Getting Started with TeamSpeak 3 SDK and Unreal Engine

Overview

This package will allow you to integrate the TeamSpeak VoIP solution into an Unreal Engine 4 project. The TeamSpeak API is a pure C API to allow integration into many different languages. This package includes a wrapper to connect the TeamSpeak C API with Unreal Engine blueprints. In addition, it is possible to build your integration with C++ by using the TeamSpeak API directly.

Currently the project supports Unreal Engine 4 on Windows.

How the wrapper works

The wrapper offers two methods of using the TeamSpeak SDK.

1. Blueprints
   Most TeamSpeak functions (see note below) are completely blueprintable. The TeamSpeak API is directly mapped to blueprints. With blueprints you can easily implement TeamSpeak functionality without programming in C++.

   Note: The TeamSpeak SDK uses 64 bit integers for handlers and channelid. Unfortunately, the integers in Blueprint are signed 32-bit integers. If you require larger numbers than that, please use 64-bit integers in the C++ interface.

2. Access the TeamSpeak C++ wrapper functions
   The idea behind the C++ wrapper is to simplify development by using the Unreal Engine data types directly, avoiding writing an own wrapper between the TeamSpeak API and Unreal Engine types.
   The wrapper will also automate take care of allocating and deallocating the memory of the native TeamSpeak API structures.

Alternatively, the TeamSpeak API can be accessed directly. However, this will require you to convert the C structures into Unreal data types. Instead, we recommend using the C++ wrapper functions.

How to use the TeamSpeak SDK

The TeamSpeak SDK is delivered as a module.

1. Extract the “Source” and “ThirdParty” folder into your existing project.
2. Add the module into your project:
   - YourProject.Target.cs and YourProjectEditor.Target.cs:
     Add to TargetRules: OutExtraModuleNames.Add(“TeamSpeak_SDK”);
   - Add the module name “TeamSpeak_SDK” to PublicDependencyModuleNames in YourProject.build.cs of your primary module.
3. Make sure you added the required ts3client_win64.dll / ts3client_win32.dll in your dependencies or put it into your Binaries subfolders. Required to launch your game or editor.

For a sample setup, see the included project files in Template\TeamSpeak SDK sample\Source

To access the blueprint functionality, add a class member variable with TeamSpeak_Manager type in your GameMode and initialize it. It's not possible to create the variable in a blueprinted GameMode class, C++ source code is required. For C++ usage you can choose the class you want. Do not forget to include “TeamSpeak_Manager.h”.

For a quick introduction, see the sample in the Template folder. This sample demonstrates how to build a simple TeamSpeak client with blueprints. To access the sample, either open the Template’s .uproject file directly or copy the “TeamSpeak SDK sample” folder into your Unreal Engine Templates directory, which adds a new project type “TeamSpeak SDK” to the engines New Project dialog.

The TeamSpeak SDK wrapper for Unreal Engine only includes the TeamSpeak client. This client is able to connect to a TeamSpeak SDK server (connecting to regular non-SDK servers is not supported).

For a quick-start, the template includes a ready-to-run SDK server executable. To customize the SDK server, download the full TeamSpeak SDK from www.teamspeak.com/?page=downloads.

When deploying or testing Unreal engine applications with TeamSpeak, make sure to include the following DLLs:

1. ts3client_win32.dll or ts3client_win64.dll (directly linked)
2. soundbackends (loaded dynamically at runtime)
    - directsound_win32.dll or directsound_win64.dll
    - windowsaudiosession_win32.dll or windowsaudiosession_win64.dll

When creating a new TeamSpeak SDK project, the files are automatically included in your project.

Additional Information:

For information how to use the TeamSpeak SDK or to see more samples, see the TeamSpeak SDK Documentation (client) and SDK sample.

It's not possible to get connection information (client and server) through the blueprint interface. Those information are uint64 values only. If you need those values, access it through the C++ interface.