

## Network Options

**UFE 2** comes with a completely revision of its core physics and network mechanics. The second iteration allows you to run online matches in a smooth lag-free environment thanks to its native *rollback mechanics* (similar to [GGPO](#)) and *deterministic physics*. The netcode also uses raw *manual tracking* to its own variables, optimizing CPU usage (including mobile devices).

The new network options also comes with its own client/server structure (using Photon Network or UNet) allowing direct player connection without the use of IP.

For detailed instructions on UFE 2 Netcode and how to convert your previous UFE 1.x project, [click here](#).

▼ Network Options

?

Online Service

Network Service: Photon

Photon Service: Photon Server

Particle Control

Control Spawned Particles ☒

Sync Random Seed ☒

Override Simulated Speed ☒

Animation Control

Force UFE Animation Control ☒

Disable Root Motion ☐

Disable Blending ☐

Disable Rotation Blend ☐

Package Options

Network Message Size Size 16 Bits

Input Message Frequency Every Frame

Only Send Input Changes ☒

Rollback Netcode

Enable Rollback ☒

Max Fast-Forwards Per Frame 60

Input Buffer Size: 60

Spawn Buffer Size: 60

Rollback Balancing Aggressive

Frame Delay Netcode

Frame Delay Type Fixed

Default Frame Delay: 2

Apply Frame Delay Offline ☐

Synchronization Test

Set Host As AI ☐

Set Client As AI ☐

Desync Action Disabled

Float Desync Threshold 0.01

Sync Check Frequency Every Frame

Log Sync Messages (Console) ☒

Record Post-Rollback Frames ☒

Generate Variable Log ☒

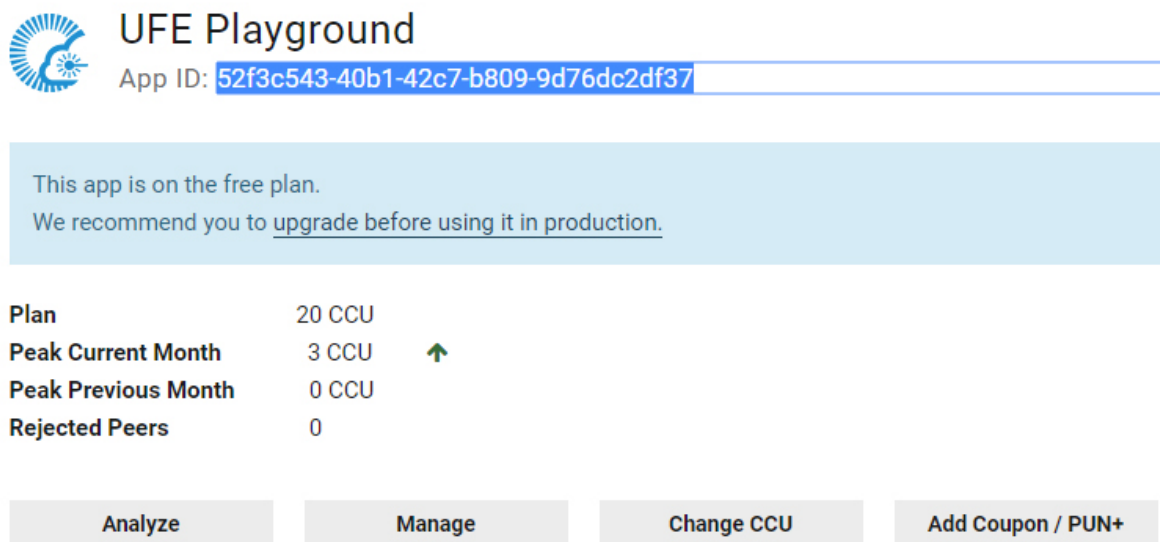
Exported File Path: ../VariableLog.txt

Recording Buffer 360

## Setup

You need **UFE 2 Standard/PRO/Source** and **PUN2** to use this feature.

1- Sign up for free at <https://www.photonengine.com/en/PUN>. You will be asked to setup the App ID. You can find it under Public Cloud → Applications inside your account page. Copy your App ID as displayed on the web page (example):



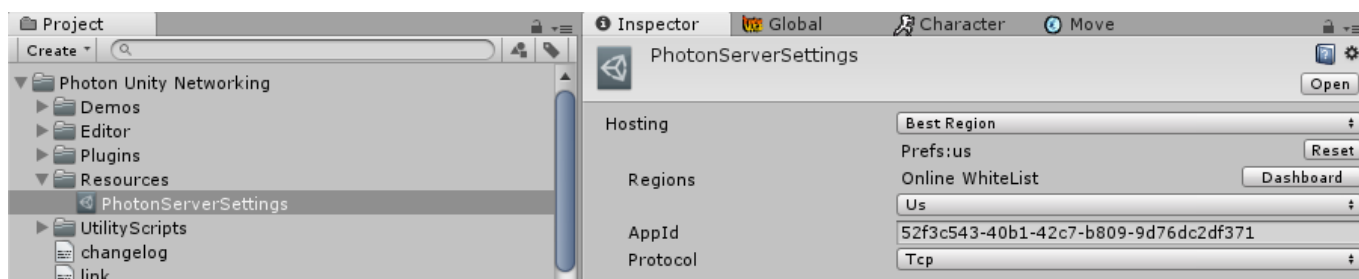
**UFE Playground**  
App ID: 52f3c543-40b1-42c7-b809-9d76dc2df37

This app is on the free plan.  
We recommend you to [upgrade before using it in production](#).

Plan	20 CCU
Peak Current Month	3 CCU ↑
Peak Previous Month	0 CCU
Rejected Peers	0

[Analyze](#)
[Manage](#)
[Change CCU](#)
[Add Coupon / PUN+](#)

2 - Download and unpack the photon package under your UFE project, and under \Photon\PhotonUnityNetworking\Resources\PhotonServerSettings, paste your App ID under Server/Cloud Settings → App Id PUN:



3 - Now, extract the files at UFE\Engine\ThirdPartySupport\Photon2APIConnector.unitypackage to connect UFE and Photon. If everything worked, you will notice the **Network Options** under Global Editor is now unlocked. Go to *Online Service* and set Photon as your Network service. Hit play and you should see the Network option enabled.

You can also set a variety of connection options with the Photon Server Settings such as protocol used, hosting type and search by separated search regions. For more information visit [this link](#).

## Online Service

**Network Service:** Choose which online server to use. If you wish to use Unity (up to Unity version 2018) you must have a [Developer Profile](#) with access to Multiplayer. You must then connect your Unity to your profile via Window → Services. For Photon, follow the instructions above. For more on deterministic network take a look at [this tutorial](#).

**Photon Service:** (Photon Only) Choose which cloud service to be used in conjunction with [Photon](#).

---

## LAN Games

**Network Port:** Set which Network port should the game use to connect between the clients in the local network. If you are testing your game using Play Mode under Unity, make sure your router and/or firewall are not blocking the port used.

**Lan Discovery Port:** The port used in the network to search for games.

**Lan Discovery Broadcast Interval:** The interval for the host to broadcast its signal.

**Lan Discovery Search Interval:** The interval for the broadcast of which the game searches for hosts.

**Lan Discovery Search Timeout:** How long before it stops searching.

---

## Animation Control

**Fake Network:** Toggle to run local network packaging tests. It forces local PvP matches to emulate a network game. Useful to test the Netcode without running 2 clients.

**Force UFE Animation Control:** UFE Netcode works by forcing a full synchronization between 2 clients. To ensure there are no desyncs, toggle this option to make it so the UFE Engine is always in charge of the [animation control](#) during online matches regardless of your offline options.

**Disable Blending:** Override all animation blending states to be 0. Using this feature helps you “mask” the times there is a rollback as it needs to refresh the animation component.

**Disable Rotation Blend:** (Mecanim only) Override the blending done during rotation to 0.

---

## Package Options

**Network Message Size:** How big should the package be. If your game only uses 4 buttons, use 8 bits. Use 16 bits for more than 4, and 32 bits if you plan to use more than 12 buttons or if you need to send different data through the same broadcast system (such as debug logs or sync checks).

**Broadcast Frequency:** How often should the network attempt to send changes to the inputs? The higher the frequency, the more bandwidth is used.

**Only Send Input Changes:** If enable, the network will only send data that has been changed (instead of sending every broadcast frequency tick)

---

## Rollback Netcode

**Allow Rollbacks:** Enable Rollbacks in your game (Make sure you've gone through [setup](#)).

**Track UFE Variables:** Toggle so UFE will use the *auto tracking* technique to search for its interface (UFEInterface) and attributed variables (RecordVar) on its own instantiated classes. The deeper a variable is from the UFE instance, the more CPU power it consumes.

**Max Fast-Forward Per Frame:** When using rollback, depending on the distance between packages sent, how many frames can your game recover from. The more frames, the more CPU power it needs.

**Input Buffer Size:** The amount of inputs that can be stored and reproduced in case of a rollback.

**Spawn Buffer Size:** The amount of game objects that can be stored in a “safe pool” before they get *garbage collected*. Adjust this value based on the amount of object spawns (effects, projectiles, etc) you have at once in your game.

**Rollback Balancing:** Use this option to balance out the rollback experience between the players.

- *Disabled:* Only one of the players feels the effects of rollback. The host (or player with better connection) plays the game like there is nothing wrong.
  - *Conservative:* One player feels the effects of rollback more often the other.
  - *Aggressive:* Both players share the same impact when a rollback is needed to stabilize the connection. Use this option to “share” the impact of a rollback between players (recommended).
- 

## Frame Delay Netcode

**Frame Delay Type:** If set to Auto, UFE will determine based on the ping what frame delay better fits the match. if you plan to toggle Apply Frame Delay Online, use Fixed instead for a better user experience.

**Min Frame Delay:** When connection is *ideal* (<10ms ping) what is the best value for *minimum* input delay in your game? Large values makes the game more tolerant to lag spikes. On ideal connections it's recommended that you leave at least 4 frames (for 60 fps) or 3 (for 30 fps) - according to your [FPS structure](#).

If you are using Rollback, the game will always attempt to use this value. Games less dependent on animation blending (such as 2D) can try for lower values, while games more dependent on hardware optimization (or better visual fidelity) should try for higher.

**Max Frame Delay:** In case of bad connections, how far should the input delay before increase before it relies on rollback to keep the network packages synchronized?

**Apply Frame Delay Offline:** Makes it so every game (including CPU matches and Training Mode)

are always running with the Min Frame Delay value.

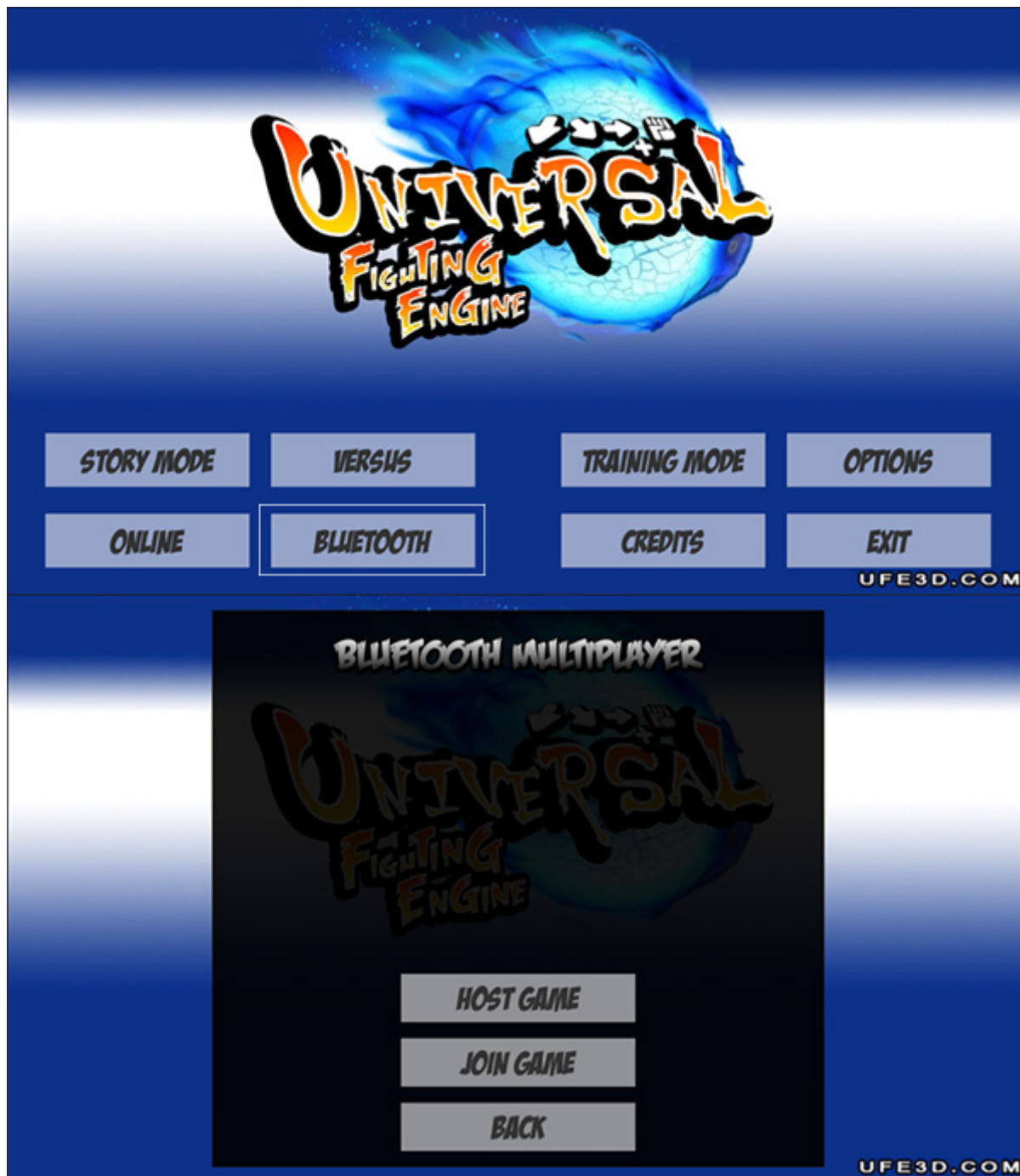
---

## Bluetooth

If you plan on having your game running on Android devices you might want to set a bluetooth connection using [Simple Bluetooth for Android](#) available on the Asset Store.

To install it, once you import the asset, navigate go to .UFE\Engine\ThirdParty. And unpack BluetoothConnector.unitypackage. If it worked, you should see the following message under Network Options → General: Bluetooth installed.

In order to test it, go to [GUI Options](#) and under *Screens*, replace the current Main Menu UI with the alternative MainMenuScreenBluetooth.prefab (located under UFE\Demo\GUI\UI Prefabs\). You should also see the option Bluetooth Game under the Network Mode group as well. Drag the BluetoothGameScreen.prefab to that field if its not there already.



Export your game to 2 Android devices and there you have it. Just select the bluetooth option and begin fighting!

## Hints

Networking is a relatively complicated aspect of the engine. Don't be alarmed if you can't get running at first. Here are a few hints to help out on common issues:

- If you are having trouble connecting to the other peers, you might need to work on your router's [port forwarding](#).
- If you are experiencing lag, try changing the hosting option and protocol under

'PhotonServerSettings'. For more information on Photon options, [click here](#).

- Make sure you have no 'self-destruct' scripts attached to your particle effects. UFE needs to control the spawn and despawn of every GameObject. If you need to use one, use `UFE\Engine\Scripts\DestroyScript.cs` instead.

---

## Video Tutorial



### Video

---

Code access:

`UFE.config.networkOptions`

---

[< Back to Global Editor](#)

From:

<http://ufe3d.com/> - **Universal Fighting Engine**

Permanent link:

<http://ufe3d.com/doku.php/global:network?rev=1704251342>

Last update: **2024/01/02 22:09**

