Overview

This article outlines some of the built-in Steam features bundled with the CRYENGINE EaaS subscription model.

Steam CVars

net_useSteamAsOnlineLobby

Set to 1 if you want to use Steam for the online lobby/matchmaking part. This also includes initialization for User Stats, Achievements and Leader boards.

This requires having a steam_api.dll or steam_api64.dll in your bin folder, a valid AppID and have steam running with a signed in user who owns that AppID.

sys_steamAppId

This is the appID that will be used by the launcher when either the cloud saving or the lobby is enabled. The default is 0, CRYENGINE's appID is 220980 which you can use to test basic features (more on this below).

sys_useSteamCloudForPlatformSaving

When this is set to 1, it will save and load the profile data and save games to the cloud.

net_steam_resetAchievements

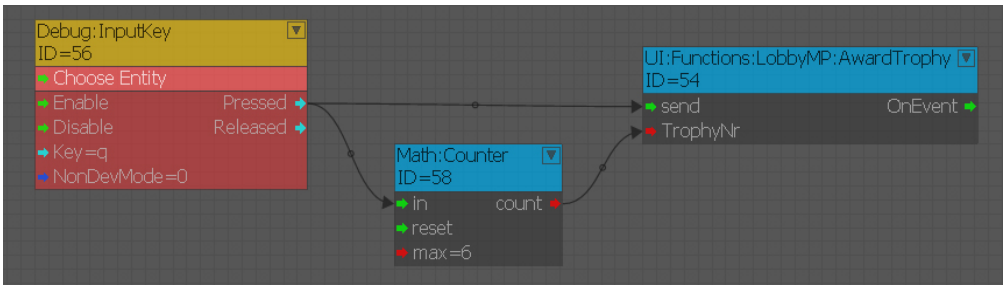
When set to 1, the engine will ask steam to reset all achievements for the appID, meaning there will be no achieved achievements anymore for that user.

Supported features

At the moment, all the features listed below are only available in the multiplayer gamemode (they work off the back of an online lobby). Also make sure that the controller is locked, so we know which user to get (Flownode: LockController).

Achievements

Achievements are awarded with the generic node "AwardTrophy".



For the 220980 app there are some example Achievements set up. When you are doing your own achievements, make sure you change the amount and the names in CrySteamReward.cpp. The index in EAchievements is the index used in the flownode.

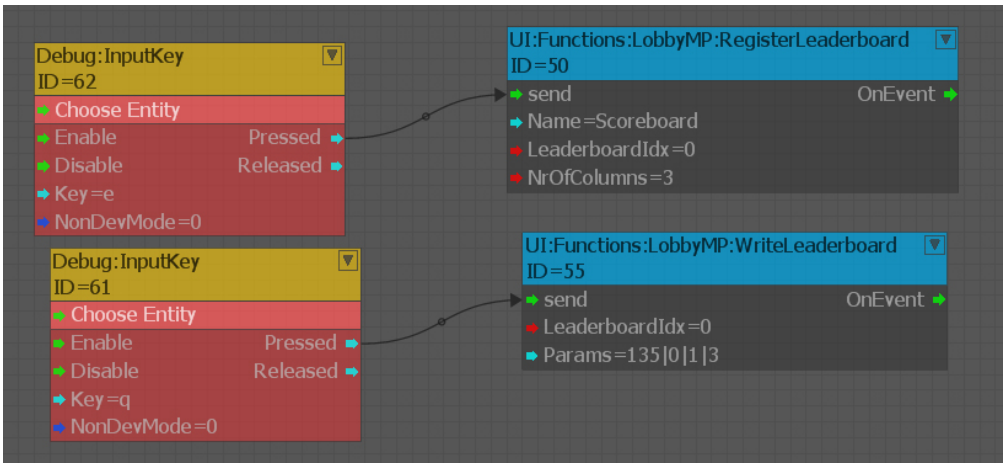
```
// Defining our achievements
enum EAchievements
{
	Achievement1 = 0,
	Achievement2,
	Achievement3,
	Achievement4,
	Achievement5,
};

// Create the achievement buffer
CrySteamReward g_Achievements[] =
{
	_ACH_ID( Achievement1, "Achievement1" ),
	_ACH_ID( Achievement2, "Achievement2" ),
	_ACH_ID( Achievement3, "Achievement3" ),
	_ACH_ID( Achievement4, "Achievement4" ),
	_ACH_ID( Achievement5, "Achievement5" ),
};
```

Leader boards

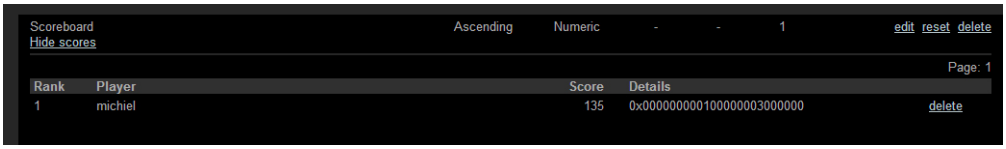
For appID 220980 there is an example leader board called 'Scoreboard' that you can test with. The steam leader board, just as any leader board in CRYENGINE works through a generalized system.

First you need to register the leader board. A good example, to get started quickly, is the Registerleaderboard node. Fill in the name and the index. Depending on the back-end setup and platform either the name or the index is relevant.

In the case of Steam only the name needs to be correct, the index can be of your choosing. After registering the leader board you can start reading from it and writing to it, below is an example where information has been written to it.



This is the result in the Steam back-end: The score column is populated, and the custom columns have been compressed and saved in the 'extra' column (0, 1 and 3).



Matchmaking

When **net_useSteamAsOnlineLobby** is set to 1 the online lobby will use Steam for matchmaking. This means you can create rooms and search for them as well as being invited to them via CRYENGINE or via Steam (in the friends panel).

Cloud saving

When **sys_useSteamCloudForPlatformSaving** is set to 1, it will store the save games and profile data into the cloud. The current setting for 220980 is 100 files maximum with a 10 MB quota.

It will save and load the profile data and save games to the cloud instead of the user folder. The steam cloud folder can be found in `C:\Program Files (x86)\Steam\userdata\<SteamUserID>\<SteamAppID>\remote\`

When this is enabled calls to GetISystem()->GetPlatformOS->GetSaveReader() and GetSaveWriter() you will return a reader/writer for steam cloud instead.

If you reach the file limit you will have to call **sys_wipeSteamCloud** to delete files from the cloud - at the moment you can't specify individual files to be deleted.

User Stats

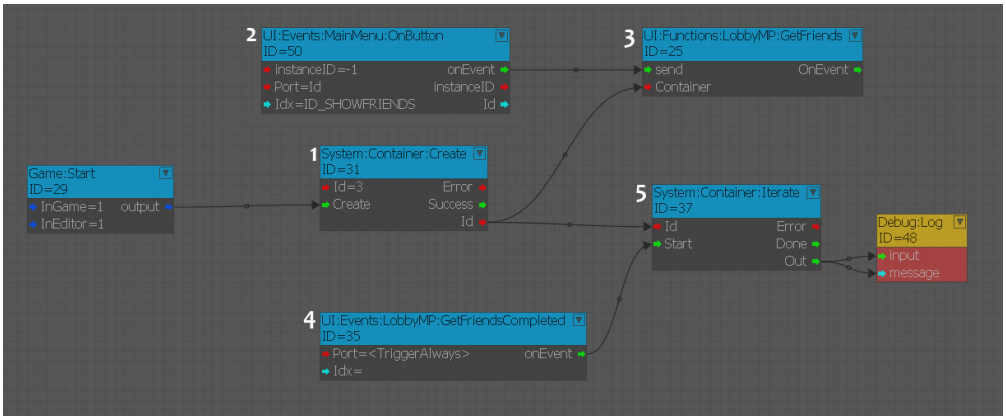
The stats that are available online can be found in: `<GameFolder>\Scripts\Network\OnlineAttributes.xml`

The flownodes **PlayerProfile:ProfileAttribute** and **PlayerProfile:OnlineAttributes** are a good starting point to see how to work with these attribute values.

When the LoadOnlineAttributes and SaveOnlineAttributes is called it loads and saves the values that are set in the playerprofile to the Steam back-end, which will be persistent

Friends

There are some nodes to give you quick access to friends, use them as a reference when making your own friend functionality, as the node might restrict you too much.



First we create a container (1) to hold the data for us. Then when the user pressed a button (2) we asked the system to fetch the friends and store them in a container (3).

GetFriendsCompleted is the callback event for GetFriends, meaning the fetching was done (4), Then simply iterate over the data and output the result.

```
                pFriends->FriendsGetFriendsList(user, 0, 10, NULL, CUILobbyMP::GetFriendsCB, (void*)
containerIdx);
```

This is the most important line related to getting the friends, notice the 0 and 10 - this is the start of the list and the maximum number of friends you want to grab.