Setup

This guide will demonstrate the setup of the Server Admin plugin and shows you where in your game's code it needs to be integrated to make full use of its functionality. We are going to be using the ShooterGame code as an example. So, wherever a ShooterGame class is changed in this guide, make these same changes to your corresponding game classes.

Important note: This plugin is made for a server-client environment. Make sure to tick the "Run Dedicated Server" checkbox when testing the functionality in the editor!

UPROJECT

After installing it to your chosen engine version, it is available to the engine but needs to be referenced by your project still. Open the ShooterGame's .uproject file and add the following lines at the bottom of the plugins list, right below the replication graph:

```
67          {
68              "Name": "ReplicationGraph",
69              "Enabled": true
70          },
71          {
72              "Name": "ServerAdmin",
73              "Enabled": true
74          }
```

PREPARING YOUR GAME CLASSES

SHOOTERGAMEGAMEMODE.H

Include the ServerAdminGameMode header in line #5 and then change the class that your gamemode inherits from to **AServerAdminGameMode** in what is now line #15.

```
5   #include "OnlineIdentityInterface.h"
6   #include "ServerAdminGameMode.h"
7   #include "ShooterGameMode.generated.h"
8
9   class AShooterAIController;
10  class AShooterPlayerState;
11  class AShooterPickup;
12  class FUniqueNetId;
13  class APlayerStart;
14
15  UCLASS(config=Game)
16  class AShooterGameMode : public AServerAdminGameMode
17  {
18      GENERATED_UCLASS_BODY()
```

SHOOTERGAMEGAMEINSTANCE.H

Replace the header line #8 and then replace the class in line 96 to read **UServerAdminGameInstance**

```
1   // Copyright 1998-2019 Epic Games, Inc. All Rights Reserved.
2
3   #pragma once
4
5   #include "ShooterGame.h"
6   #include "OnlineIdentityInterface.h"
7   #include "OnlineSessionInterface.h"
8   #include "ServerAdminGameInstance.h"
9   #include "Engine/NetworkDelegates.h"
10  #include "ShooterGameInstance.generated.h"
```

```
95  UCLASS(config=Game)
96  class UShooterGameInstance : public UServerAdminGameInstance
97  {
98  public:
99      GENERATED_UCLASS_BODY()
```

SHOOTERGAMEPLAYERCONTROLLER.H

In the includes part of the **ShooterGameController** header file include the **ServerAdminPlayerController** header and replace the class in line #13.

```
3   #pragma once
4
5   #include "Online.h"
6   #include "ShooterLeaderboards.h"
7   #include "ServerAdminPlayerController.h"
8   #include "ShooterPlayerController.generated.h"
9
10  class AShooterHUD;
11
12  UCLASS(config=Game)
13  class AShooterPlayerController : public AServerAdminPlayerController
14  {
15      GENERATED_UCLASS_BODY()
```

SHOOTERGAMESTATE.H

In the includes of the **ShooterGameState** header file include the **ServerAdminGameState** header and replace the class in line #12.

```
3   #pragma once
4
5   #include "ServerAdminGameState.h"
6   #include "ShooterGameState.generated.h"
7
8   /** ranked PlayerState map, created from the GameState */
9   typedef TMap<int32, TWeakObjectPtr<AShooterPlayerState> > RankedPlayerMap;
10
11  UCLASS()
12  class AShooterGameState : public AServerAdminGameState
13  {
14      GENERATED_UCLASS_BODY()
```

SHOOTERGAMESESSION.H

In the ShooterGameSession header, include ServerAdminGameSession.h as seen on line 7 and the class declaration on line 33.

```
7   #include "ServerAdminGameSession.h"
```

```
31
32  UCLASS(config=Game)
33  class SHOOTERGAME_API AShooterGameSession : public AServerAdminGameSession
34  {
35      GENERATED_UCLASS_BODY()
36
```

SHOOTERGAME.BUILD.CS

In the ShooterGame build file in line 32, add the following line at the bottom of PublicDependencyModuleNames: "ServerAdmin".

```
21          PublicDependencyModuleNames.AddRange(
22              new string[] {
23                  "Core",
24                  "CoreUObject",
25                  "Engine",
26                  "OnlineSubsystem",
27                  "OnlineSubsystemUtils",
28                  "AssetRegistry",
29                  "NavigationSystem",
30                  "AIModule",
31                  "GameplayTasks",
32                  "ServerAdmin",
33              }
34          );
35
```

GAMESTATE VIRTUAL FUNCTIONS

Some features of this plugin provide you with virtual functions that need to be overridden in your game's GameState class in either C++ or BP. The reason for that is that they have to modify properties of your GameState class that are declared there, i.e. the ServerAdminGameState cannot access them, if you choose to inherit from that class.

In the ShooterGame GameState this will look like this:

```
84
85  ////////////////////////////////////////////////////////////////////////////
86  // Server Admin
87  ////////////////////////////////////////////////////////////////////////////
88
89  void AShooterGameState::SetRemainingTime(float NewRemainingTime)
90  {
91      RemainingTime = NewRemainingTime;
92  }
93
94  float AShooterGameState::GetRemainingTime()
95  {
96      return RemainingTime;
97  }
98
99  float AShooterGameState::GetRoundTime()
100 {
101     const AShooterGameMode* MyGameMode = GetDefaultGameMode<AShooterGameMode>();
102
103     if (MyGameMode)
104     {
105         return MyGameMode->GetRoundTime();
106     }
107     else
108         return 0.0f;
109 }
110
```

And the header file

```
39      ////////////////////////////////////////////////////////////////////////////
40      // Server Admin
41      ////////////////////////////////////////////////////////////////////////////
42      void SetRemainingTime(float NewRemainingTime);
43
44      virtual float GetRemainingTime() override;
45
46      virtual float GetRoundTime() override;
47  };
48
```

This will allow the Admin Menu to access members of your GameState class despite not being able to directly access them.

CHANGES TO GAMEMODE

The Shooter GameMode DefaultTimer, which handles how the game progresses through its states, will restart the game at the end of each round's WaitingPostMatch state. What we want instead is for it to call the plugin function that figures out the next map. So, change the RestartGame(); call with DetermineNextMap();

```
87      AShooterGameState* const MyGameState = Cast<AShooterGameState>(GameState);
88      if (MyGameState && MyGameState->RemainingTime > 0 && !MyGameState->bTimerPaused)
89      {
90          MyGameState->RemainingTime--;
91
92          if (MyGameState->RemainingTime <= 0)
93          {
94              if (GetMatchState() == MatchState::WaitingPostMatch)
95              {
96                  ////////////////////////////////////////////////////////////////////////
97                  // ServerAdminGameMode::DetermineNextMap();
98                  ////////////////////////////////////////////////////////////////////////
99                  DetermineNextMap();
100             }
101             else if (GetMatchState() == MatchState::InProgress)
102             {
```

Another small addition that is needed is to override one of the ServerAdminGameMode's virtual functions: GetRoundTime(). Since ServerAdminGameMode cannot access this member of ShooterGameMode, this class has to deliver the data for the GameRules tab in the admin menu.

Declare the function in ShooterGameMode.h

```
60      virtual void DefaultTimer();
61
62      /** Get the remaining time*/
63      float GetRoundTime() const;
64
```

And add it to ShooterGameMode.cpp

```
127 float AShooterGameMode::GetRoundTime() const
128 {
129     return RoundTime;
130 }
```

In order for the TK counter to work, the AShooterGameMode::Killed function needs to be enhanced a bit as well. Modify the Killed function as you can see here. Particularly the if() between the lines 209 and 307.

```
294 void AShooterGameMode::Killed(AController* Killer, AController* KilledPlayer, APawn* KilledPawn, const UDamageType* DamageType)
295 {
296     AShooterPlayerState* KillerPlayerState = Killer ? Cast<AShooterPlayerState>(Killer->PlayerState) : NULL;
297     AShooterPlayerState* VictimPlayerState = KilledPlayer ? Cast<AShooterPlayerState>(KilledPlayer->PlayerState) : NULL;
298
299     if (KillerPlayerState && VictimPlayerState  && KillerPlayerState->GetTeamNum() == VictimPlayerState->GetTeamNum())
300     {
301         KillerPlayerState->ScoreKill(VictimPlayerState, KillScore);
302         KillerPlayerState->InformAboutKill(KillerPlayerState, DamageType, VictimPlayerState);
303
304         // Call the ServerAdmin parent function to increase the TK Counter
305         if(KillerPlayerState->GetTeamNum() == VictimPlayerState->GetTeamNum())
306             TeamKill(Killer, KilledPlayer, KilledPawn, DamageType);
307     }
308
    Stefan
309     if (VictimPlayerState)
310     {
311         VictimPlayerState->ScoreDeath(KillerPlayerState, DeathScore);
312         VictimPlayerState->BroadcastDeath(KillerPlayerState, DamageType, VictimPlayerState);
313     }
314 }
```

GAMEMODE.INI

Now it's time to configute the GameMode.ini to store the ServerAdmin configuration.

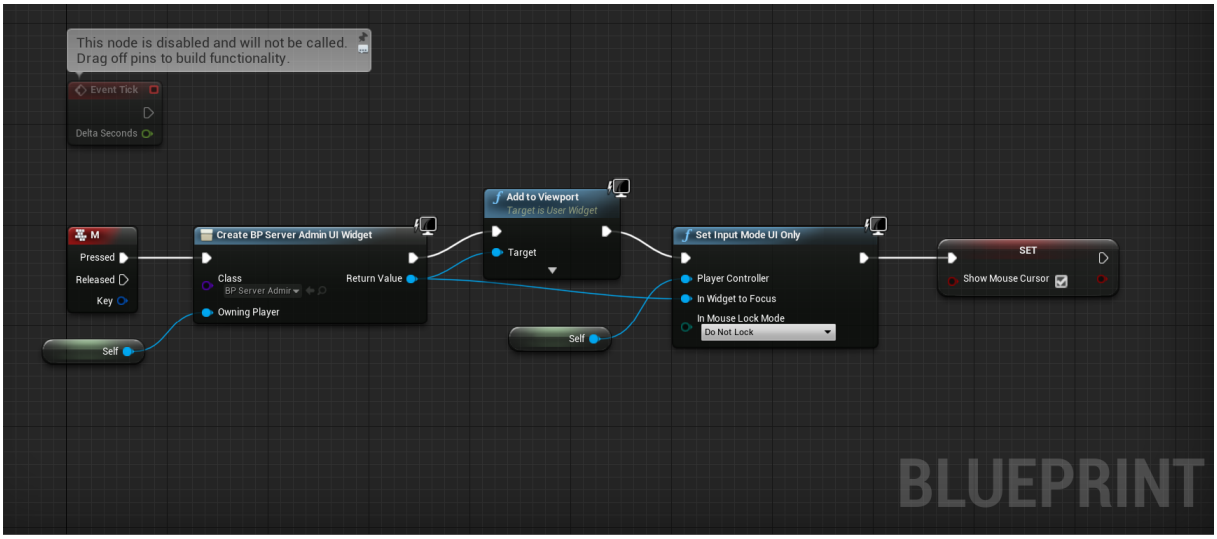
```
[ServerAdmin]
; Every server admin can conveniently be added here.
; Enter their 64bit SteamID to the list of ServerAdminSteamIDs
; and they will automatically be verified when they join.
+ServerAdminSteamID=12345678910111213
;+ServerAdminSteamID=12345678910111213
; Use the password option if you cannot or do not want to use any of the
; unique OnlineSubSystem user IDs.
; Note: Adding a ServerAdminSteamID will disable the password prompt.
AdminPassword="ThisIsMyPassword"
; Friendly Fire enabled or disabled
TeamKillsEnabled=1
; Team Kill threshold until the TKer is punished
TeamKillLimit=5
; Team Kill punishment: 0 = kick, 1 = ban
TeamKillPunishment=0
; Team Kill count decay. Decreases the count of accumulated TKs per player by one per X minutes
TeamKillCountDecay=1
; Configurable path for looking up the game's maps'
MapsPath="/Game/Maps"
; Maps excluded from showing up in the admin menu
; such as the entry map or submaps that live in the Game/Maps folder
; Note: It is recommended to store sublevels in a separate folder of Game/Maps/.
; That will automatically exclude them from the map list in the admin menu.
+ExcludedMaps="ShooterEntry"
+ExcludedMaps="Highrise_Audio"
+ExcludedMaps="Highrise_Gameplay"
+ExcludedMaps="Highrise_Lights"
+ExcludedMaps="Highrise_Meshing"
+ExcludedMaps="Highrise_Vista"
+ExcludedMaps="Highrise_Collisions_Temp"
; Map Rotation List. Add all maps that are supposed to run by default here
+MapRotation="/Game/Maps/Highrise"
+MapRotation="/Game/Maps/Sanctuary"

```

CALLING THE SERVER ADMIN MENU IN GAME

This last step will show you how to call the ServerAdmin UMG menu. This implementation is "quick and dirty" and it is strongly encouraged to integrate it into your own ingame menu structure instead. However, for a proof of concept it will do.

Launch the editor of the project you just compiled and create a new Blueprint PlayerController and Blueprint Game Mode. Have them inherit from **ShooterGameController** and **ShooterGame_TeamDeathMatch** respectively. In the BP Game mode, reference the BP PlayerController as default Player Controller and then use the new BP GameMode as GameMode override in the world settings or project settings.

Now, open the BP Player Controller and add this bit of logic:



In the *Create Widget* node, reference **BPServerAdminUI** as class.

It will create and open the ServerAdmin menu when you press the key "M" while you are playing in game.