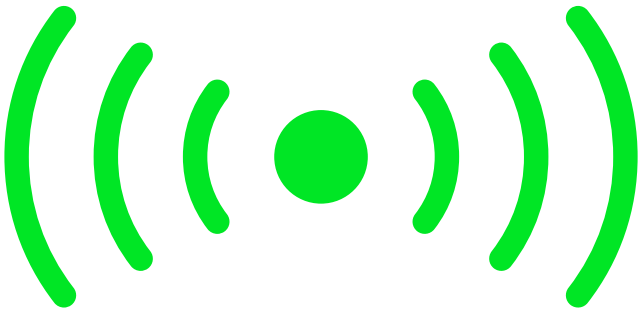
VOICE BRO

CUSTOMIZABLE

VOICE CLIENT & SERVER SYSTEM



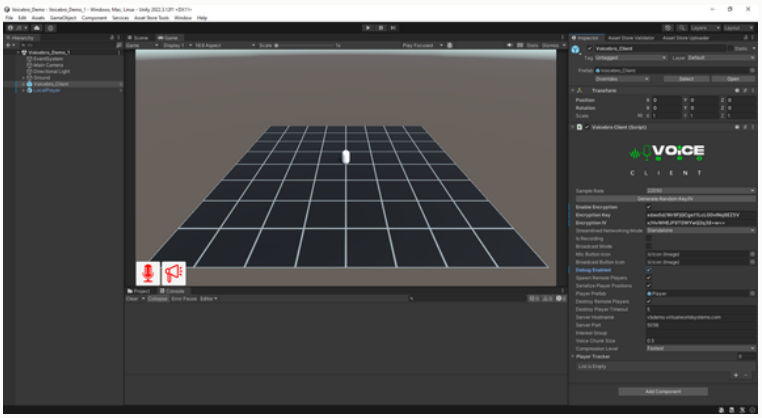
VER 1.0.5

TABLE OF CONTENTS

OVERVIEW

Voicebro is a versatile voice solution for Unity projects that lets you run your own servers without strict licensing mechanisms, that can run on the internet and closed intranet systems. It supports client-side cryptography and many other features.

Voicebro Server Console



Unity 2022.3.12f1 Demo Project

Voice bro consists of two main components:

A) Server software (binaries & source code)
B) Unity client plugin & scripts

All you have to do to get started is:

1) Import the Voicebro asset or UnityPackage into your Unity project

2) Open the demo scene provided with Voicebro

3) Click on the Voicebro_Client object in the scene and set the hostname or IP address that the server is running on (default port for server is UDP 5056, please adjust server firewalls as needed) -- or leave it set to our demo server for testing

4) Build the unity project OR hit play -- run on two or more machines or more than one instance of your build, editor, etc. to test multiuser functionality

NOTE: A demo server is provided to customers of this product at vbdemo.virtualworldsystems.com, but it is not guaranteed to be available at all times nor is it meant for heavy production usage. You will need to compile and run your own Voicebro server using the provided zipped visual studio 2022 project on a VPS or a physical server somewhere.

Why run my own voice server?

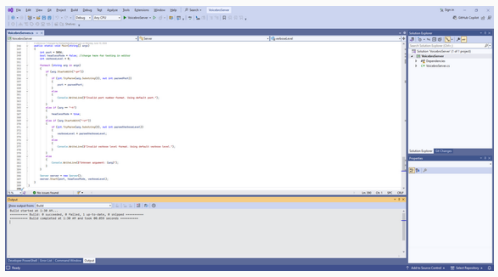
Cost & savings - You will now be running your voice network for your game/project at the cost of a VPS, Droplet, or for intranet only projects you would run it on a server locally inside of the DMZ.. If your still using third party providers for other mesh networking in parallel to Voicebro, at least the cost of your voice services will be at cost and minimal

Encryption - Some projects in certain niche's of professional Unity project development such as Education, Healthcare, Government, etc require that encryption is strong and privacy is fully protected. Voicebro uses client side only cryptography, so only the users that know the same key/iv/interest group can successfully hear/speak to each other.

Compatability & Ease of use - We have included plug 'n play functionality for Voicebro to make life super easy for developers already or planning on building projects with Netcode, NetcodePlus, Mirror, Photon Fusion or PUN2. You can also make your own custom integration directly in the VoicebroPlayer script.

SERVER

Voicebro has it's own server, and it needs to run somewhere such as a public VPS, or a physical computer inside your LAN. Your game client builds should all have the hostname/IP address & port of the same server.



To use your own server, you will need to compile the server yourself in Visual Studio 2022. Simply open the visual studio project after extracting the server source zip file outside of your Unity project.

The server source code is located inside the file
Assets\Voicebro\Server\VoicebroServer_Source.zip file

You can run multiple copies of the Voicebro Server on one system at a time. There are some command line flags and parameters that can be passed into then when launching them via your mechanisms, see below.

Example to start that copy of the voice server on port 12345: VoicebroServer.exe -p=12345

Start in headless mode (text lines only, no text based GUI):
VoicebroServer.exe -h

Set verbose level (value range is 0 to 5, 0 is default)
Example with verbsose level set to 2:
VoicebroServer.exe -v=2

All three options at once:
VoicebroServer.exe -p=12345 -h -v=2

NOTES: Performance will be dependent on the hardware and network abilities & limitations of the server itself, but because the server project is made and compiled directly in visual studio .NET as a C# cross-platform console app, the performance is actually quite decent with low overhead. It will be up to you to decide how and where you want to launch/shutdown copies of the voice server and how to point your game-client to them appropriately per scene, world, player group, etc. -- be it per port or via interest groups on one instance of the server. It usually makes sense to use 'interest groups' for the concept of 'rooms' when overall you have lower daily active user counts, or run multiple voice servers on different ports and/or hosts for larger daily active user counts.

When making your own builds of the VoicebroServer, here are some helpful PowerShell commands.

After doing a clean/build etc, you can publish your compiled server binary and include all runtime files all into a single binary file, examples for windows & linux platforms:

dotnet publish -c Release -r win-x64 --self-contained -p:PublishSingleFile=true
dotnet publish -c Release -r linux-x64 --self-contained -p:PublishSingleFile=true

If you are running your VoicebroServer on Linux such as an Ubuntu VPS, Elastic Compute Cloud, Droplet, etc - you will need to install LibSSL and .NET Runtime SDK for the binary to launch properly:

sudo apt-get install -y dotnet-runtime-7.0
sudo apt-get install -y dotnet-runtime-8.0
(You can change the Framework version to 7.0/8.0/etc if you prefer in the VoicebroServer.csproj file directly)

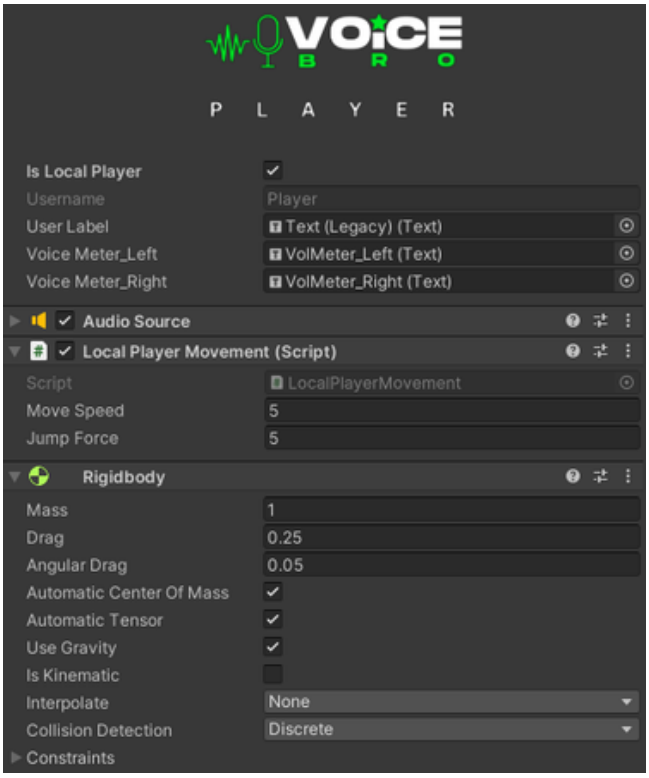
LibSSL installation:
sudo apt-get install libssl1.0.0

If you are looking for a good way to keep your Voicebro and other game servers running in the background and manage them on Linux, you might give 'tmux' a try, please take a look at tmuxcheatsheet.com

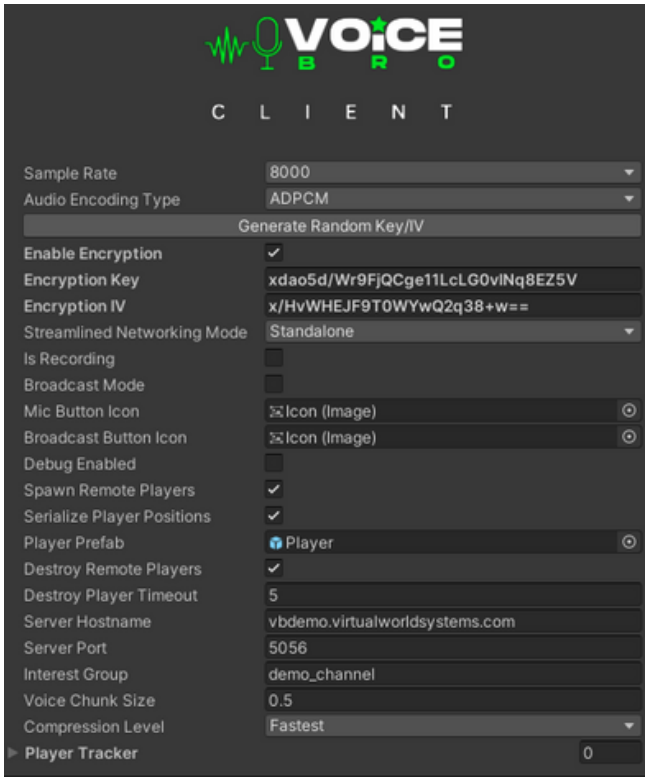
tmux new-session -d -t 0
tmux attach -t 0
Ctrl+B+D to detach and keep things running in there

CLIENT

The Voicebro client portion runs inside your Unity project. It consists of two main scripts: VoicebroClient.cs and VoicebroPlayer.cs; There are also two editor scripts, as well as the VoicebroCore.dll cross platform plugin.



Voicebro PLAYER script



Voicebro CLIENT script

The VoicebroPlayer script

Each script contains editor tooltips that explain what each value is used for:

isLocalPlayer: This value needs to be set to true before the script is enabled/activated/started on the local player prefab at runtime, and disabled on the spawnable player prefab itself

username/uuid: For visual debug purposes only, do not change at runtime; To change the local users name at runtime, please set it via VoicebroClient.instance.username via your other scripts before connecting.

UserLabel: Slot in a Text component if you want to show usernames of each player serialized thru VoiceBro on it

VoiceMeter_Left: Slot in a Text component if you want to use them for displaying voice level meters of each player (left)

VoiceMeter_Right: Slot in a Text component if you want to use them for displaying voice level meters of each player (right)

Overview

This script goes on your player prefab, isLocalPlayer must be set to true when it is spawned or ahead of time if already placed in the scene, and before the VoicebroPlayer script is activated, etc. It should be set to false by default in your prefab directly. The provided demo-scene has and is expecting the LocalPlayer prefab variant directly in the scene with it checked. Voicebro does not provide a demo option for spawning/destroying the *local player*, only *remote players*.

You should be spawning your players via your games backend/server-client system or cloud networking service directly for complete control of all the various serializations your players will need still, This is only a voice system. Extra serialization's sent within voice chunks only show up once per interval which is every half a second by default (voiceChunkSize), and usually isn't frequent enough to be serializing other things for the player anyways. VoicebroServer should probably be ran on a separate droplet/vps/ecc than your actual game instancing servers if they use up alot of memory.

The VoicebroClient script

The most important thing we need to understand about it is that there should only be one object in the scene with the VoicebroClient script on it, but that object+script should exist once in any scene that you want the users to have voice communications working in. Feel free to disable this object in the scene by default and enable the gameObject or script when you want the local user to actually try to connect to the voice server initially.

Please also make sure your encryption key/iv matches for all users connecting to the same server/port across all clients, as well as the compression setting. Interest groups can be changed on the fly to ‘change rooms’

Script properties

Each script contains editor tooltips that explain what each value is used for:

isRecording: When true, the local player will transmit audio from the systems default microphone/input device. This value can easily be changed via VoicebroClient.instance.isRecording via your own scripts

StreamlinedNetworkingMode: Please see detailed description below

AudioEncodingType: Choose from PCM (Uncompressed), μ-law, or ADPCM compression to control bandwidth usage

enableEncryption: When enabled, voice packets are encrypted & decrypted on the client side -- requires a KEY/IV to be generated using the provided button when enabled & the key needs to be the same by all receiving parties

encryptionKey: Encryption Key -- please generate with button above, and ensure the key&iv match on all connected clients

encryptionIV: Encryption Initialization Vector -- please generate with button above, and ensure the key&iv match on all connected clients

broadcastMode: When enabled, the local player will be heard by all other players in the interest group a.k.a. 'channel', reguardless of their world position (SpatialBlend = 2D), when disabled the local player will only be heard when someone is near them.

MicButtonIcon: Please slot in your UI button's image for the Mic/Mute button (optional)

BroadcastButtonIcon: Please slot in your UI button's image for the Broadcast button (optional)

debugEnabled: When enabled, verbose debug entries will be created in the console

spawnRemotePlayers: When enabled, Voicebro client will spawn remote players when voice packets are received without a player gameobject already existing in the Scene & PlayerTracker; This does NOT spawn the localPlayer -- it needs to be put in the scene manually when this mode is enabled

serializePlayerPositions: When enabled, remote player positions will be serialized through Voicebro (once every interval, based on voiceChunk size)

playerPrefab: The player prefab to use when spawnRemotePlayers is enabled

destroyRemotePlayers: Destroy remote players when they have timed out, based on the Voicebro system (for use with spawnRemotePlayers enabled)

destroyPlayerTimeout: The timeout (in seconds) that Voicebro decides a player has left/stopped talking to the server for use when destroyRemotePlayers is enabled

serverHostname: The hostname or IP address of the remote voice server

serverPort: The remote port of the voice server

interestGroup: The voice 'channel' that the local player should belong to, can be changed on the fly at runtime -- make this unique for each scene or 'room' in your project

voiceChunkSize: The size (in seconds) of each voice chunk/frame -- recommended value is 0.5

compressionLevel: Compression level of voice data -- must be consistent for all players in interest group/server

sampleRate: The sampling rate the local player encodes their mic data at, can be unique per player and changed on demand from other scripts via VoicebroClient.instance.sampleRate

Streamlined Networking Mode property

If you are using a common player networking system and want to run Voicebro in parallel with it, you can import the SDK for that system into your project first and then select it from the StreamlinedNetworkingMode dropdown on your VoicebroClient script in the scene. This will set a custom scripting symbol in your project that tells Voicebro how to serialize the Voice Actor’s UUID and username across all game clients so it can cross reference them to the actual player prefabs in the scene at runtime and know who’s AudioSource to spit out what signals thru. If you have this setting set to the default ‘Standalone’ mode, that is meant for when you want Voicebro to spawn/destroy/serialize player prefabs based on voice packets -- in standalone mode you need to make sure searchForRemotePlayers, spawnRemotePlayers, SerializePlayerPositions, and destroyRemotePlayers are all set to **true**, and in any other mode they are all set to **false**.

NOTE: Voicebro does not tunnel voice data through these other networking systems or providers, this simply allows it to work seamlessly with other player networking systems your project might already be or planning to use. Obviously, some of these are not good for offline/intranet only projects, but others are like Netcode/NetcodePlus.

Please take notice of the empty #ifdef blocks in VoicebroPlayer.cs titled ‘VOICEBRO_CUSTOM”, these are left intentionally blank with a comment in each one that shows you exactly where you would need to add code similar to each’s neighbor, to make your own custom connector for any other player networking system we did not provide one for already such as LiteNetLib.

FAQ

Frequently Asked Questions

Q: How many copies of the server am I allowed to run?
A: As many as you want, as many that your servers/resources can handle, there is no strict or enforced license system or subscriptions with Voicebro

Q: How many users can be in a 'voice room' or 'interest group' at once?
A: There is no hard limit, if you have a large amount of users, you may want to consider adding in a hard-limit in your game logic directly or use player-distance culling where you destroy players that are too far away. The answer is 'whatever the servers AND each players hardware can handle'

Q: From my own unity C# script, how can I make the local player muted/umuted?
A: Add the 'using Voicebro;' directive at top of script, then do:
```
VoicebroClient.instance.isRecording = true; //or false
```

Q: From my own unity C# script, how can I make the local player toggle broadcasting mode?:
A: VoicebroClient.instance.broadcastMode = true; //or false

Q: My online game project doesn't have the local player already in the scene, it gets spawned at runtime by some other networking system at runtime, how do I handle this?
A: Then you don't want the 'Standalone' option for the StreamlinedNetworkMode value on your VoicebroClient script in the scene, but rather it needs to match whatever system that is, and if it isn't listed you will need to make your own connector or ask us to do it for you (contact on discord and we can discuss)

Q: Can this voice system be used for projects with strict compliance requirements?
A: It supports custom encryption, closed system / offline / intranet usage, no license checks or phoning home to a third party server ever, and the only code that could be considered 'hidden' is inside VoicebroCore.dll, which can still be fully inspected via tools like ILSpy or DotPeek, it is not obfuscated -- you can even move the code from this directly into your unity project if you really want, but I think performance of this code is possibly better in an external library. With all that being said I think so yes, but please check your projects exact requirements yourself to be sure.

Q: Do I need to open/forward ports on my VPS, Droplet, ECC etc for VoicebroServer?
A: Yes, whatever ports (default is 5056) your VoicebroServer listens on need to be opened on a VPS/Droplet/ECC, or unblocked on firewall and port forwarding setup if the server device is behind a router.

Q: Am I allowed to resell any type services where customers basically pay us and we run Voicebro servers for them?
A: If you can find a market for it, we don't mind at all -- but you might want to check with Unity for any possible licensing or legal issues.

Q: We see on the product page that Voicebro works on Windows, Linux, macOS and Android builds, but does it work for iOS?
A: It hasn't been tested yet by us, but it is very likely that it already works just fine within iOS builds.

Q: With the various options for StreamlinedNetworkMode, what components from each networking SDK does VoicebroPlayer expect to be on the same transform as it (root of player prefab) at runtime for each mode? Can you also give extended info about each mode as well?
A: Yes! See below:

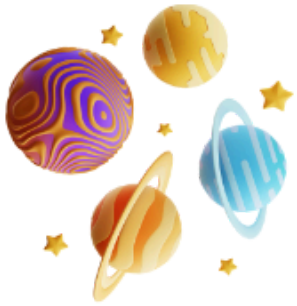
Version		Scripting Symbol	VoicebroPlayer inherits from	Sister Transform Component	Unity Versions Tested+Pass
2.0.3	✅	VOICEBRO_FUSION	NetworkObject	NetworkObject	2022.3.12f1, 6000.0.12f1
2.46	✅	VOICEBRO_PUN2	MonoBehaviour, IPunObservable	PhotonView	2022.3.12f1, 6000.0.12f1
89.8.0	✅	VOICEBRO_MIRROR	NetworkBehaviour	NetworkIdentity	2022.3.12f1, 6000.0.12f1
		VOICEBRO_NETCODE	NetworkObject	NetworkObject	-----------
NGO 1.4.0	✅	----------	----------	----------	2022.3.12f1
NGO 1.9.1	✅	----------	----------	----------	6000.0.12f1
1.0.3		VOICEBRO_NETCODEPLUS	SNetworkPlayer	SNetworkPlayer+SNetworkObject	-----------
NGO 1.6.0	✅	----------	----------	----------	2022.3.12f1
NGO 1.9.1	✅	----------	----------	----------	6000.0.12f1
1.0.5	✅	VOICEBRO_STANDALONE	MonoBehaviour	N/A	2022.3.12f1, 2021.3.12f1,
-----	✅	----------	----------	---	6000.0.12f1

Q: Does Virtualworld Systems have a community discord server?
A: Yes! You are officially invited, Here is the link!

Q: What projects did you base each 'StreamlinedNetworkMode' off of and test with?
A: Special thanks to the creators of the following demo's for this purpose:

Photon Fusion SDK demo 'FusionDemoGameplayHost'
Photon PUN2 SDK demo 'PUN Asteroids'
Mirror SDK demo 'MirrorTanks'
Netcode-Multiplayer demo by Rishav Nath Pati @ ConvAI
NetcodePlus by Indie Marc 'Tanks', 'Puzzle' and 'Simple' demos

VIRTUAL
WORLD
SYSTEMS



THANK YOU FOR YOUR SUPPORT

CONTACT

SUPPORT@VIRTUALWORLDSYSTEMS.COM
VIRTUALWORLDSYSTEMS.COM