gRPC Wrapper v1.0

gRPC Wrapper is a library plugin that provides gRPC C++ API in Unreal Engine environment.

Current supported target build platform: **Win64**

gRPC library build version v1.51.1
https://github.com/grpc/grpc/tree/v1.51.1

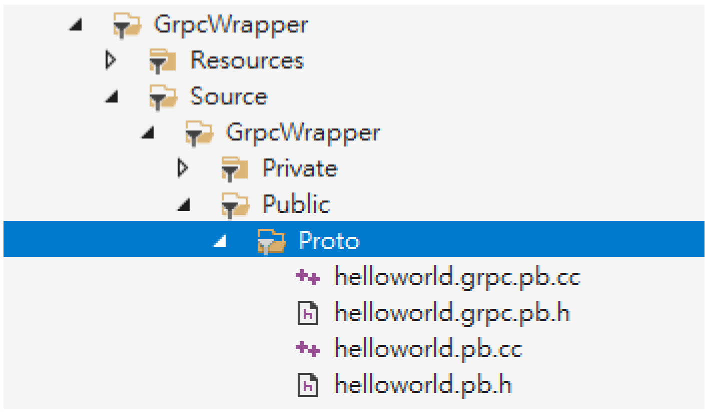
grpc-wrapper-sample: You can find latest example code here
https://github.com/WandererDev1988/grpc-wrapper-sample

Usage:

After install plugin, gRPC headers and libraries are placed here:
`/Engine/Plugins/Marketplace/GrpcWrapper/Source/ThirdParty/grpc/include/`
`/Engine/Plugins/Marketplace/GrpcWrapper/Source/ThirdParty/grpc/lib/`

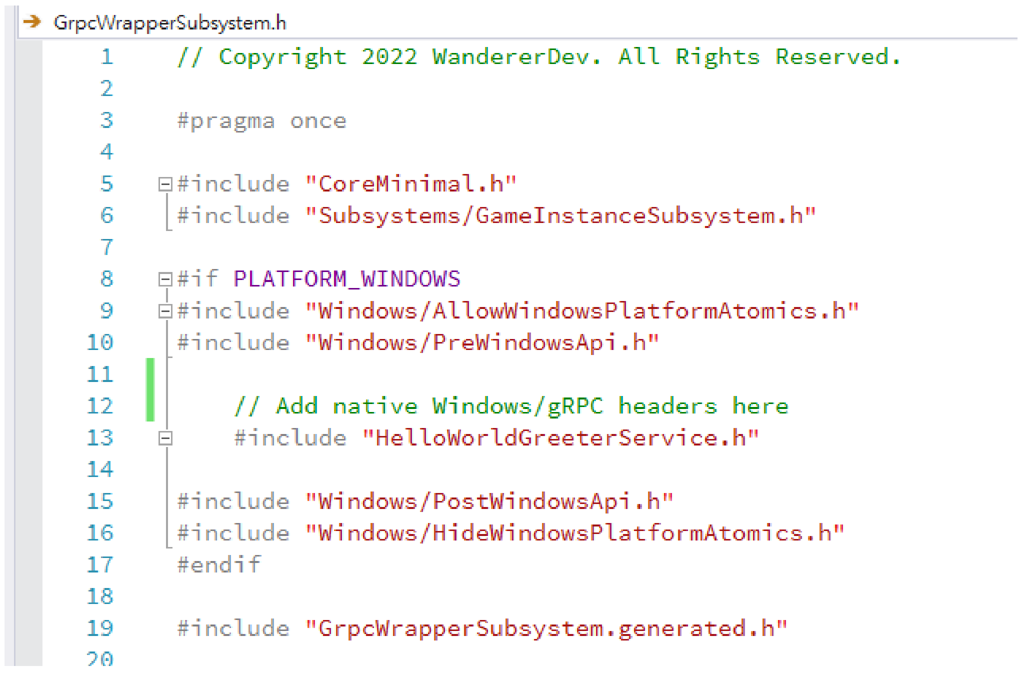
Grpc.Build.cs has added these build paths, therefore you can include gRPC headers in your code.

After defined gRPC service *.proto file, you can generate cpp files by protoc.exe tool, and place all `*.pb.h *.pb.cc` in `Source` directory



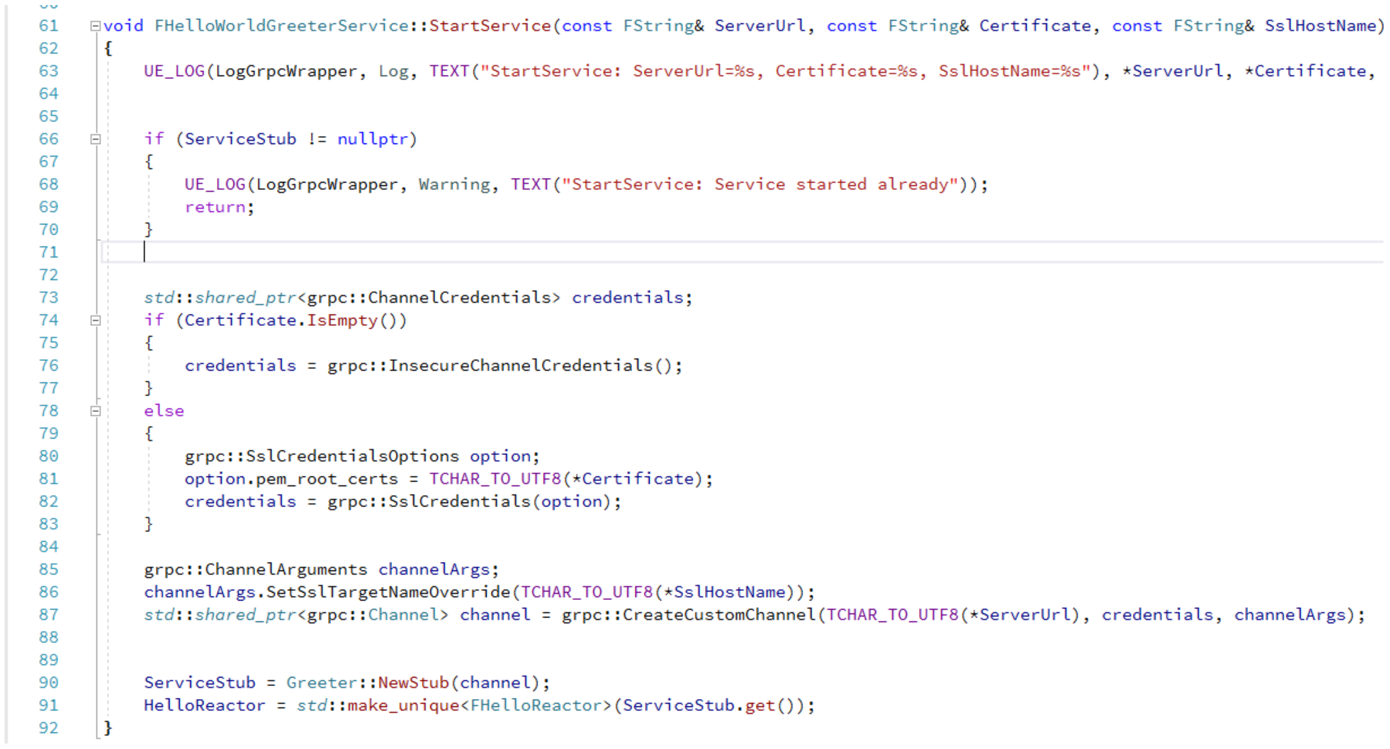
```
GrpcWrapper
  Resources
  Source
    GrpcWrapper
      Private
      Public
        Proto
          helloworld.grpc.pb.cc
          helloworld.grpc.pb.h
          helloworld.pb.cc
          helloworld.pb.h
```

The following pic shows implementation, make sure to add UE redirect headers around any native Windows/gRPC headers.


GrpcWrapperSubsystem.h

```cpp
// Copyright 2022 WandererDev. All Rights Reserved.

#pragma once

#include "CoreMinimal.h"
#include "Subsystems/GameInstanceSubsystem.h"

#if PLATFORM_WINDOWS
#include "Windows/AllowWindowsPlatformAtomics.h"
#include "Windows/PreWindowsApi.h"

    // Add native Windows/gRPC headers here
    #include "HelloWorldGreeterService.h"

#include "Windows/PostWindowsApi.h"
#include "Windows/HideWindowsPlatformAtomics.h"
#endif

#include "GrpcWrapperSubsystem.generated.h"

```

Then use gRPC native API as usual, you can mix Unreal and std cpp in your code.



```cpp
void FHelloWorldGreeterService::StartService(const FString& ServerUrl, const FString& Certificate, const FString& SslHostName)
{
    UE_LOG(LogGrpcWrapper, Log, TEXT("StartService: ServerUrl=%s, Certificate=%s, SslHostName=%s"), *ServerUrl, *Certificate,


    if (ServiceStub != nullptr)
    {
        UE_LOG(LogGrpcWrapper, Warning, TEXT("StartService: Service started already"));
        return;
    }


    std::shared_ptr<grpc::ChannelCredentials> credentials;
    if (Certificate.IsEmpty())
    {
        credentials = grpc::InsecureChannelCredentials();
    }
    else
    {
        grpc::SslCredentialsOptions option;
        option.pem_root_certs = TCHAR_TO_UTF8(*Certificate);
        credentials = grpc::SslCredentials(option);
    }

    grpc::ChannelArguments channelArgs;
    channelArgs.SetSslTargetNameOverride(TCHAR_TO_UTF8(*SslHostName));
    std::shared_ptr<grpc::Channel> channel = grpc::CreateCustomChannel(TCHAR_TO_UTF8(*ServerUrl), credentials, channelArgs);


    ServiceStub = Greeter::NewStub(channel);
    HelloReactor = std::make_unique<FHelloReactor>(ServiceStub.get());
}
```