Voice Chat Interface

An overview of the Voice Chat Interface

Voice communication between players is a common feature that provides a quick and way easy to communicate with others. However, the specific implementation of voice chat varies between different online services and platforms. With the **Voice Chat Interface**, you can easily operate voice communications over a variety of different services without having to write custom code for each one, including **Epic Online Services** (EOS), Vivox, and modern gaming consoles. The Voice Chat Interface provides an agnostic API that supports all the features needed to integrate voice chat into your product: joining voice channels, selecting input and output devices, muting or blocking other users, and transmitting and receiving audio data.

If you plan to use EOS as a voice communication provider, read the EOS Voice Chat Plugin documentation for specific information about configuring it to work with your product.

User Management

The Voice Chat Interface identifies individual users by instantiating one **Voice Chat User Interface** for each of them. This separation enables the system to support multiple local users, as well as make distinctions between system-wide and per-user behaviors and settings.

Creating and Destroying

Since voice communication occurs between users, you must create at least one local **Voice Chat User** with the `CreateUser` function. If you are using a service or platform that supports multiple users on the same system, you can call `CreateUser` more than once, and each call will return a new `IVoiceChatUser` interface. Your Voice Chat User Interface instance performs all voice chat interactions specific to an individual user. Remember to call `ReleaseUser` on each Voice Chat User Interface when you no longer need it. See the following reference subsection for more specific detail.

Code Flow Reference

To create your own IVoiceChat and IVoiceChatUser instances, follow this lifecycle. Begin by creating and initializing a single IVoiceChat instance, and using it to create one IVoiceChatUser instance for each local user (or player). With the IVoiceChatUser instance, the user can freely enter and leave any trusted server channels manually, and can communicate with other users. When you are ready to end voice communication, generally during application shutdown, uninitialize any instances you created. This flow involves the following functions, which must be called in order:

Function	Usage
`IVoiceChat::Initialize`	Initialize an IVoiceChat instance. You must call this on any instance you create before using it.
`IVoiceChat::Connect`	Connect to the pre-configured voice server.

Function	Usage
`IVoiceChat::CreateUser`	Create an IVoiceChatUser interface instance. Each local user that intends to join a channel needs its own instance to log in. Unlike the other functions in this flow, you can call `CreateUser` at any time.
`IVoiceChatUser::Login`	Log a local user into the voice chat server. The `PlayerName` parameter is the EOS Product ID of the user, stringified with the `EOS_ProductUserId_ToString` API and converted to an FString with UTF8_TO_TCHAR or similar. It is not required that the EOS Product Id in question actually be logged in via the connect interface of the platform instance used by EOSVoiceChat, just that it is a valid EOS Product Id. The required auth is provided by the trusted server in the form of channel join credentials which are then passed to `IVoiceChatUser::JoinChannel`.
`IVoiceChatUser::JoinChannel`	Join a channel after logging into the server. The user can join multiple channels at a time, and can leave and rejoin a channel; each channel will continue to follow the flow individually from this point forward. This process requires credentials; See the section on joining a channel below for information about obtaining them.
`IVoiceChatUser::LeaveChannel`	(Optional) Cause the local user to leave a channel that they previously joined with `JoinChannel`.
`IVoiceChatUser::Logout`	(Optional) Log a local user out of the voice chat server. Doing this will also cause the user to leave all channels. After logging out, you can call `Login` to log back in without having to create a new interface instance.
`IVoiceChat::Disconnect`	(Optional) Disconnect from the voice server. This will also effectively log out and leave any channels the user has joined.
`IVoiceChat::Uninitialize`	Shut down the IVoiceChat interface instance. Calling this function also leaves any channels the user has joined, logs

Function	Usage
	out, and disconnects from the voice chat server. This is usually called during application shutdown.

Configuring User Settings

Voice chat services may have settings that you can configure on a per-user basis. The specific settings and acceptable values vary from one service to another, but the Voice Chat Interface wraps this functionality with the `SetSetting` and `GetSetting` functions. Both of these functions use `FString` as the key and value type.

Blocking and Muting

Use `BlockPlayers` and `UnblockPlayers` with a list of player names to block or unblock, respectively, all the players on the list. If you want to mute or unmute a player use `SetPlayerMuted`. You can check to see whether or not a specific player is muted with `IsPlayerMuted`.

Managing Audio Devices

Managing a user's audio input and output devices is straightforward. You can query the devices on the user's system if you want to give the user options about which device to use, or select a different input or output device from the default. You can also adjust the volume of a device, including the option to mute it.

Identifying and Selecting Devices

Use `GetInputDeviceInfo` and `GetOutputDeviceInfo` to identify the current input and output devices, respectively. You can also call `GetDefaultInputDeviceInfo` and `GetDefaultOutputDeviceInfo` to get the identity of the default devices, or `GetAvailableInputDeviceInfos` and `GetAvailableOutputDeviceInfos` to get a listing of all available devices. Once you know what device you want to use, call `SetInputDeviceId` or `SetOutputDeviceId` to select it.

Changing Volume and Muting

You can use `SetAudioInputVolume` and `SetAudioOutputVolume` to adjust the volume of your devices, or `GetAudioInputVolume` and `GetAudioOutputVolume` to check the current values. Similarly, `SetAudioInputDeviceMuted` and `SetAudioOutputDeviceMuted` can mute (or unmute) your devices, while `GetAudioInputDeviceMuted` and `GetAudioOutputDeviceMuted` report whether or not your devices are currently muted.

Communicating with Other Users

Voice communication between users requires that they are all logged in, and that the Voice Chat Interface itself is connected to a voice chat server.

Joining and Leaving

There are three layers of connection involved in communication with other users. The Voice Chat Interface must be connected to a voice communication server, the Voice Chat User Interface must log in, and finally, the Voice Chat User interface must join a channel with other users.

Begin by calling `Login` on your Voice Chat User, followed by the Voice Chat Interface `Connect` function, and wait for your `FOnVoiceChatConnectCompleteDelegate` callback function to confirm that you have successfully connected. You can also check on the status of your connection at any time by calling `IsConnected`. If you are not currently connected, `IsConnecting` will inform you of whether or not you are attempting to connect at that moment.

While connected, you can query the server's channel listing with `GetChannels`, and use the `JoinChannel` and `LeaveChannel` functions to participate in different group conversations.

When you are finished, disconnect from the server by calling `Disconnect`. This will disconnect the Voice Chat Interface from the server, which will stop all local users from communicating. If you only want to log a specific user out, call the `Logout` function on that user's Voice Chat User Interface. Before shutting down, you should disconnect and log each user out.

Functions that affect the entire local system, like connecting to or disconnecting from a server, operate on the Voice Chat Interface. Operations that relate to only one local player, such as joining or leaving a channel, or interacting with audio hardware, are part of the Voice Chat User Interface.

Transmitting

The system will transmit a user's audio input to other users through the channels that the user has joined. You can set the user to transmit to all channels that they are currently in with `TransmitToAllChannels`, or to a specific channel with `TransmitToSpecificChannel`. Use `TransmitToNoChannels` to cease all channel transmissions.