Easy Code For Vivox

PlayMaker Extension

Last Updated - 7/28/2021

Supports Unity Game Engine

Vivox Voice and Text Chat is a separate free asset and does support all major platforms and consoles.

PlayMaker is a seperate paid Asset on the Unity Asset Store. PlayMaker is only required if you want to use the PlayMaker Extension part of this asset. If not you don't have to import it the PlayMaker part and are free to delete it without messing up core functionality

This Asset only supports

- Windows
- Android

This asset may work with IOS, Mac, Linux, or Consoles. Implement at your own discretion. Source is included so you can modify it.

This asset is a simple **API** to interact with the **Vivox Unity SDK** available in the **Unity Asset Store** as **Vivox Voice And Text Chat**.

This asset is built on top of **Vivox Voice And Text Chat** and will not work without it (You can also download the SDK through Vivox's website and this asset will still work if that is your preferred method). You must create an account with Vivox at Login - Vivox Developer Portal and agree to Vivox's Terms before you can use their services.

This Asset is free to modify, resell

Getting Started

Open the **EasyCodeForVivox** folder and check out the Demo Scene to see how easy it is to start using Vivox Voice Chat. If you open the **EasyScripts** folder you will find all the code necessary to get started with **EasyCodeForVivox.**

The **EasyExample.cs** script is the main script used in the demo scene if you want to see all the methods you will need to access most of Vivox's functionality.

EasyManager.cs is necessary for **EasyCodeForVivox** to work and you need to have at least one instance of EasyManager script attached to a gameobject in your scene unless you are inheriting from EasyManager or using the **EasyExample.cs** script. This is only required for any scene that needs to handle Vivox Voice or Text Chat functionality. Voice and Text communications will carry across scenes but without an **EasyManager.cs** script you cannot interact with **Vivox** to leave a channel or send a channel message, etc.

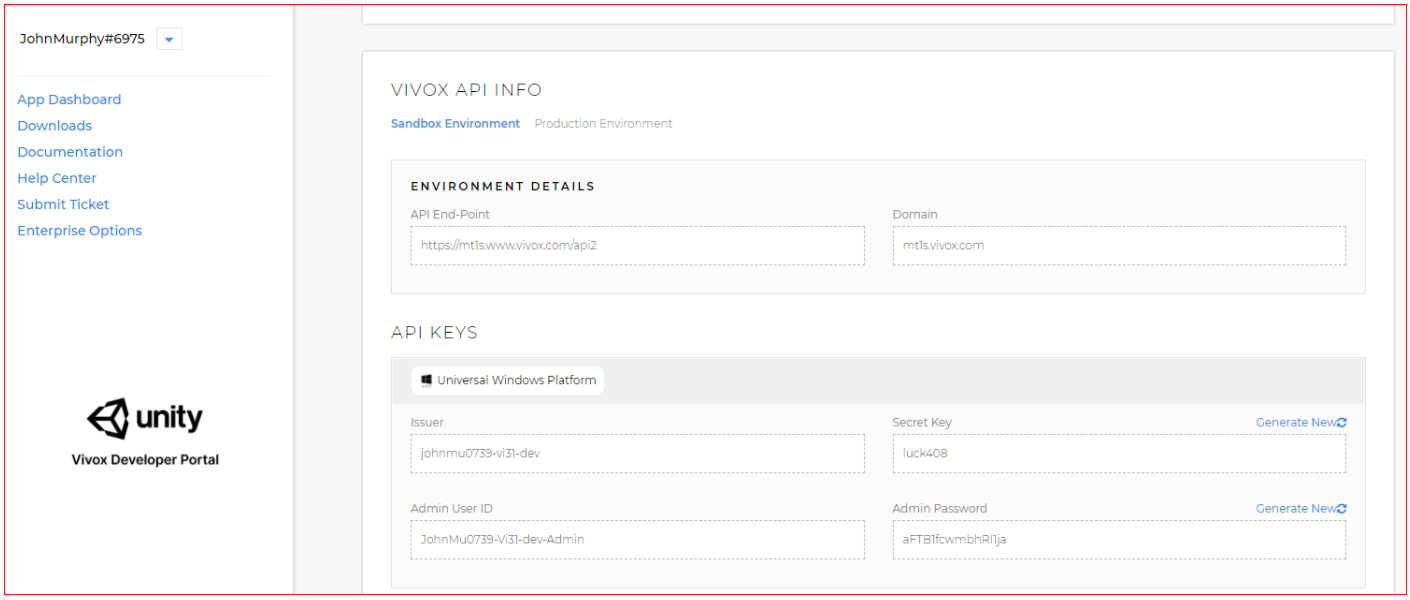
Setting Up Your Vivox Credentials

In **EasySession.cs** script you must hardcode your credentials. This is used for development purposes. Vivox recommends not hardcoding your credentials in your application. You can implement custom logic to retrieve your credentials at runtime from your game server.

Your credentials should be placed in these variables

- **public static Uri APIEndpoint** { **get; set;** } = new **Uri** ("");
- **public static string Domain** { **get; set;** } = "";
- **public static string Issuer** { **get; set;** } = "";
- **public static string SecretKey** { **get; set;** } = "";

Vivox Developer Portal
Credentials Dashboard



EasyExample

EasyExample covers most functionality you will need for Vivox Voice Chat. You can edit this script and rename it without breaking any functionality.

You may notice **EasyExample** inherits **EasyManager**. Because of this inheritance you don't need to add an EasyManager to the project. The **[SerializedField]** properties are only applicable to the Demo Scene and serve as an example of how to incorporate a user Interface(**UI**) to get the player values need to send to Vivox. You can do all this thru code without user input if you want.

All the **public void** Methods in EasyExample are linked to button events or other UI events in the Canvas.

All the **public override** methods are inherited from EasyManager and overridden. They are all called when a **Vivox event fires** I left the base methods **[base.MethodNameHere();]** in as an example but you can delete them. There is no major functionality in the base methods, just a simple Unity **Debug.Log()** statement for each event that fires. Each override updates the **Text UI** in the Demo Scene and is not necessary just an example of what information may be relevant to common use cases.

You can Attach EasyExample to a gameObject and hook up your own UI to it. I would probably rename EasyExample to a something that fits your naming convention.

Certain events update numerous times a second and should be used wisely to avoid spamming and to help with performance.

Vivox Features You Can Access From EasyCodeForVivox

- **Login 1 person**
- **Join 1-10 Non-Positional Channels (Conference Channels)**
- **Join 1 3D Positional Channel (Counts)**
- **Maximum Joined Channels is 10 (9 Non-Positional, 1 Positional)**
- **Send Channel Messages**
- **Send Direct Messages as long you know the User's Name**
- **Toggle Voice/Audio in channel**
- **Toggle Text in channel**
- **Adjust Local Players Volume**
- **Adjust Remote Player's Volume if you know their name**
- **Mute Self**
- **Mute other players in channel if you know their name**
- **Text-To-Speech(TTS) - All of Vivox TTS options available**

Features Supported By Vivox but not available in EasyCodeForVivox

- **Presence is not added because their is currently a bug in Vivox SDK**
- **IOS/Mac/Linux support is not added or confirmed (Vivox does support IOS/Mac/Linux but EasyCodeForVivox doesn't)**
- **Logging in multiple users**
- **No client/server architecture**
- **No Web Server API support**
- **No IsTyping Callback**

PlayMaker Extension

You need to purchase PlayMaker to use this part of the Extension

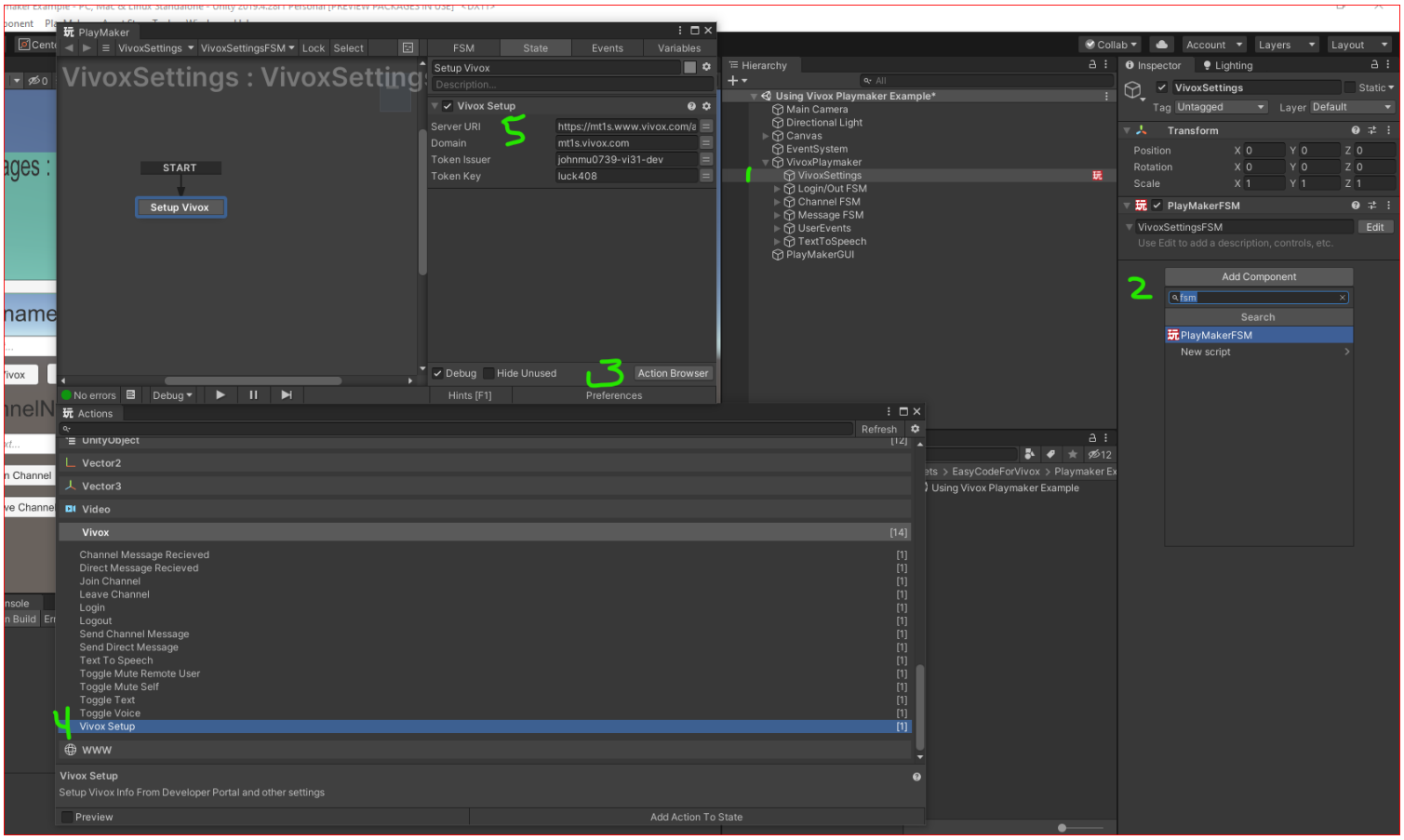
This Extension is in Beta and is being released for Testing. **Please backup your project before using! Or start a new project to experiment with!**

Open the Demo scene in the **Playmaker Extension (Beta)** folder to try it out and experiment. Would love to hear any feedback you have.

You can email me with questions or comments at **fortheone.dev@gmail.com**
or Join the Discord I run and leave feedback there https://discord.gg/3kqbp6hgCE

EasyCodeForVivox - PlayMaker Extension

Step 1 : Setup your Vivox Credentials



1. Add an empty gameobject to you scene
2. Add an PlayMakerFSM component to your empty gameobject
3. Click **Edit** in the PlayerMakerFSM you created on the empty game object and then click the **Action Browser** at the bottom of the PlayMaker editor.
4. Search for the **Vivox** category and add the **VivoxSetup** Action to your **PlayMakerFSM**. Make sure you are in the **State** tab.
5. In the **State** tab add your Vivox Credentials from the Credentials Dashboard in the Vivox Developer Portal