webrtc--AudioProcessing的使用

1.AudioProcessing的实例化和配置：

AudioProcessing* apm = AudioProcessing::Create(0);

apm->level_estimator()->Enable(true)；//启用重试次数估计组件

apm->echo_cancellation()->Enable(true);//启用回声消除组件

apm->echo_cancellation()->enable_metrics(true);//

apm->echo_cancellation()->enable_drift_compensation(true);//启用时钟补偿模块（声音捕捉设备的时钟频率和播放设备的时钟频率可能不一样）

apm->gain_control()->Enable(true);//启用增益控制组件，client必须启用哦！

apm->high_pass_filter()->Enable(true);//高通过滤器组件，过滤DC偏移和低频噪音，client必须启用

apm->noise_suppression()->Enable(true);//噪声抑制组件，client必须启用

apm->voice_detection()->Enable(true);//启用语音检测组件，检测是否有说话声

apm->voice_detection()->set_likelihood(VoiceDetection::kModerateLikelihood);//设置语音检测的阀值，阀值越大，语音越不容易被忽略，同样一些噪音可能被当成语音。

apm->Initialize();//保留所有用户设置的情况下重新初始化apm的内部状态，用于开始处理一个新的音频流。第一个流创建之后不一定需要调用此方法。

2.AudioProcessing的工作流程：

AudioProcessing也是事件驱动的，事件分为初始化事件、捕捉音频事件、渲染音频事件。

初始化事件：

apm->set_sample_rate_hz(sample_rate_hz)；//设置本地和远程音

10. CentOS6.5 一键安装vpn + 添加账号

随笔分类(398)

ADO.NET(1)
Android(12)
AOP(1)
ASP.NET(6)
C Programming(7)
C#.Net(6)
C++(16)
CodeSmith(1)
DIV CSS(41)
EnterPrise Library(1)
ERP分类(1)
Flex(4)
FreeBSD(5)
html5(1)
ipad
iphone
JAVA(10)
JS(30)
Linux(44)
lua(2)
MySQL(21)
NOSQL(1)
Oracle(2)
P2P(1)
Perl(1)
PHP(63)
PostgreSQL(31)
Project Management(1)
SHELL(7)
SQL(13)
UML(3)
Unity3D(4)
webrtc(4)
WebService(2)
xcode(2)
大型系统架构(9)
翻译分类
管理,营销(5)
加密解密(5)
架构设计(5)
软件测试(7)
软件开发(1)
设计模式
数据结构与算法(1)
水晶报表(1)
搜索引擎(5)
网页技术(2)
项目管理(12)

频流的采样率

apm->echo_cancellation()->set_device_sample_rate_hz(); //设置音频设备的采样率，我们假定音频采集和播放设备采用同样的采样率。（drift组件启用时必须调用）

apm->set_num_channels(num_capture_input_channels, num_capture_output_channels);//设置本地和远程音频流的通道数

播放事件：

apm->AnalyzeReverseStream(&far_frame));//分析远端音频流的10ms的frame数据，这些数据为回声抑制提供参考。（启用回声抑制的时候需要调用）

捕捉事件：

apm->gain_control()->set_stream_analog_level(capture_level);

apm->set_stream_delay_ms(delay_ms + extra_delay_ms);//设置本地和远端音频流之间的延迟，单位毫秒。这个延迟是远端音频流和本地音频流之间的时差，计算方法为：

delay = (t_render - t_analyze) + (t_process - t_capture);

其中

t_analyze是远端音频流交给AnalyzeReverseStream（）方法的时间；

t_render是与刚才同样的远端音频frame的播放时间；

t_capture是本地音频frame捕捉的时间；

t_process是同样的本地音频frame被交给ProcessStream（）方法的时间。

apm->echo_cancellation()->set_stream_drift_samples(drift_samples);//设置音频设备捕捉和播放的采样率的差值。（drift组件启用时必须调用）

int err = apm->ProcessStream(&near_frame);//处理音频流，包括各个环节的处理。（如增益调节、回声消除、噪声抑制、语音检测、高通过率等，没有解码哦！是针对pcm数据做处理的）

capture_level = apm->gain_control()->stream_analog_level();//模拟模式下，必须在ProcessStream之后调用此方法，获取新的音频HAL的推荐模拟值。

stream_has_voice =apm->voice_detection()->stream_has_voice();//检测是否有语音，必须在ProcessStream之后调用此方法

ns_speech_prob = apm->noise_suppression()->speech_probability();//返回内部计算出的当前frame的人声优先概率。