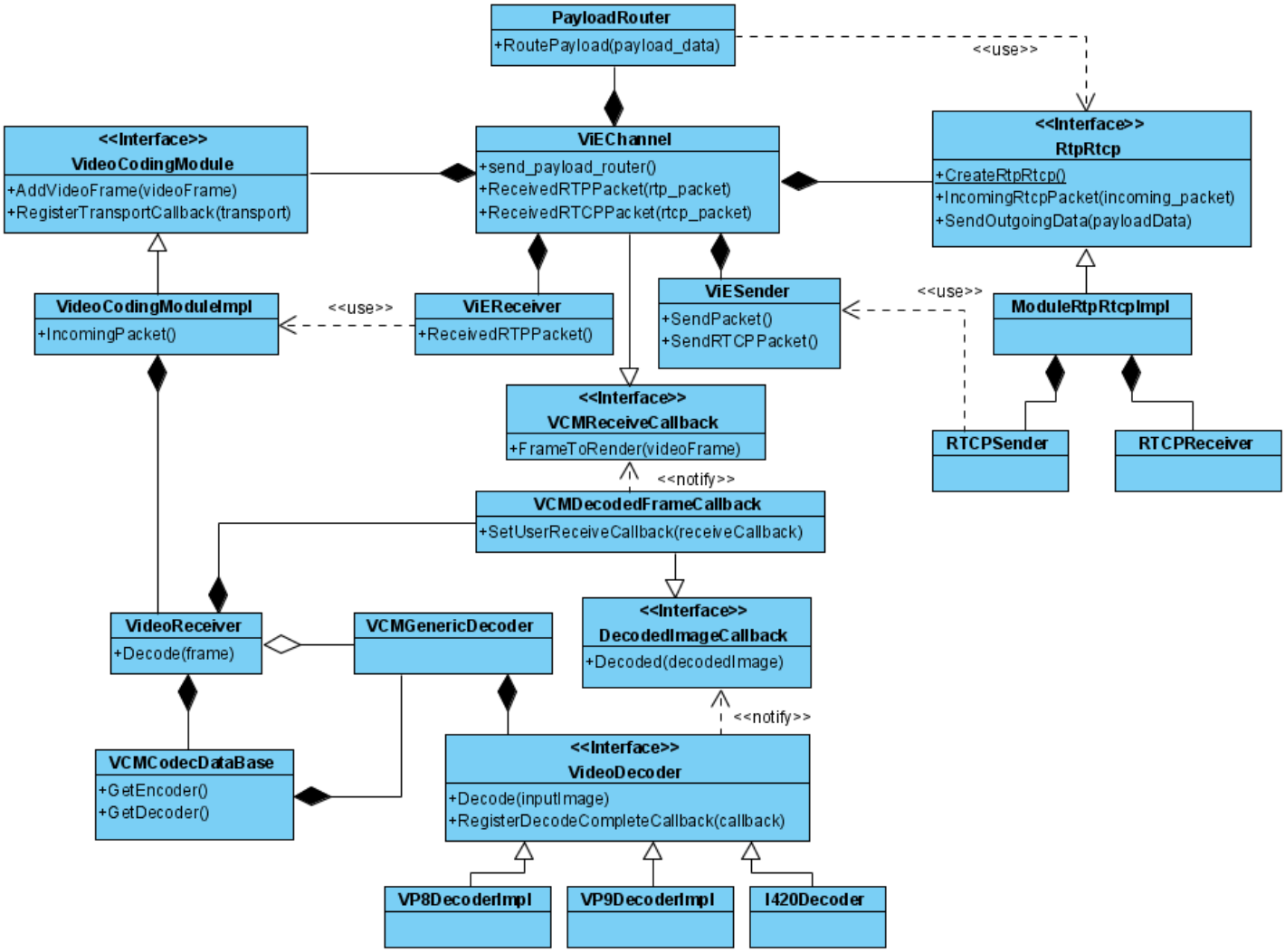
ViEChannelManager维护了ViEEncoder和ViEChannel对象，ViEEncoder实现了ViEFrameCallback接口从ViECapturer对象中接收视频帧数据，ViEEncoder对接收到的视频帧数据进行编码，然后将编码后的数据传给ViEChannel（通过两者之间共享的PayloadRouter对象），ViEChannel将编码后的视频数据通过RTP/RTCP协议发送出去。下面分别分析一下ViEEncoder和ViEChannel。

**1） ViEEncoder类**：封装了视频编码流程。




视频编码由VideoCodingModule模块统一管理，视频帧传入接口是通过VideoCodingModule的的AddVideoFrame方法，编码后的视频传出接口是借助VCMPacketizationCallback接口来回调。具体选取哪种视频编码的逻辑位于VCMCodecDataBase类，当前支持VP8编码、VP9编码和视频格式到I420格式的转换。

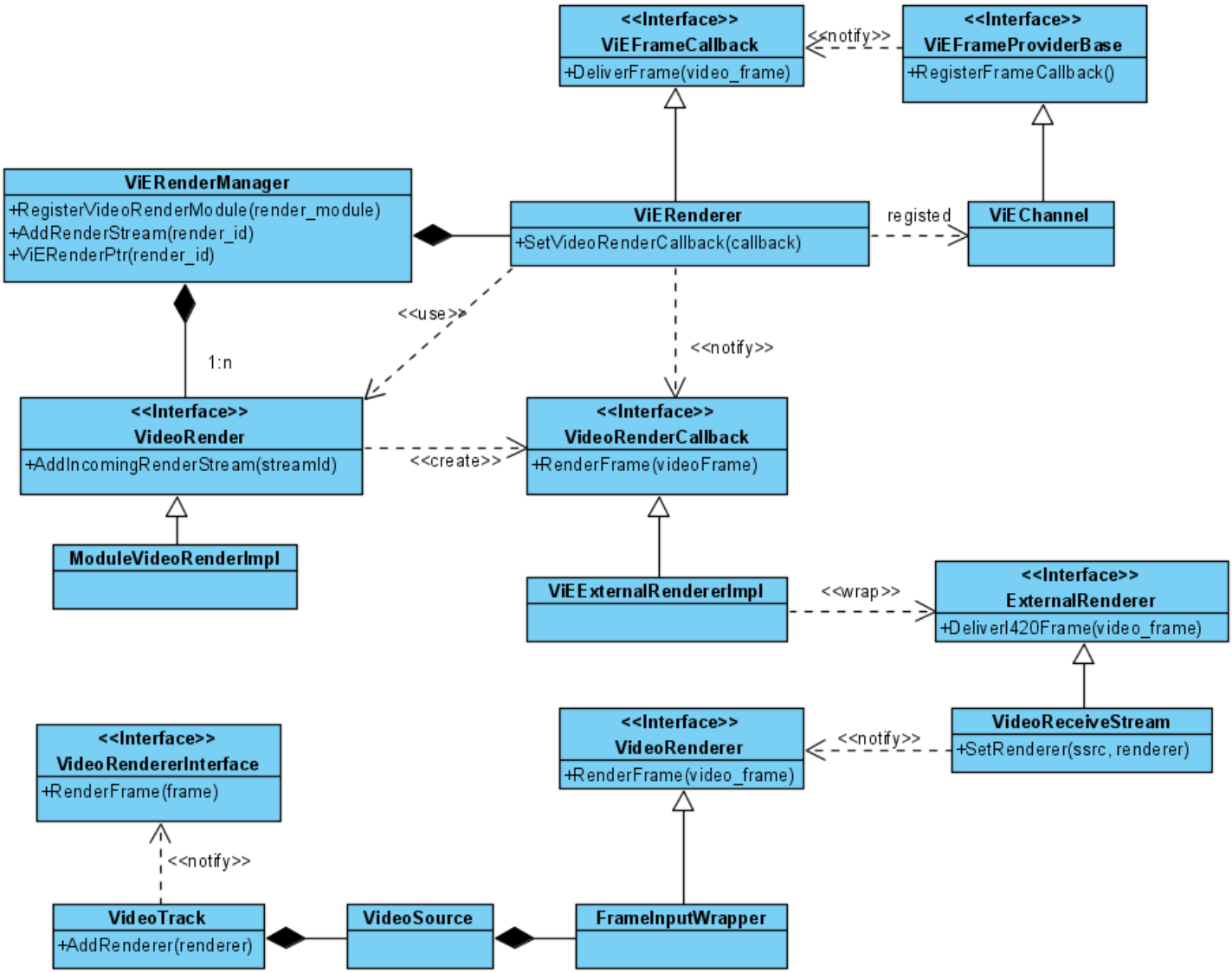
**2）ViEChannel类**：封装了编码后的视频数据发送逻辑和视频数据接收解码逻辑。




视频数据发送逻辑是通过PayloadRouter对象委托给RtpRtcp模块做RTP协议的封装，具体的网络发送操作还是回托给ViESender做数据的网络发送操作。ViESender的逻辑相对简单，限于篇幅，图中无法做详细的标注。ViESender的发送操作依赖外部设置给它的Transport接口（通过VideoEngine模块的ViENetwork接口来完成设置）。

当WebRtcVideoChannel2接收到网路数据包后（通过OnPacketReceived或OnRtcpReceived方法响应），会在VideoReceiveStream对象中通过VideoEngine模块暴露出去的ViENetwork接口来响应数据包处理，最终触发到ViEChannel的ReceivedRTPPacket或ReceivedRTCPPacket方法。ViEChannel中将接收并解码网络视频数据的任务分配给ViEReceiver对象。ViEReceiver先调用RTP/RTCP模块做协议的解析（图中限于篇幅未标注出来），解析完成后调用VideoCodingModule模块进行数据的解码操作（参见ViEReceiver的OnReceivedPayloadData方法），VideoCodingModule模块内部维护了一个与VideoSender对应的VideoReceiver来完成解码逻辑，这块与VideoSender的编码逻辑完全对称，这里不再表述。

- **ViERenderManager**：这个类封装了视频渲染逻辑，结构如下：




当ViEChannel接收到网络数据解包并解码后，就会开启触发渲染流程（参见FrameToRender方法），ViEChannel会调用向其注册的ViEFrameCallback接口来派发视频帧数据。ViERenderManager维护了一个ViERenderer对象来实现ViEFrameCallback接口，它将数据进一步派发，最终通过ExternalRenderer接口派发给WebRtcVideoChannel2的VideoReceiveStream对象。VideoReceiveStream通过VideoSource设置进来的VideoRenderer接口将数据派发给VideoTrack，用户可以挂接VideoRendererInterface接口来接收视频帧数据。真够绕的，而且那么多命名的相似性（比如VideoRender/VideoRenderer），感觉各模块开发期间，都实现了自己的一套接口规范，最后强行串在一起了。