Community Express SDK

Copy DLLs into place:

Create a file called steam_appid.txt and type your Steam AppID inside of the file and nothing else. (Our example uses 480)

Copy the following files into “Program Files\Unity\Editor”:

Steam.dll

steam_api.dll

steamclient.dll

tier0_s.dll

steam_appid.txt

CommunityExpressSW.dll

CommunityExpress.dll

UnityCommunityExpress.dll

Copy the following files into the directory that will be running your game's executable:

Steam.dll

steam_api.dll

steamclient.dll

tier0_s.dll

steam_appid.txt

CommunityExpressSW.dll

Copy the following files into “your game's executable folder/ “gamename”_Data/Managed” directory:

CommunityExpress.dll

UnityCommunityExpress.dll

Copy the following files into your project's “Assets/Plugins” directory (create Plugins directory if it doesn't exist) :

CommunityExpressSW.dll

CommunityExpress.dll

UnityCommunityExpress.dll

Modify Unity's Compatibility Settings:

In the Unity Editor, modify the “PC and Mac Standalone” settings and change “Api Compatibility Level” from “.Net 2.0 Subset” support to “.Net 2.0” .

This can be done by selecting “File-> Build Settings”, selecting “PC and Mac Standalone” from the list of platforms, and click “Player Settings” at the bottom left. From “Player Settings”, under “Per-Platform Settings”, click the “Other Settings” tab and you will find “Api Compatibility Level” under “Optimization”. Change “.Net 2.0 Subset” to “.Net 2.0”.

Import Community Express DLL's as Assets:

Import the dlls in the “Assets/Plugins” directory by clicking Assets->Import New Asset... and selecting the dlls.

Change Compile Execution Order:

The UnityCommunityExpress needs to be compiled before the rest of the script. To do this, go to Edit->Project Settings->Script Execution Order and drag the UnityCommunityExpress script, located in the UnityCommunityExpress asset, on the “Default Time” box in the inspector. Then drag the “UnityCommunityExpress” box above the “Default Time” box and click Apply.

Create instance of Unity Community Express in Editor:

Create an empty game object (GameObject->CreateEmpty) and drag the UnityCommunityExpress script onto the empty game object.

Using instance in script:

You can use the gameobject.GetComponent method (http://docs.unity3d.com/Documentation/ScriptReference/GameObject.GetComponent.html)

or you can access the instance directly, since there should only ever be one UnityCommunityExpress component on the stage.

Direct Access Example:

C#

```
private UnityCommunityExpress communityExpress = UnityCommunityExpress.Instance;
```

JavaScript

```
private var communityExpress : UnityCommunityExpress = UnityCommunityExpress.Instance;
```

Creating a Server:

Create two ushort variables, these will be our listen port and master port:

```
ushort listenPort = 8793;
ushort masterPort = 27015;
```

Create a button in Unity's OnGUI method:

```
if(GUI.Button(new Rect(xPosition, yPosition, width, height), “Create Server”))
{
}
```

Inside of this button, Initialize a server with Unity's Network.InitializeServer method:

```
Network.InitializeServer(32, masterPort, true);
```

32, is the max number of connections allowed. masterPort is the port that this network will listen to. true sets the useNAT.

After we set up the Unity server, we need to Initialize the steam server:

```
communityExpress.GameServer.Init(    false,                              // Is this a dedicated server
                                     System.Net.IPAddress.Any,           //Set IP of Steam server
                                     listenPort,                         //Steam server port
                                     listenPort,                         //Steam query port
                                     masterPort,                         //Our Unity server port
                                     masterPort,                         //Spectator Port
                                     CommunityExpressNS.EServerMode.”server mode”,//See below for modes
                                     “Server Name”,                      //Your server name
                                     “Spectator Server Name”,            //Your spectators server name
                                     “Region Name”,                      //Your regian (US, UK, etc.)
                                     “Game Name”,                        //Your Games Name
                                     “Game Description”,                 //Your Games Description
                                     1.0.0,                              //Your Games version
                                     “Map Name”,                         //Your Games Map Name
                                     false,                              //Password protected
                                     “Mod Dir”,                          //Mod directory (for filters)
                                     OnGameServerClientApproved,         //Called when client approved
                                     OnGameServerClientDenied,           //Called when client denied
                                     OnGameServerClientKicked            //Called when client kicked
                                     );
```

Skipping over the obvious ones, the CommunityExpressNS.EServerMode is a enum that contains 3 enumerations:

eServerModeAuthenticationAndSecure `//Authenticate users, list on the server list and VAC protect clients`

eServerModeAuthentication `//Authenticate users, list on server list, don't run VAC on clients that connect`

eServerModeNoAuthentication `//Don't authenticate user logins and don't list on the server list`

“Mod Dir” is normally used for a game that has different servers for different mods. We are going to use it to filter our server search later on.

OnGameServerClientApproved is a function that you will have in your code. Whenever you have EServerMode set to something other than no authentication, this function will get called when someone connects. You can use it to instantiate player characters or even alert other players that someone else has connected.

OnGameServerClientDenied is essentially the opposite of OnGameServerClientApproved in that if someone tries to connect to a secured server and cannot be authenticated, this function will be called. You can use this to tell the person why they couldn't connect.

OnGameServerClientKicked is called when a player is booted from the server. This can be used to alert players still in the game that someone was kicked and to alert the person why he/she was kicked.

Documentation:

UnityCommunityExpress.Instance.GameServer.Init(***bool, IP Address, ushort, ushort, ushort, ushort, EServerMode, string, string, string, string, string, string, string, ushort, bool, OnGameServerClientApproved, OnGameServerClientDenied, OnGameServerClientKicked*** **):**Used to initialize the Steam server after you have created a Unity server.

UnityCommunityExpress.Instance.Matchmaking.RequestInternetServerList(***Dictionary<string, string>, OnServerReceivedCallback, OnServerListReceivedCallback*****):** Gets all of the servers for your appID. The Dictionary<string, string> is a filter used to narrow your search results.
OnServerReceivedCallback gets called every time a new server is found.
OnServerListReceivedCallback is called when the request has reached the end of the list of servers.

UnityCommunityExpress.Instance.RemoteStorage.FileExists(***string*** **):** Checks if a file exists in the Steam remote storage by file name. Returns bool (true=file exists; false=file doesn't exist).

UnityCommunityExpress.Instance.RemoteStorage.WriteFile(*string, string*): Creates a file or overwrites pre-existing file. The first string is the file name, the second is what will be written to the file.

UnityCommunityExpress.Instance.RemoteStorage.GetFile(*string*): Retrieves file from Steam remote storage, if one exists, by file name. Returns CommunityExpressNS.File.

UnityCommunityExpress.Instance.UserStats.RequestCurrentStats(*OnUserStatsRecieved, string[]*): Calls for a users stats based on their SteamID. OnUserStatsRecieved gets called when the requested stats have been found. The string[] is an array of stat names that you are requesting.

UnityCommunityExpress.Instance.UserAchievements.InitializeAchievementList(*string[]*): This is used to initialize the achievements for the player so they can be unlocked. The string[] contains all of the achievement names to be unlocked.

UnityCommunityExpress.Instance.Friends: This is a list of the players steam friends.

UnityCommunityExpress.Instance.ConvertImageToTexture2D(*Image*): This is used to convert a players avatar image to a format that Unity can display. Image is either LargeAvatar, MediumAvatar, SmallAvatar.

UnityCommunityExpress.Instance.Leaderboards.FindOrCreateLeaderboard

(*OnLeaderboardRetrieved, string, CommunityExpressNS.ELeaderboardSortMethod, CommunityExpressNS.ELeaderboardDisplayType*): This creates or finds a leaderboard by the leaderboards name. OnLeaderboardRetrieved is called whenever the requested leaderboard has either been succesfully created or found. The string is the name of the leaderboard you are looking for or wish to create. CommunityExpressNS.ELeaderboardSortMethod is an enum and is used to sort the scores. CommunityExpressNS.ELeaderboardDisplayType is an enum used to arrange the scores.

UnityCommunityExpress.Instance.UserAchievements.UnlockAchievement

(*CommunityExpressNS.Achievement, bool*): This is called whenever you want to unlock an achievement. The CommunityExpressNS.Achievement contains the AchievementName, DisplayName, DisplayDescription, and IsAchieved. Alternativly you can use the AchievementName instead of the Achievement. The bool is storeStats and if it is set to true, the unlocked achievement will go ahead and be saved to the players profile.

UnityCommunityExpress.Instance.UserStats.WriteStats(): This method will save all of the stat values that have been set locally.

CommunityExpressNS.Friend: Contains friend's PersonaName, Avatar, SteamID, and PersonaState

CommunityExpressNS.File: Contains FileName, FileSize, Exists, Delete(), and ReadFile(). ReadFile() returns a string.

OnLeaderboardRetrieved(***CommunityExpressNS.LeaderboardEntries*** **):** Called whenever a requested leaderboard has been found or a new one has been created. LeaderboardEntries is a list of all the players, their scores, and their score details.

OnUserStatsRecieved(***CommunityExpressNS.Stats, CommunityExpressNS.Achievements*** **):** Called when the stats requested have been found. Passes in Stats which is a list of all the stats requested. Achievements is a list of achievements that is always null.

OnGameServerClientApproved(***CommunityExpressNS.SteamID*** **):** Called when a player succesfully connects to the server. The players SteamID is passed in.

OnGameServerClientDenied(***CommunityExpressNS.SteamID, CommunityExpressNS.EDenyReason, string*** **):** Called when a player fails to connect to the server. The players SteamID, the reason they couldn't connect, and other optional data about the reason are passed in.

OnGameServerClientKicked(***CommunityExpressNS.SteamID, CommunityExpressNS.EDenyReason*** **):** Called when a connected player is booted from the server. The players SteamID and the reason they were booted are passed in.

OnServerReceivedCallback(***CommunityExpressNS.Servers, CommunityExpressNS.Server*** **):** Called whenever a new server is found. Passes in Servers which is the list of servers found thus far, and server which is the server found at that callback.

OnServerListReceivedCallback(***CommunityExpressNS.Servers*** **):** Called when all the servers have been found. Passes in Servers which is the list of all the servers found.

PersonaName: This is a players Steam Name.

PersonaState: This is a type of CommunityExpressNS.EPersonaState. Can be compared to EPersonaState to display a players current online status.

LargeAvatar, MediumAvatar, SmallAvatar: These are a players account image. To display them, use the ConvertImageToTexture2D() method.

UploadLeaderboardScore(*CommunityExpressNS.ELeaderboardUploadScoreMethod, int, List<int>*): This is a method within CommunityExpressNS.LeaderboardEntries and is used to upload a players scores. The CommunityExpressNS.ELeaderboardUploadScoreMethod is an enum which contains different enumerations on how to upload the score. The int is Score which is the main number that you wish to upload. The List<int> is ScoreDetails. (Example: If Score is the number of kills, then ScoreDetails is number of headshots, grenade, and road kills)

CommunityExpressNS.LeaderboardEntries: A variable of this type contains a list of LeaderboardEntry, which is a PersonaName, Score, GlobalRank, and ScoreDetails. LeaderboardEntries contains the method called UploadLeaderboardScore() which is used to upload a players stats to this leaderboard.

CommunityExpressNS.ELeaderboardUploadScoreMethod: Contains three types:

k_ELeaderboardUploadScoreMethodKeepBest, k_ELeaderboardUploadScoreMethodForceUpdate, k_ELeaderboardUploadScoreMethodNone. KeepBest will keep the players best score, which is determined by CommunityExpressNS.ELeaderboardSortMethod. ForceUpdate will overwrite the score no matter what the sorting method is. None will do nothing.

CommunityExpressNS.EPersonaState: This is an enum of a player's online status and contains four enumerations. EPersonaStateOffline, EPersonaStateOnline, EPersonaStateBusy, EPersonaStateAway, EPersonaStateSnooze.

CommunityExpressNS.ELeaderboardSortMethod: Contains three types: k_ELeaderboardSortMethodAscending, k_ELeaderboardSortMethodDescending, k_ELeaderboardSortMethodNone. This tells the server if the top score is the highest number or the lowest number then sorts it accordingly.

CommunityExpressNS.ELeaderboardDisplayType: Contains four types: k_EleaderboardDisplayTypeNumeric, k_EleaderboardDisplayTypeTimeMilliSeconds, k_EleaderboardDisplayTypeTimeSeconds, k_EleaderboardDisplayTypeNone. Tells the leaderboard what the score represents. Numeric represents a simple number score, TimeMilliSeconds represents milliseconds, and TimeSeconds represents seconds. None of course represents nothing.