Setting Up Your Sound Card for Digital Communications

14 August 2019

Setting Up Your Sound Card for Digital Communications

Table of Contents

On Application Implementation of Audio Support (both Windows and Mac OS)

When supporting audio devices, application developers have several choices:

1. Implement the application with full dependence upon default device selection and not provide the user with any ability to select the device.

2. Implement the application with full dependence upon default device selection, but provide the user with any ability to select the device.

3. Provide a full implementation that properly explores the device information that is exposed by the operating system, and include user interface updates based on devices being attached or detached after the application has launched.

The most common implementation is #2. This option is easy for the application developer to implement, requiring very little code. But this option is not without its limitations, which are sometimes exacerbated by the use of frameworks that are intended to make software development easier. The result is application / user interface misbehavior that the user must determine how to resolve. In some cases, such as when an application does not implement support for device attachment and detachment notifications, it may be necessary to quit and re-launch the application after attaching or removing a device. In other cases, a device may have to be removed in order to enable selection of the desired devices.

Imposing work arounds on the end user was not the intent of establishing plug and play devices. Operating systems and device specifications were generated with the goal of making it seamless for the user to attach and use a device. But a failure of application developers to have their applications fail to register with the operating system for device attachment and detachment notifications, and a failure of device manufacturers to properly adhere to the spirit of device specifications, coupled with operating system behavior that imposes the 1960's style audio device redirection that is inherited from a 3.5mm headphone jack that cut off the speakers when a headphone was attached, has lead to some significant difficulties for the end user to understand how computer audio works and to resolve issue related to audio devices that could have been implemented better.

This article intends to provide guidance for the radio amateur so that they can enjoy a better operating experience when operating on digital modes, and to avoid issues that may be actionable by the FCC.

Users are encouraged to read the entire article, regardless whether Windows or Mac OS is used.

Default Operating System Behavior (both Windows and Mac OS)

In order to avoid having alert sounds (i.e. email notification sounds, text message sounds, VOIP / Skype notification sounds) from being transmitted over the radio, and to avoid having VOIP / Skype use the radio as a microphone source instead of the the computer's microphone, it is critical that the Sound hardware on the PC be properly configured.

When a USB audio device is attached to, both a Windows and a Mac OS computer will automatically configure the USB audio device as the:

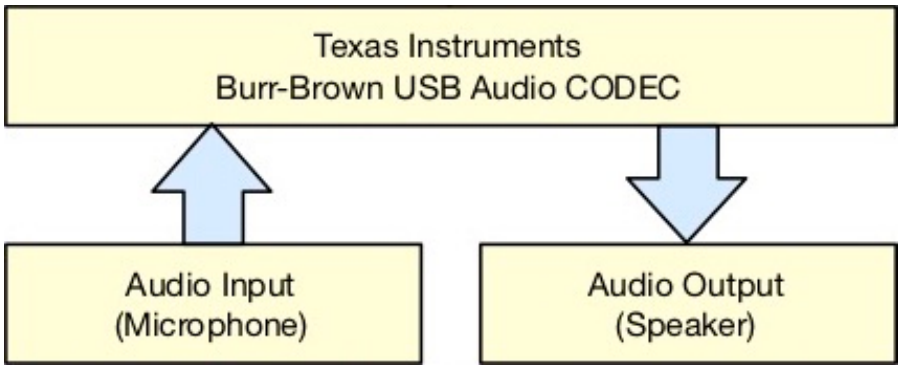
- Default playback audio device, where all sound played by the computer are routed to the USB audio device.
- Default playback communications audio device, where all sound generated by the VOIP applications, such as Skype, and intended to be heard by the user, are routed to the USB audio device.
- Default microphone audio device, where all sound recorded by the PC is routed from the USB audio device.
- Default communications microphone audio device, where all sounds recorded by VOIP applications, such as Skype, and intended to carry the users voice, source audio from the USB audio device.

Depending on the audio content being inadvertently played to the transceiver/transmitter, this can leave the operator in violation of FCC regulations.

USB Audio Devices use a USB Composite Device interface, that encapsulates a USB Audio Streaming Interfaces for both Input and Output, and a USB Audio Control interface, which describes the signal topology within the device, and is shared by both the input and output streaming interfaces. It is the streaming interfaces that are depicted in the Windows user interface as Sound devices.

All USB audio devices, whether built-in to the transceiver (i.e. as a Kenwood TS-590SG or similar transceiver with built-in USB Audio), or externally interfacing the transceiver (e.g. an external USB Audio Device such as a SignaLink USB), present themselves to the operating system as described above.

Attaching the USB Audio Device to the PC is subject to the above described behavior.



Although this operating system behavior is appropriate when attaching a USB headset, and emulates the behavior that is common for 1/4 inch and 3.5mm jacks, this operating system behavior is not appropriate when attaching a transceiver to the PC. Windows does provide manual control to override this default behavior, and steps must be taken to ensure that the USB Audio Interface is only available to those software applications (e.g. FLDIG and the NBEMS suite, WinLink, JS8Call, WSJT-X, etc.), that are intended to use the transceiver.

While performing this configuration, we'll also be setting the USB sound device levels.

IMPORTANT

If the USB Audio Device's USB cable is disconnected and then reconnected to the PC, and where reconnection occurs to a different USB connector, you will need to perform the Sound configuration procedure again. Windows will preserve the settings for the same USB audio device attached to the same USB connector, but moving the USB audio device to another USB connector is treated as s different device and will require reconfiguration of the Sound devices.

For this reason, it is advisable to not disconnect the USB Audio Device interface to the transceiver after completing the configuration steps in his tutorial.

IMPORTANT

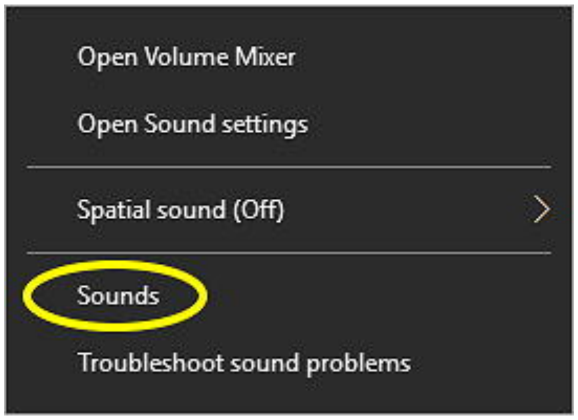
Digital communications presents a transmitter duty cycle from 80% to 100%. Your transmitter is not designed to operate at these duty cycles. Further, any ALC activity, including ALC activity that is not necessarily displayed on the ALC meter, will introduce distortion on your transmitted signal that will reduce the ability of receving stations to copy your station's transmissions. Many radio manuals advocate controlling the power level with the power control, and while this procedure avoids expensive customer returns for the radio manufacturer, it does not result in minimum transmitted distortion and best operating results. The level settings (i.e. the combined USB Audio Device level settings and the trannsceiver USB Audio Device Level settings) should be adjusted, starting at a minimum value (to avoid transmitter damage), to achieve no more than 50% of the transmitter rated output with the transmitter power control set at maximum.

Windows Sound Devices Configuration

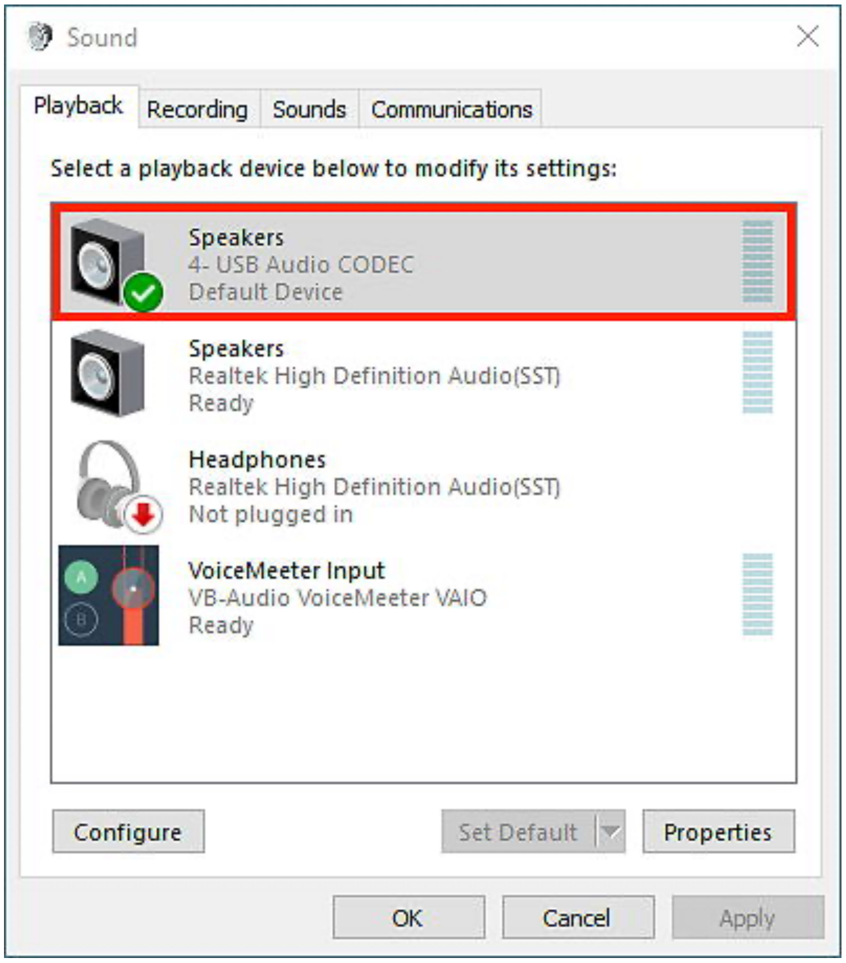
The steps necessary to properly configure the Sound support in Windows, with the USB Audio Device transceiver interface attached to the PC, are as follows:

- Attach the USB cable from the transceiver USB Audio Device to the PC.
- Right-click on the speaker icon on the Task Bar to bring up a contextual menu.

- Select the **Sounds** menu item.



- With the **Playback** tab view selected, you will see that the USB Audio CODEC indicates that it is the Default Device. You will need to set another device as the Playback Default Device.



Select a playback device below to modify its settings:

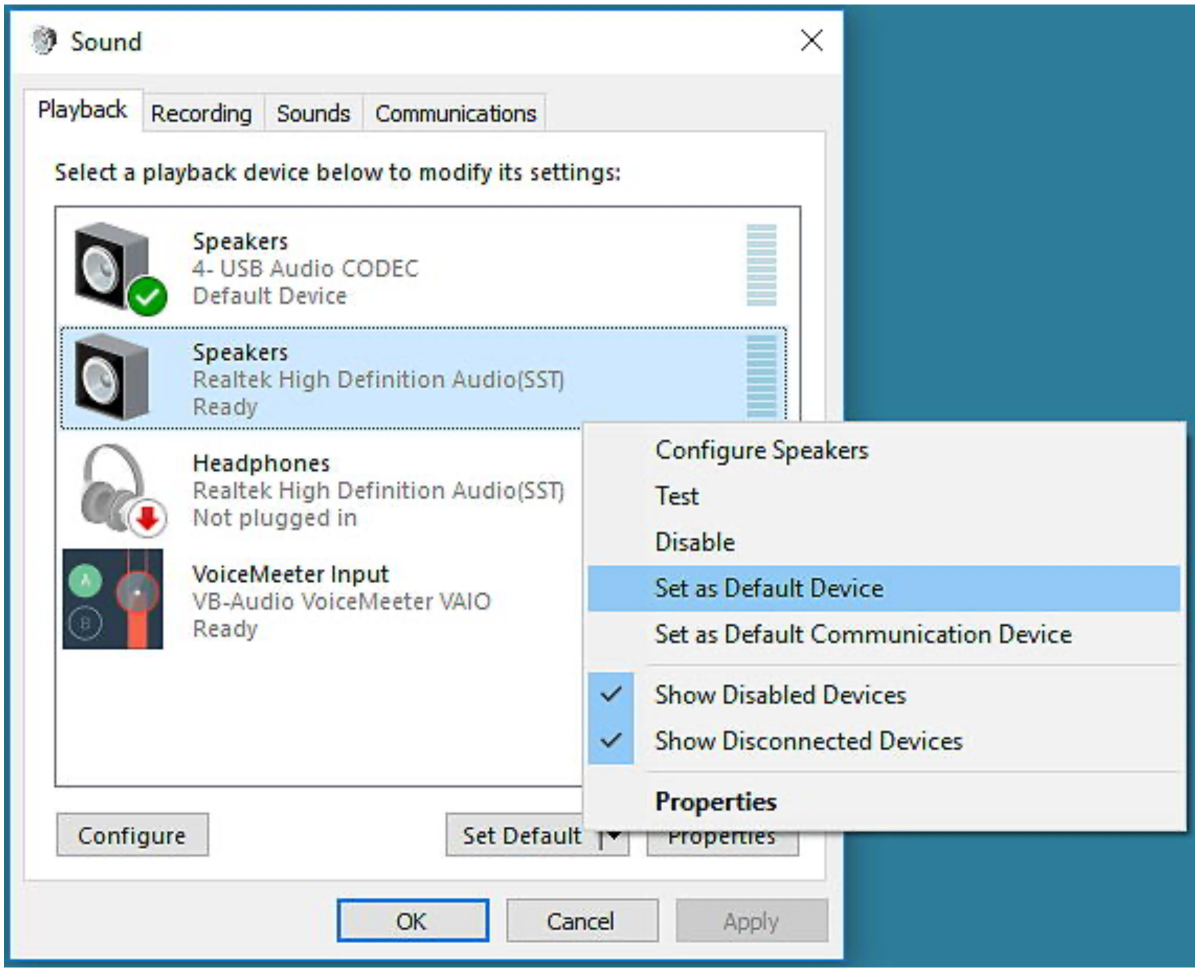
Speakers
4- USB Audio CODEC
Default Device

Speakers
Realtek High Definition Audio(SST)
Ready

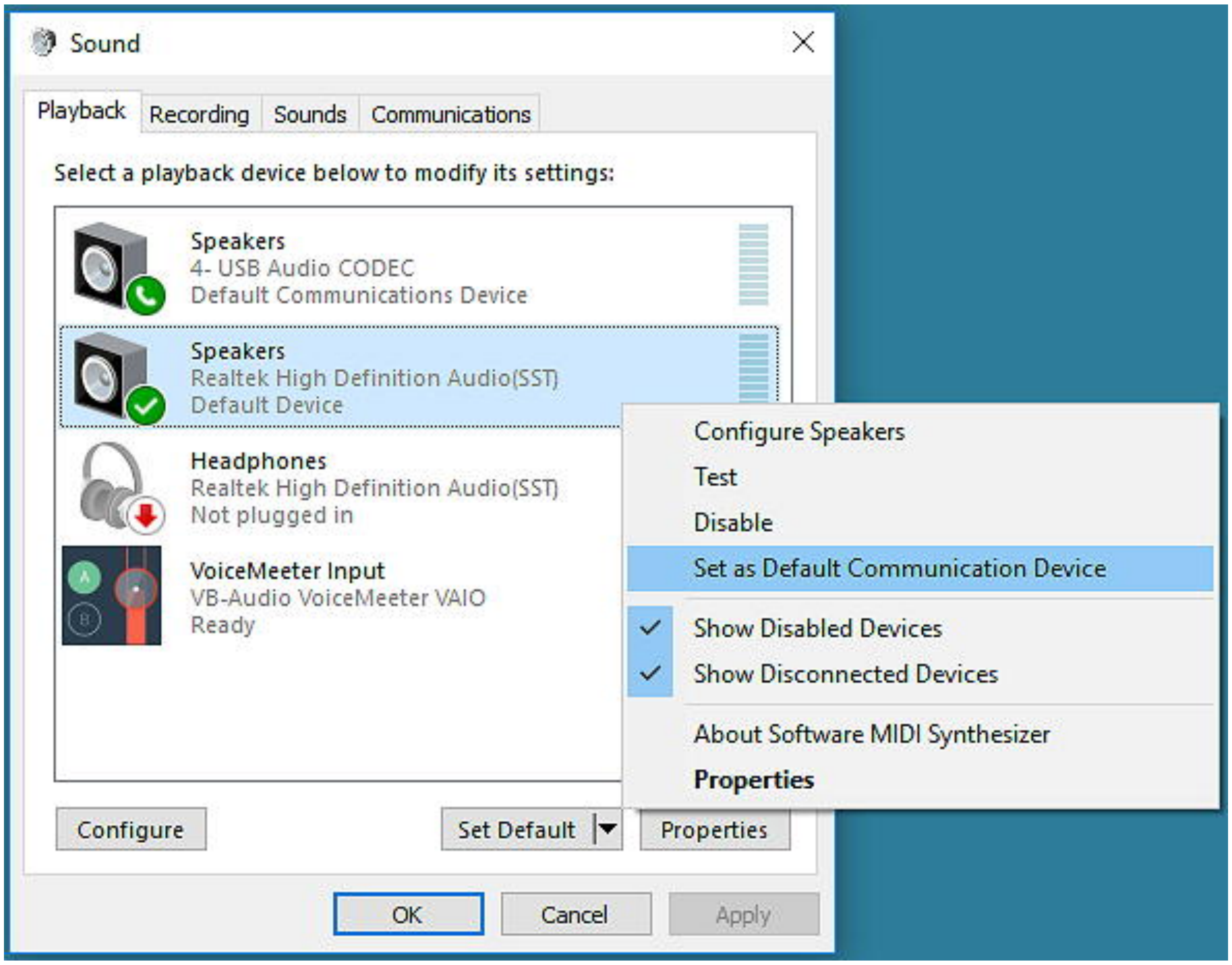
Headphones
Realtek High Definition Audio(SST)
Not plugged in

VoiceMeeter Input
VB-Audio VoiceMeeter VAIO
Ready

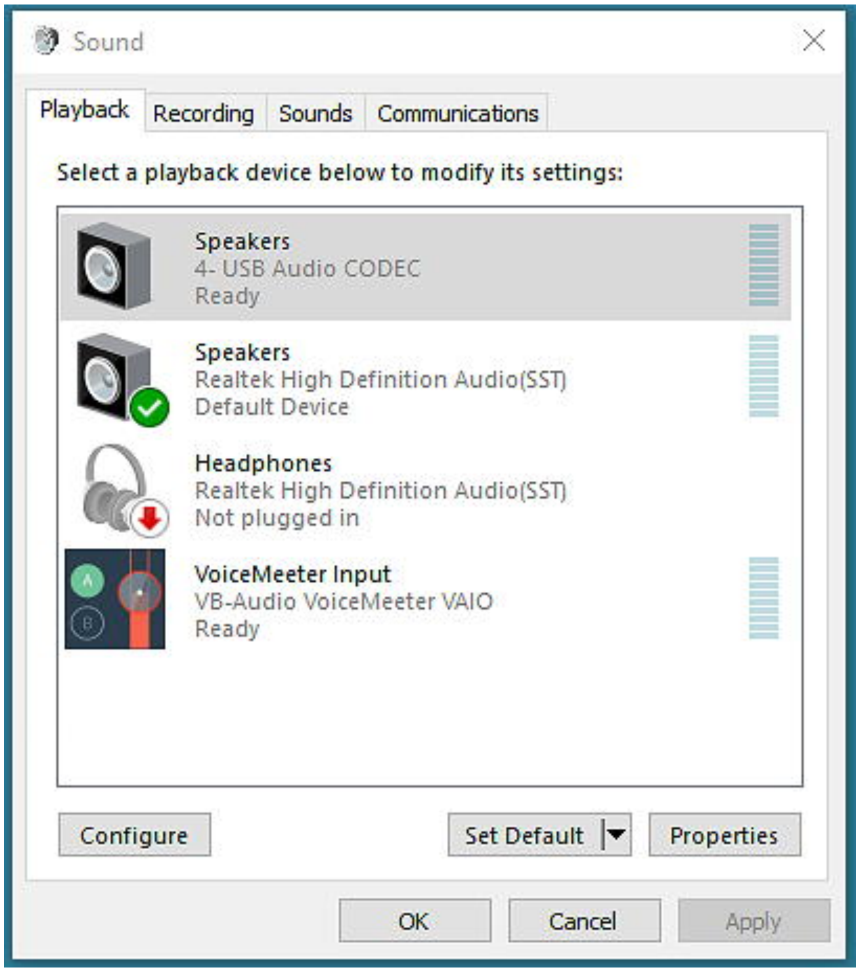
- Locate another Playback audio device, which does not indicate **Not Plugged In**, and right-click on that device to bring up a contextual menu. In this example, the Realtek High Definition Audio Speakers are used. When the contextual menu is displayed, select the **Set as Default Device** menu item.



- On the same Playback audio device used in the previous step, right-click on the device to bring up the contextual menu. When the contextual menu is displayed, select the **Set as Default Communications Device** menu item.


Sound
Playback
Recording
Sounds
Communications
Select a playback device below to modify its settings:
Speakers
4- USB Audio CODEC
Default Communications Device
Speakers
Realtek High Definition Audio(SST)
Default Device
Headphones
Realtek High Definition Audio(SST)
Not plugged in
VoiceMeeter Input
VB-Audio VoiceMeeter VAIO
Ready
Configure Speakers
Test
Disable
Set as Default Communication Device
Show Disabled Devices
Show Disconnected Devices
About Software MIDI Synthesizer
Properties
Configure
Set Default
Properties
OK
Cancel
Apply

- Single-click on the USB Audio Codec Speakers device that is the transceiver USB Audio Output Device.



Select a playback device below to modify its settings:

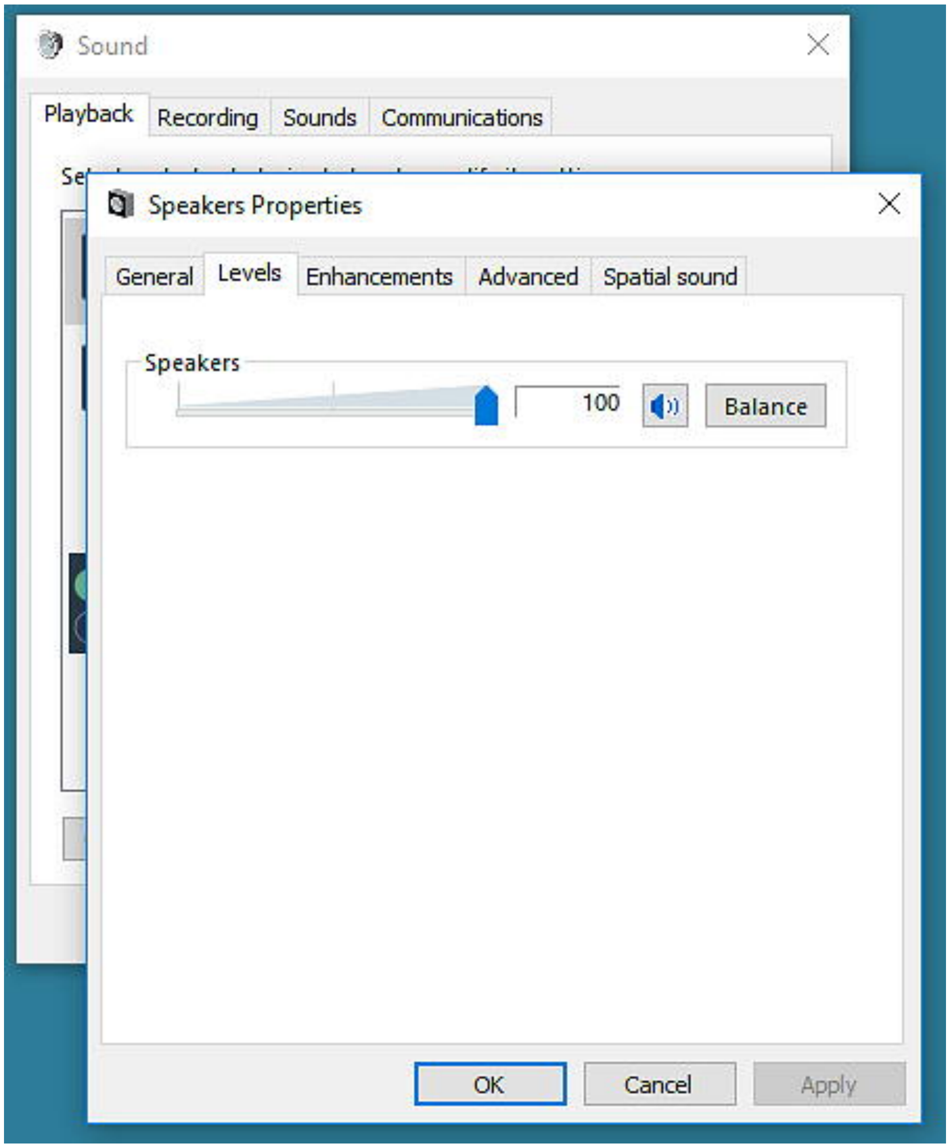
Speakers
4- USB Audio CODEC
Ready

Speakers
Realtek High Definition Audio(SST)
Default Device

Headphones
Realtek High Definition Audio(SST)
Not plugged in

VoiceMeeter Input
VB-Audio VoiceMeeter VAIO
Ready

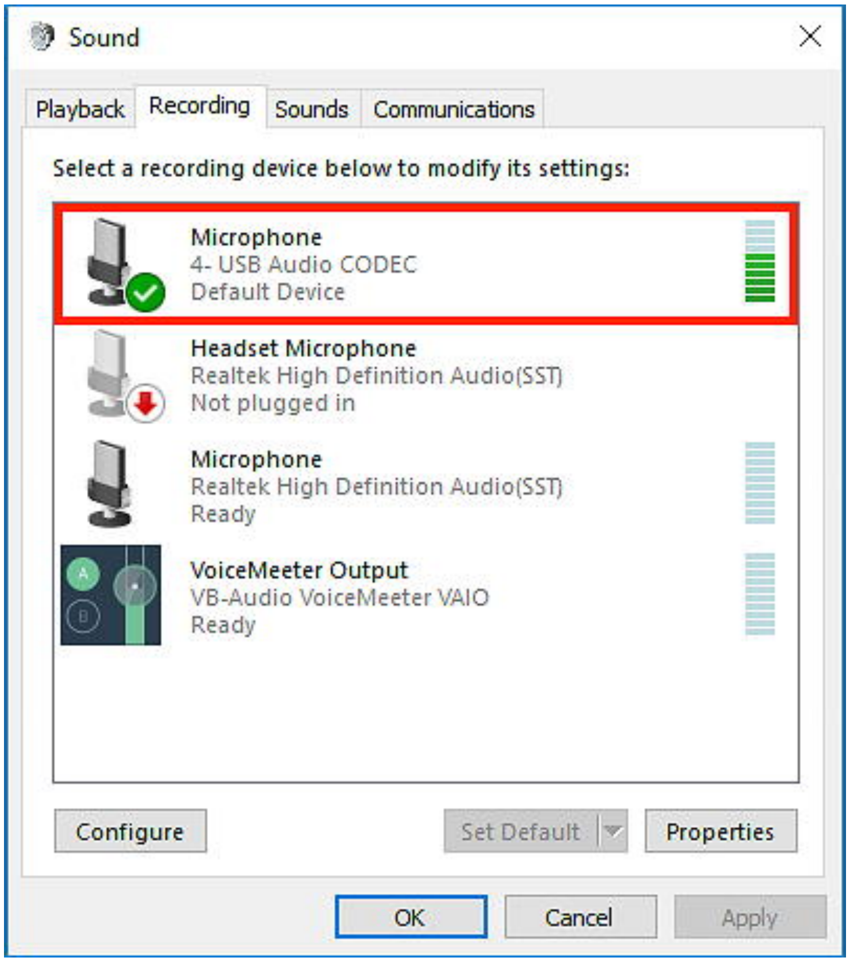
- Click on the **Properties** button and select the **Levels** tab view. If the USB Audio Interface is built-in to the transceiver, set the level to 100 so that the menus within the transceiver have the full range of control. If using an external USB Audio Device that has level control knobs (e.g. SignaLink or similar), set the level to 100 so that the full range of control is available. If you are using an external (not built in to the transceiver) USB Audio Device interface that has no knobs for level control, you will need to set the level control as appropriate for your transceiver.



Speakers 100

- Click the **OK** button to close the Speakers Properties window.

- With the Recording tab view selected, you will see that the USB Audio CODEC indicates that it is the Default Device. You will need to set another device as the Playback Default Device.



Sound

Select a recording device below to modify its settings:

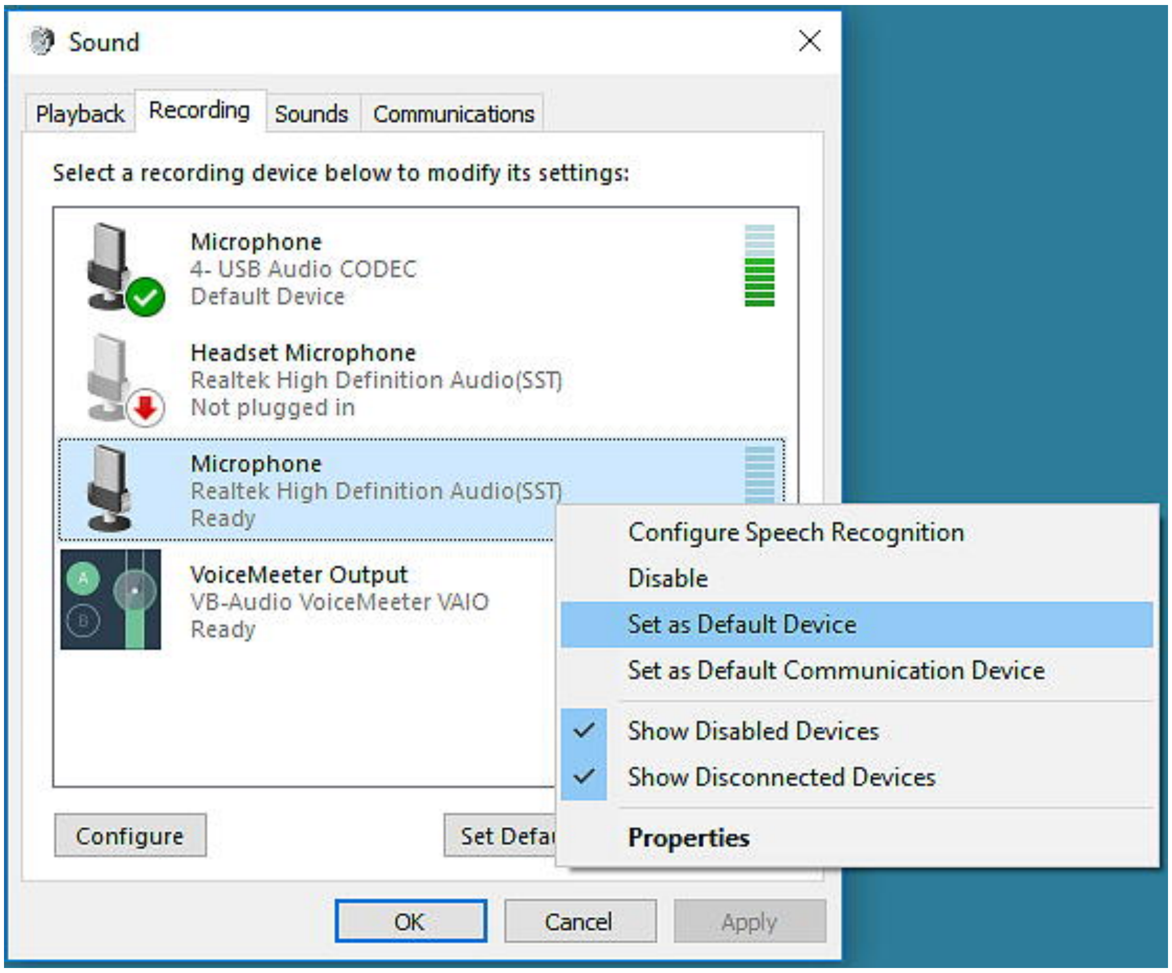
Microphone
4- USB Audio CODEC
Default Device

Headset Microphone
Realtek High Definition Audio(SST)
Not plugged in

Microphone
Realtek High Definition Audio(SST)
Ready

VoiceMeeter Output
VB-Audio VoiceMeeter VAIO
Ready

- Locate another **Recording** audio device, which does not indicate **Not Plugged In**, and right-click on that device to bring up a contextual menu. In this example, the Realtek High Definition Audio Microphone is used. When the contextual menu is displayed, select the **Set as Default Device** menu item.



Select a recording device below to modify its settings:

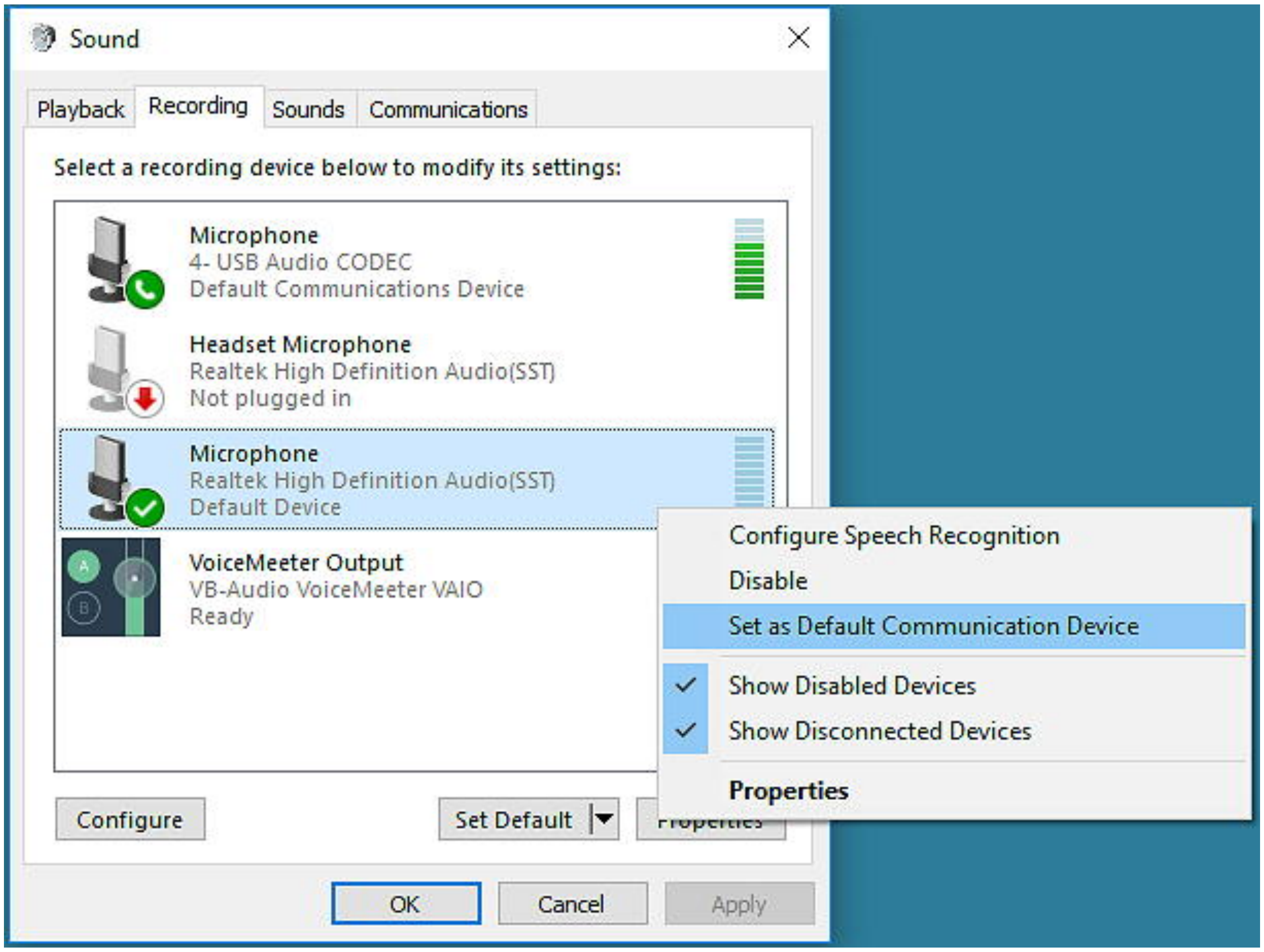
Microphone
4- USB Audio CODEC
Default Device

Headset Microphone
Realtek High Definition Audio(SST)
Not plugged in

Microphone
Realtek High Definition Audio(SST)
Ready

VoiceMeeter Output
VB-Audio VoiceMeeter VAIO
Ready

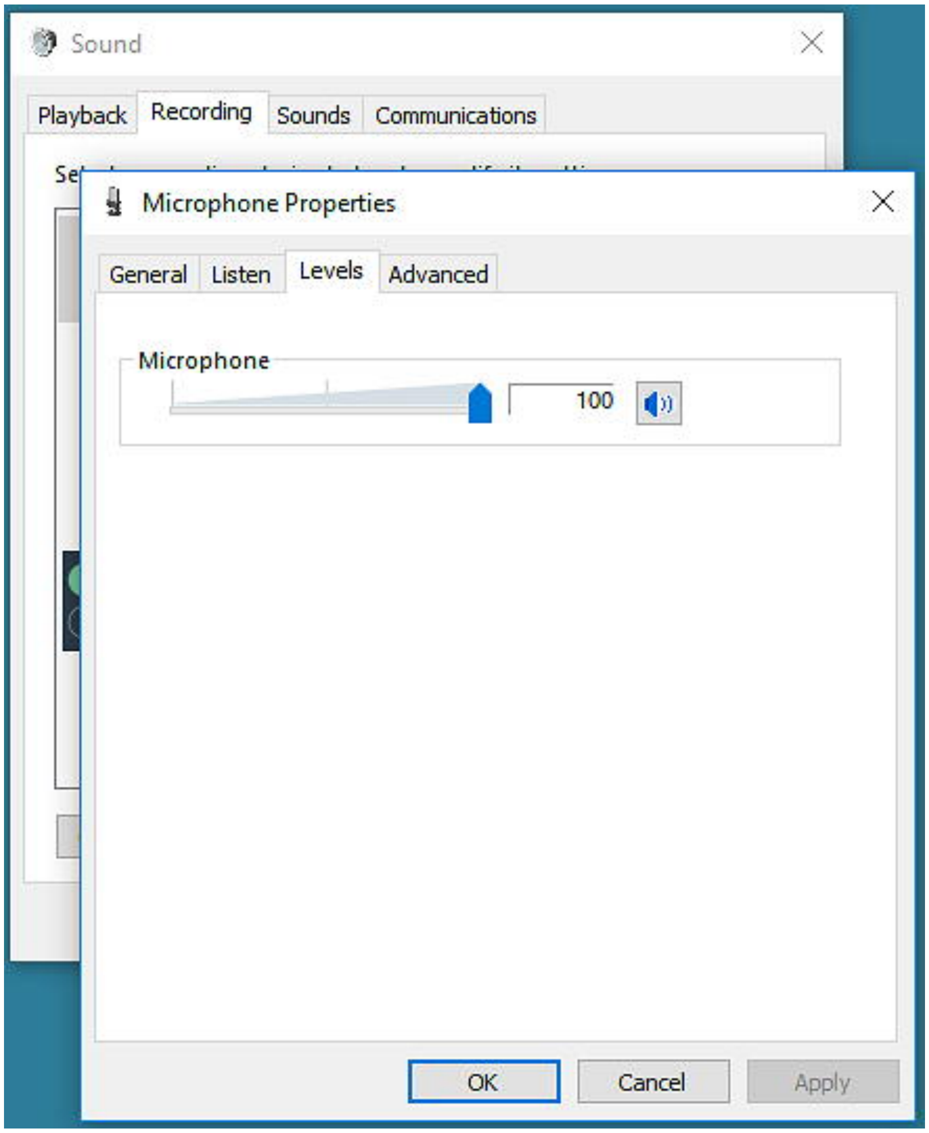
- On the same Recording audio device used in the previous step, right-click on the device to bring up the contextual menu. When the contextual menu is displayed, select the **Set as Default Communications Device** menu item.



Sound
Playback
Recording
Sounds
Communications
Select a recording device below to modify its settings:
Microphone
4- USB Audio CODEC
Default Communications Device
Headset Microphone
Realtek High Definition Audio(SST)
Not plugged in
Microphone
Realtek High Definition Audio(SST)
Default Device
VoiceMeeter Output
VB-Audio VoiceMeeter VAIO
Ready
Configure Speech Recognition
Disable
Set as Default Communication Device
Show Disabled Devices
Show Disconnected Devices
Properties
Configure
Set Default
OK
Cancel
Apply

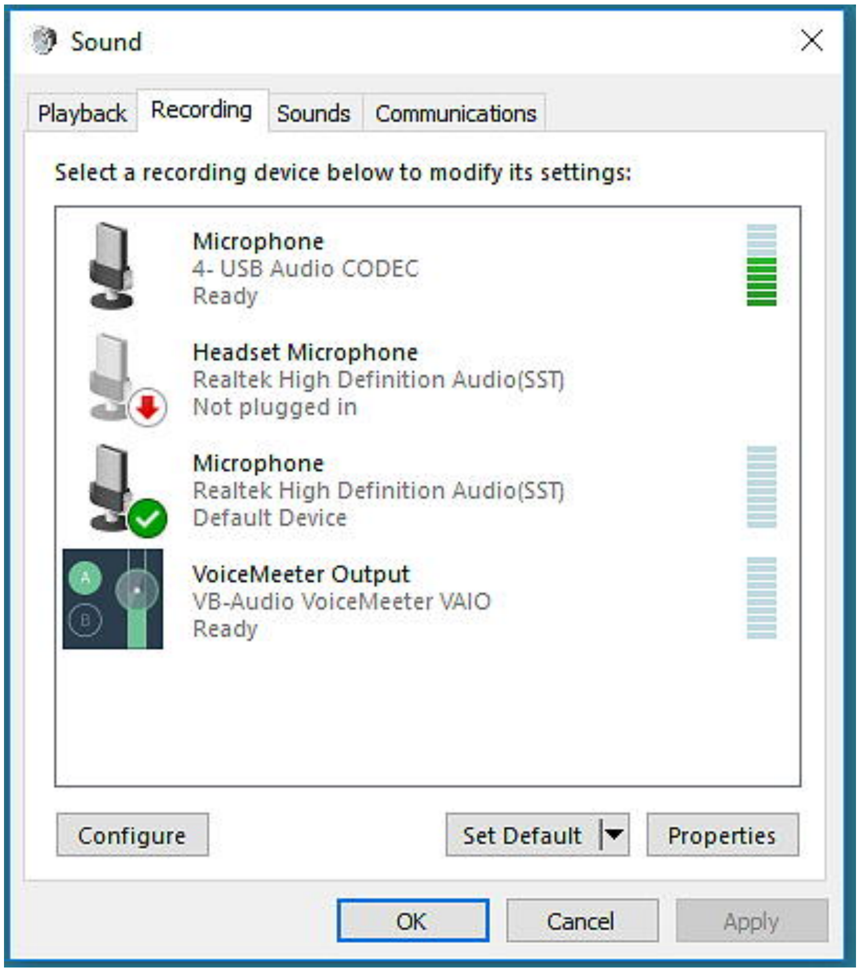
- Single-click on the USB Audio Codec Microphone device that is the transceiver USB Audio input Device.

- Click on the **Properties** button and select the **Levels** tab view. Set the Microphone and set the levels to 100.



Microphone 100

- Click the **OK** button to close the **Microphone Properties** window.



Select a recording device below to modify its settings:

Microphone
4- USB Audio CODEC
Ready

Headset Microphone
Realtek High Definition Audio(SST)
Not plugged in

Microphone
Realtek High Definition Audio(SST)
Default Device

VoiceMeeter Output
VB-Audio VoiceMeeter VAIO
Ready

- Click the **OK** button to close the **Sound** window.

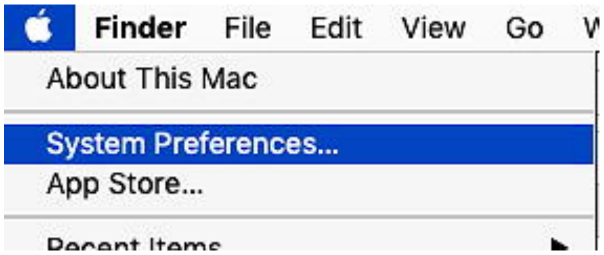
MAC OS Sound Devices Configuration

Mac OS has very similar issues to that of Windows. Upon attaching a USB Audio Device that has an endpoint that describes a speaker, Mac OS selects that device as the default audio device for output. Similar issues occur with input devices.

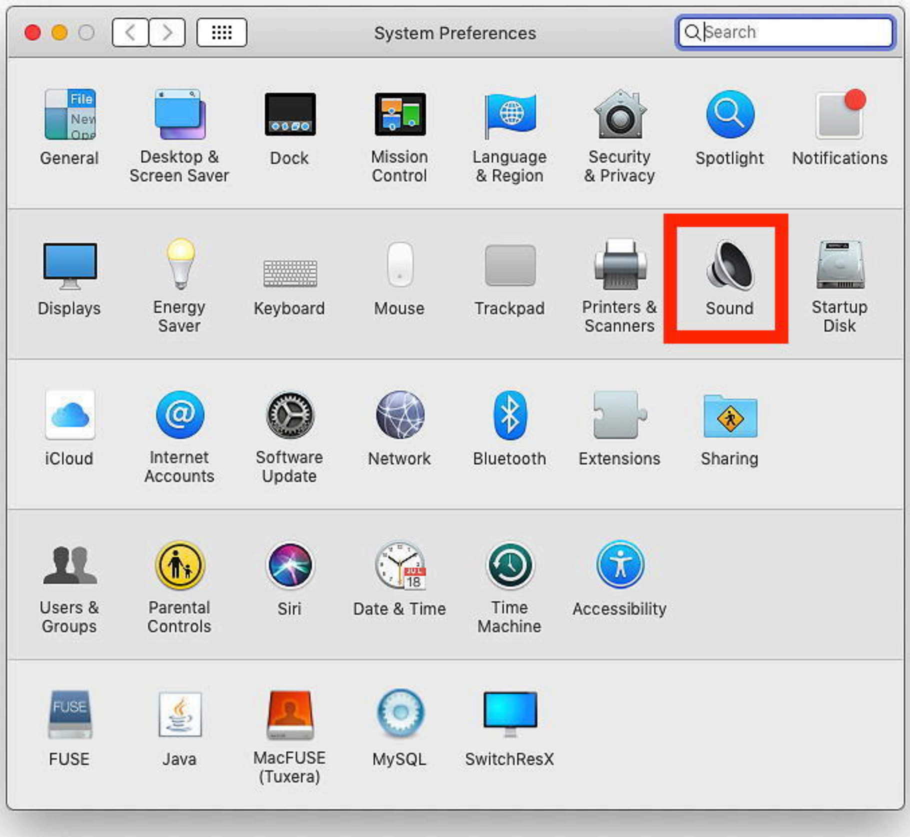
Like Windows, this is not desirable in that audio that is not intended to be transmitted, or audio, that if transmitted, could result in a violation of FCC regulations, could be imposed upon the end user by this operating system behavior.

To resolve this issue on Mac OS, after attaching the USB Audio Device (i.e. transceiver with a built-in USB Audio Device, or radio sound card USB Audio Device, such as the SignaLink USB), you will need to perform the following steps to configure the audio support to avoid such issues:

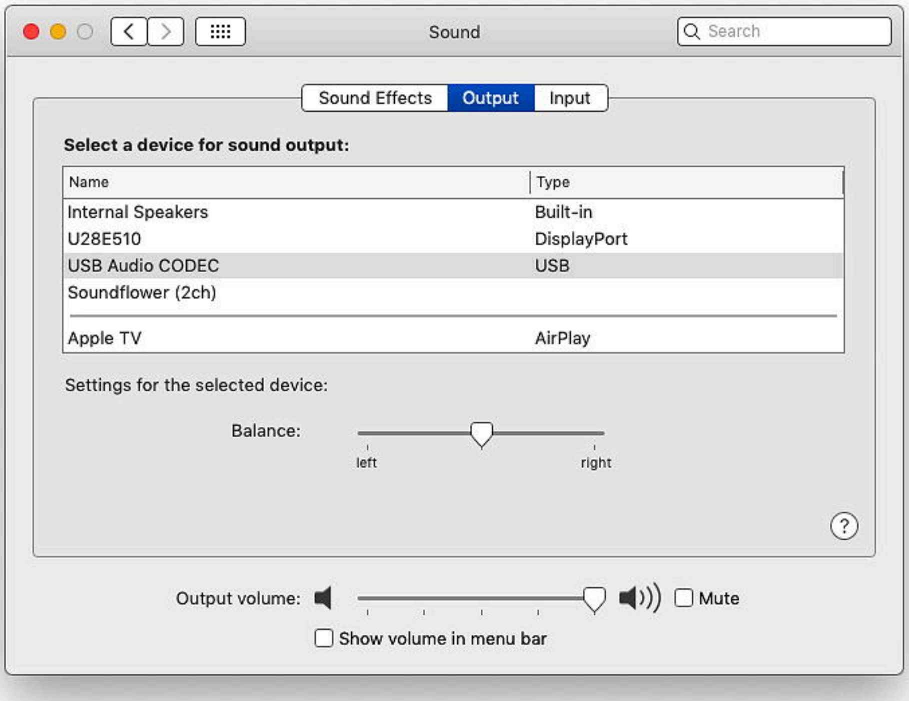
- Navigate to the Apple menu and select **System Preferences**.



- In the **System Preferences** window, select the **Sound** icon.



- In the **Sound** pane, select the **Output** tab view.

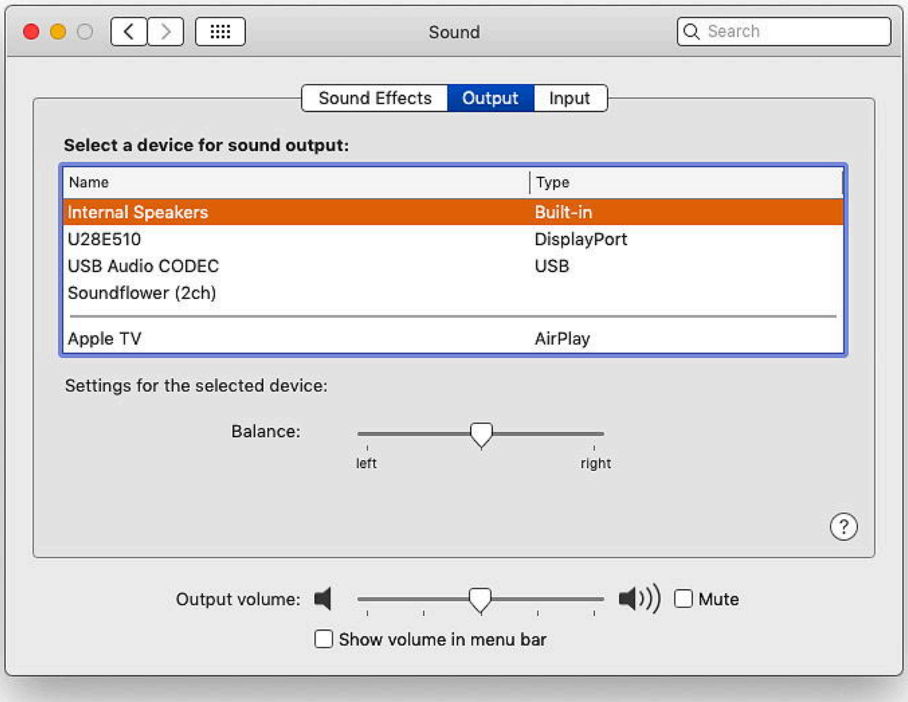


Select a device for sound output:

Name	Type
Internal Speakers	Built-in
U28E510	DisplayPort
USB Audio CODEC	USB
Soundflower (2ch)	
Apple TV	AirPlay

Settings for the selected device:

- Select the output device that you wish to have audio routed to for all no radio digital communications applications (i.e. Mail sounds, Messages sounds, FaceTime Sounds, Skype sounds, iTunes sounds, etc.). Typically, this will be Internal Speakers, Headphones, or an external device that is attached via the DisplayPort connector.



Select a device for sound output:

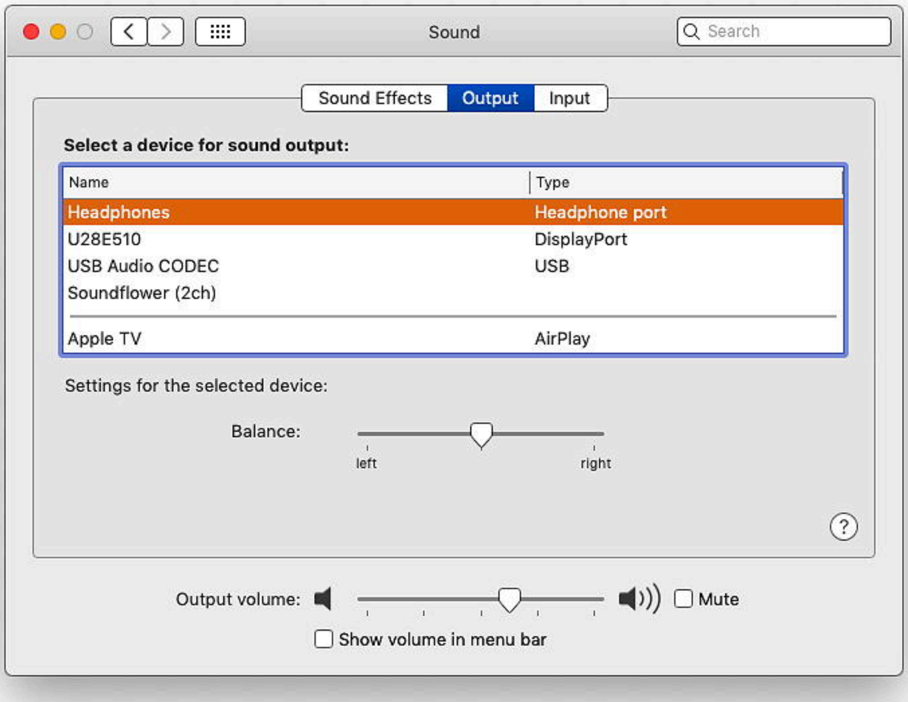
Name	Type
Internal Speakers	Built-in
U28E510	DisplayPort
USB Audio CODEC	USB
Soundflower (2ch)	
Apple TV	AirPlay

Settings for the selected device:

Balance: left right

Output volume: Mute

Show volume in menu bar



Select a device for sound output:

Name	Type
Headphones	Headphone port
U28E510	DisplayPort
USB Audio CODEC	USB
Soundflower (2ch)	
Apple TV	AirPlay

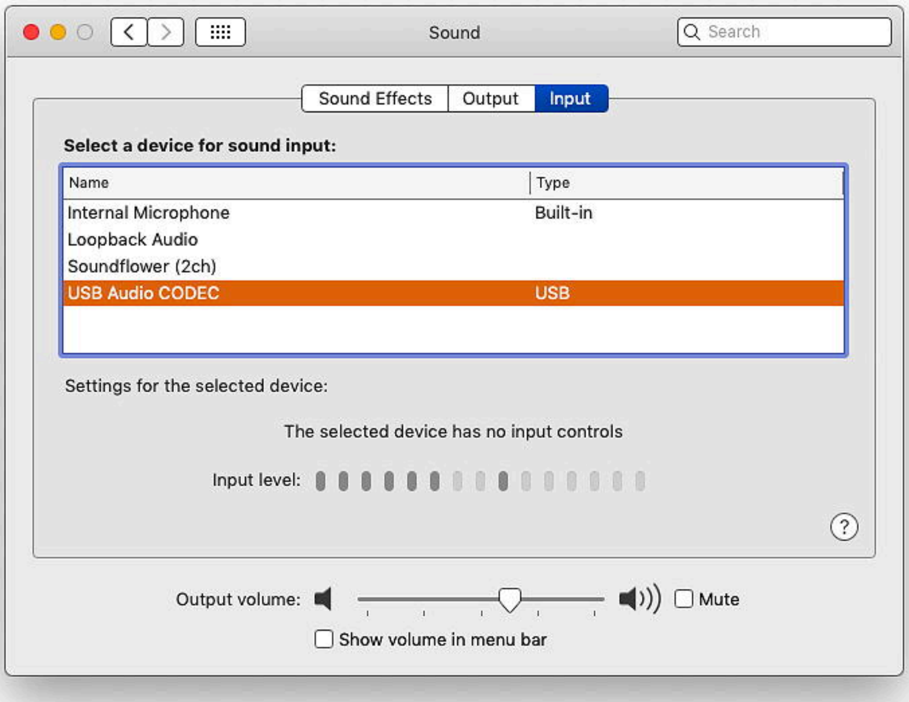
Settings for the selected device:

Balance: left right

Output volume: Mute

Show volume in menu bar

- In the **Sound** pane, select the **Input** tab view.



Select a device for sound input:

Name	Type
Internal Microphone	Built-in
Loopback Audio	
Soundflower (2ch)	
USB Audio CODEC	USB

Settings for the selected device:

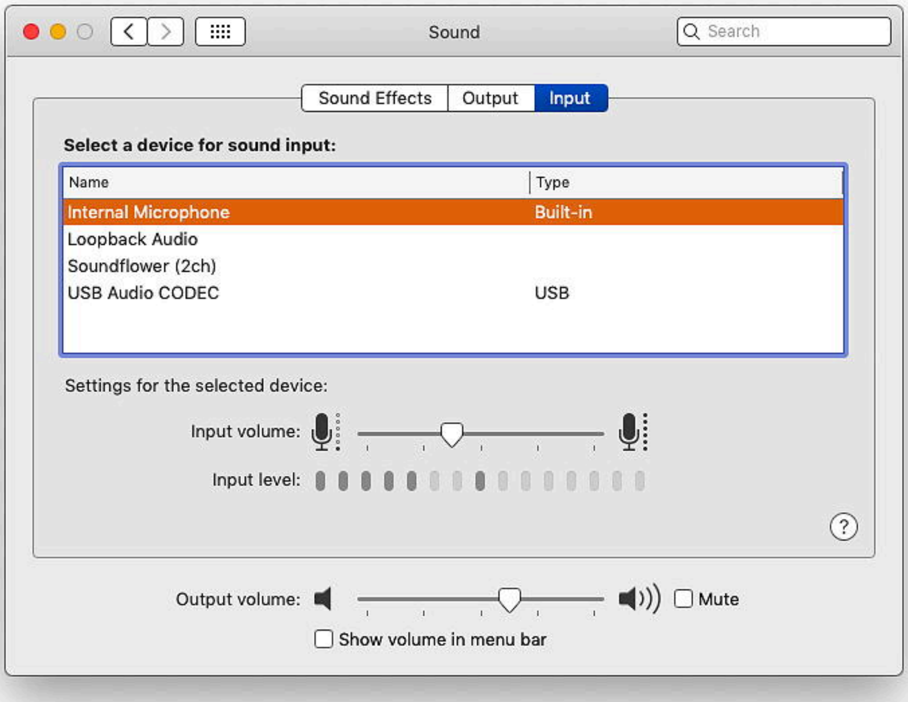
The selected device has no input controls

Input level:

Output volume: ☐ Mute

☐ Show volume in menu bar

- Select the audio device that you wish to use for recording (i.e. Skype, FaceTime, Garage Band, etc.). Typically, this will be the Internal Microphone.



Select a device for sound input:

Name	Type
Internal Microphone	Built-in
Loopback Audio	
Soundflower (2ch)	
USB Audio CODEC	USB

Settings for the selected device:

Input volume:

Input level:

Output volume:

Mute

Show volume in menu bar

- Close **System Preferences**.

After Configuring the Operating System Default Audio Devices

After completion of configuring the operating system default audio devices, launch what ever digital communications applications you use and then use the user-interface in that application to select the audio device that you wish to use with that application.

Hot Microphones

Current transceiver design, with built-in USB Audio Device interfaces, internally route audio only from the source that is keying the radio. For radios with this design paradigm, the microphone is not hot while the transceiver is keyed and transmitting through the USB Audio Device interface. For early transceivers that have a built-in USB Audio Device Interface, or for transceivers using an external USB Audio Device interface, such as the SignaLink USB, the microphone is most certainly hot while transmitting digital communications via the USB Audio Interface.

A number of radio operators are not aware that their microphone is hot while transmitting digitally. There are countless cases of hearing business calls and arguments between spouses by stations transmitting digital data, and particularly on digital modes that have long key down times, such as JT-65.

Many digital modes consume very little radio spectrum. Microphone audio, on the other hand, can be several kilohertz wide, and can reduce communications capabilities of other stations due to the broader band interference caused by a hot mic.

It is vitally important that, if your transceiver presents a hot microphone when transmitting digital communications, you must disconnect your microphone to avoid causing interference (or possibly embarrassing transmissions).

To test if the microphone is hot, reduce the audio drive (either by menu on the transceiver with a built-in USB Audio Device interface or by turning down the TX level knob on an external USB Audio Device such as the SignaLink). Then key the radio from the digital communications software (use the Tune button if available) and tap on your microphone while monitoring the transmitter output power level. If you observe fluctuation of the transmitter output while tapping on the microphone, then your microphone is HOT during digital communications and you must disconnect the microphone during digital communications.

Difficulties Presented by Number of USB Audio Devices

There are other issues, related to the implementation of USB Audio Devices by manufacturers of radios, radio sound cards, audio mixers, and a host of other audio related products, that present issues for the radio amateur who wishes to use more than one USB Audio Device at the same time.

Each USB device includes a Device Descriptor, which contains attributes to uniquely identify the USB device. Among these are a Vendor ID, Product ID Manufacturer String and Product String. These attributes are intended to be used to uniquely identify the device, for the purposes of loading the appropriate driver for the device, and to uniquely represent the device for user interface elements involving device selection.

The Vendor ID and Product ID are used by the operating system to load the appropriate driver for the device when a vendor and product specific driver is required. For standard devices, such as a USB Audio Device, which does not require a vendor and product specific driver, the device class and subclass fields inform the operating system as to which driver to load.

The Manufacturer String and Product String are used to present the device in the user interface where device selection needs to be supported.

The issue is that the majority of USB Audio Devices identify themselves simply as **USB Audio Codec**. This is due to the device manufacturers (i.e. the radio manufacturers, radio sound card manufacturers, etc.) purchasing silicon from the integrated circuit manufacturers without ordering with a masking option to implement a vendor ID, device ID and product string that would uniquely identify the end product that is purchased by a consumer. Since customer specific masking was not ordered, all of these device then revert to a default USB device descriptor. The result is that all of these consumer devices are then identified as being the exact same thing.

Here is an example of a TigerTronics SignaLink USB Radio Sound Device:

```
Device Descriptor
    Descriptor Version Number:   0x0110
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x2904
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 Burr-Brown from TI
    Product String:              2 USB Audio CODEC
    Serial Number String:        0 (none)
```

Here is an example of the Kenwood TS-590SG Transceiver:

```
Device Descriptor
    Descriptor Version Number:   0x0200
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x29B3
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 Burr-Brown from TI
    Product String:              2 USB Audio CODEC
    Serial Number String:        0 (none)
```

Here is an example of the Icom IC-F8101 Transceiver:

```
Device Descriptor
    Descriptor Version Number:   0x0110
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x2901
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 Burr-Brown from TI
    Product String:              2 USB Audio CODEC
    Serial Number String:        0 (none)
```

Here is an example of the Behringer XENYX 1204 USB Audio Mixer:

```
Device Descriptor
    Descriptor Version Number:   0x0110
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x2902
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 Burr-Brown from TI
    Product String:              2 USB Audio CODEC
    Serial Number String:        0 (none)
```

For all the products above, the Manufacturer String and Product String contents are identical. This results in one of two behaviors:

- All devices are listed in user interface elements with the exact same name (i.e. USB Audio CODEC), making it nearly impossible for the user to make a proper device selection in the user interface when more than one device of the same Product String is attached.
- In cases where the Product String is populated into an array (e.g. a menu list on Mac OS), the array excludes allowing entries for multiple devices of the same name, making only the first device discovered by the operating system available for user selection.

Similarly, the Serial Number String has also been left with a default value of zero, creating the exact same problem for a user attaching two of the same type of device (e.g. two SignaLink devices, as might be used for a VHF radio and a UHF radio).

The list of devices on the market that exhibit this common data in the USB Device Descriptor is endless.

This issue is wide-spread. All of the device manufacturers blame the operating system, and all of the operating system vendors blame the device manufacturers. The truth is that the entire issue would not exist if the device manufacturers paid for the masking charge and had their own Vendor ID, Product ID, Vendor String and Product String masked into the device.

Who wouldn't want their Kenwood TS-590SG to show up in the operating system as a Kenwood TS-590SG?

To fix this issue, it is only necessary for the device manufacturer to specify a USB Audio Device Descriptor that changes the Manufacturer and Product Strings. It is not necessary to fix the Vendor ID and Product ID values (although that might be nice).

Imagine a TigerTronics SignaLink USB Radio Sound Device USB Audio Device Descriptor that looked like:

```
Device Descriptor
    Descriptor Version Number:   0x0110
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x2904
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 TigerTronics
    Product String:              2 SignaLink
    Serial Number String:        0 (none)
```

Imagine a Kenwood TS-590SG Transceiver USB Audio Device Descriptor that looked like:

```
Device Descriptor
    Descriptor Version Number:   0x0200
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x29B3
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 Kenwood
    Product String:              2 TS-590SG
    Serial Number String:        0 (none)
```

Imagine an Icom IC-F8101 Transceiver USB Audio Device Descriptor that looked like:

```
Device Descriptor
    Descriptor Version Number:   0x0110
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x2901
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 Icom
    Product String:              2 IC-F8101
    Serial Number String:        0 (none)
```

Imagine a Behringer XENYX 1204 USB Audio Mixer USB Audio Device Descriptor that looked like:

```
Device Descriptor
    Descriptor Version Number:   0x0110
    Device Class:                0   (Composite)
    Device Subclass:             0
    Device Protocol:             0
    Device MaxPacketSize:        8
    Device VendorID:             0x08BB   (Texas Instruments Japan)
    ProductID:                   0x2902
    Device Version Number:       0x0100
    Number of Configurations:    1
    Manufacturer String:         1 Behringer
    Product String:              2 XENYX 1204
    Serial Number String:        0 (none)
```

These imaginary device descriptors are exactly what was intended when USB was created, and is part of the basis for **Plug And Play** features. The failure of these devices to not live up to the goals of **Plug And Play** is not the fault of the silicon manufacturer or the operating system manufacturers, but is most likely due to the device manufacturers either being unaware that they should be masking the device descriptor or in the economics of avoiding the expenditure of doing so.

Probably the only way that this issue is going to get resolved is by customers applying pressure to the device manufacturers to have the silicon masked uniquely for their product. For all the difficulty this causes, having the consumer pay $1 more per device would certainly remove hundreds of dollars worth of headache.