Jack OS X

- an OS X implementation of Jack -

version 0.90

December 22, 2011

Table of Contents

What's New? [New]

Version 0.90 fixes a problem on OSX Lion systems where JackRouter component could not be selected as default device.

Version 0.89 fixes a problem with audio output from Flash-based movies, improves support for Avid/Digidesign soundcards and also adds support for ProTools 9.

Version 0.88 implements a new feature that allows MIDI communication between Jack and CoreMIDI – essentially, a Jack MIDI / CoreMIDI "bridge." Jack OS X communicates MIDI information with sample accuracy.

Version 0.87 fixes a problem with saving studio setups within JackPilot.

Version 0.86 addresses several issues in the CoreAudio driver, especially when used with Apogee cards. A new feature, Monitor Ports (input ports that loopback everything that is sent to the audio output and make it available as an input) has been added. See the new documentation about this feature in the How Do I Use Jack section. The Jackdmp server has also been updated to the latest version.

Version 0.85 addresses a problem with the Jack OS X plugins being closed after the main Jack server has been shut down. It also addresses a problem with some Digidesign cards, and updates the Jackdmp server to the latest version. In addition, additional documentation around the new device selection model (described below) has been added to the Setting Preferences section.

Version 0.84 fixes a number of issues and usability problems related to the management of CoreAudio devices, in particular the requirement of Jack to identify a single duplex (input and output) audio device to work with, as well as the interaction of other applications with the audio device(s) that Jack OS X too was interacting with. It also provides several server-related bug fixes, as well as a sleep/wake problem introduced on Snow Leopard (10.6).

The first issue, that of requiring a single duplex device, became problematic with the appearance of Intel Macs, on which the input and output devices are represented as separate devices. Up until now, this has required users of Jack OS X on Intel machines to create and use an Aggregate Device, which allowed Jack OS X to interact with what appeared to it to be a single device. This was confusing for new users, and also caused problems when sub-devices of the Aggregate had sample rate changes, etc...

Version 0.84 and subsequent versions do away with the requirement to use an Aggregate Device, and allow the user to specify distinct input and output devices within its Preferences. Internally the Jack server creates a "private" Aggregate Device automatically, which the end user never sees nor interacts with, and which is destroyed when the Jack server is stopped. Furthermore, should the two specified devices actually represent audio devices controlled by distinct clocks (e.g. a USB microphone as input device, and the Built-In Audio as output device), the user can have Jack OS X compensate for potential clock drift between the two devices.

The second issue relates to problems introduced when applications automatically impose their desired sample rates onto the audio device they are configured to work with. If Jack was already running and had previously set the device to a different sample rate, it would fail silently when these other applications were started, because of the device's sample rate change imposed by the other applications. To remedy this situation, we have implemented the “hog mode” feature, in which Jack OS X can be configured to “hog” the audio devices it is selected to use, and therefore prevent other applications from automatically changing the devices' sample rates. Furthermore, if hog mode is *not* enabled, and an application that changes the devices' sample rate is started, Jack will now warn the user, who can then restart the Jack server with the updated sample rate.

More information about the new capabilities are described in the **How Do I Use Jack? / Setting Preferences** section.

Starting with version 0.83, Jack OS X added Mac OS 10.6 (Snow Leopard) support, including 64 bit operation of the Jack server when using the “64/32 bit” version. **This version is for use on OS 10.6 only**. Why is it called “64/32 bit”? Since the current JackPilot code has problems running in 64 bits, it will always start the JackPilot application in 32 bit mode. JackPilot however will start the Jack *server* in 64 bit mode. This will allow Jack to be set as the “Default device” for the Mac (default devices on 10.6 require 64 bit operation, allowing apps like iTunes, etc.. that only use the Mac’s default device, to be routed through Jack).

This “64/32 bit” version may cause problems however if you are using Jack OS X with a 3rd party audio device (i.e. anything other than “Built-In Audio”) that does not yet have 64 bit drivers installed on your system (since the Jack server runs in 64 bit mode). In these instances, if you are running OS 10.6, you will be forced to use the “32 bit” version, and you will lose the ability to set Jack as the Mac’s “Default” device as above.

A “32 bit” package is also provided for users of pre-10.6 operating systems.

Please note that the Jack AU and Jack VST plugins are still 32 bit in **both** packages.

The full change log is listed in the Version History section at the end of this document. Enjoy!

What’s Jack?

Jack is a low-latency audio server, written originally for the GNU/Linux operating system, and now with OS X and Windows support. It can connect a number of different applications to a single hardware audio device; it also allows applications to send and receive audio among one another. Its clients can run in their own processes (ie. as normal applications), or they can run within the Jack server (ie. as a "plugin").

Jack is different from other audio server efforts in that it has been designed from the ground up to be suitable for professional audio work. This means that it focuses on two key areas: synchronous execution of all clients, and low latency operation.

Jack is the brainchild of Paul Davis; it now has developers from all over the world actively working on improving this open source project.

What's Jack OS X?

Jack OS X is an OS X implementation of Jack, and it installs everything necessary to take full advantage of Jack on OS X. (Jack OS X was originally called Jack Tools prior to being renamed). It includes:

- The Jack Server
  - The infrastructure you need in place to use Jack.
- The JackRouter *
  - The JackRouter is a CoreAudio "user space" driver that allows ***any*** OS X CoreAudio application to become a Jack client. (This was previously known as the JAS (Jack Audio Server) in the first release of Jack OS X, but was renamed to avoid confusion with the Jack server). It is also occasionally still referred to as the JAR in some parts of the documentation.
- The Jack Plugins *
  - Both AU and VST "Jack-aware" audio plugins are provided, which can further expand the limitless audio routing possibilities when using Jack.
- The JackPilot Application *
  - JackPilot offers an easy to use GUI interface that allows you to control the Jack server, and manage the audio connections between applications and/or plugins.
- The NetJack Module * ***(this modules is temporarily not installed, until it is fixed)***
  - NetJack is a JackPilot module that allows for low latency transmission of audio streams over a network, and incorporates these streams into the Jack environment.

* *OS X-specific Jack development efforts*

Jack OS X - System Architecture

It is important to understand some of the basic architectural principles of how Jack works within OS X.

The "core" of Jack revolves around the Jack Server. It sends and receives audio data between "Jack-enabled" applications, as well as with a hardware audio device. This hardware device can be either the built-in Mac audio, or any OS X-compatible audio interface. Jack can (currently) only communicate with a single hardware audio device, though it is important to note that Jack can "share" that device with other applications. More on this in a moment.

There are three distinct types of "Jack-enabled" applications:

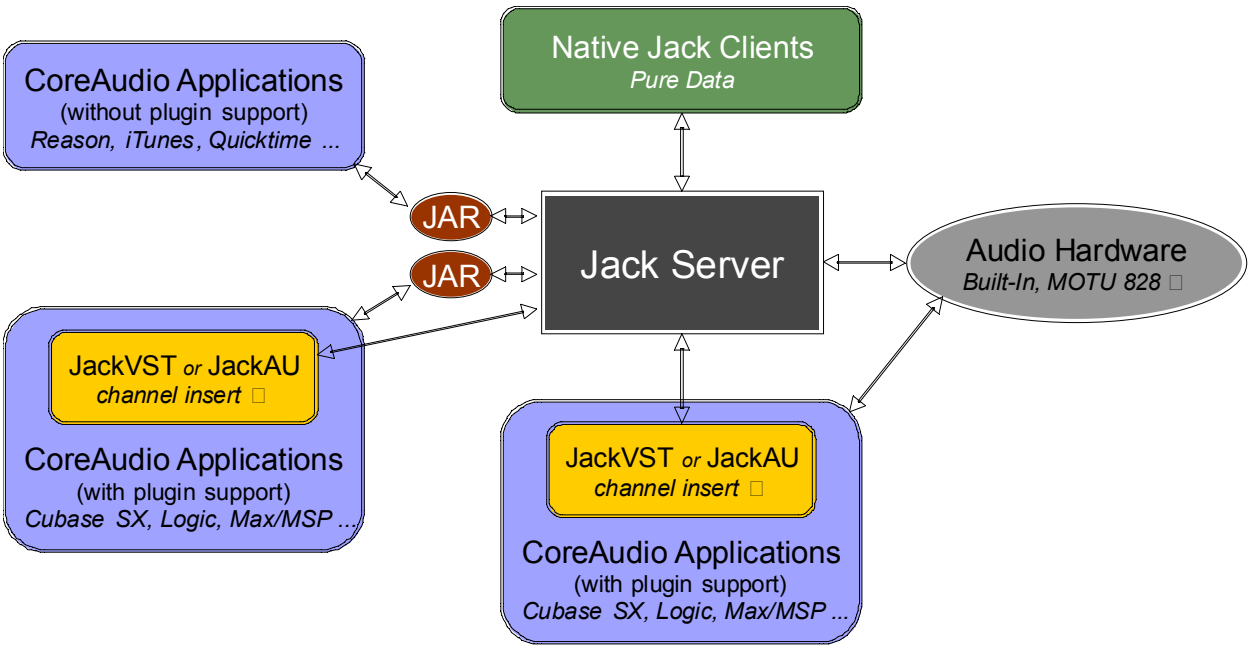
1. **Native Jack applications**. These have built in support for interfacing with a Jack server. There are relatively few of these applications currently available – an

example is Pd (also known as Pure Data) or Ardour. Recently (2009), Native Instruments have incorporated native Jack ability (initially in an unsupported fashion) in many of its applications.

2. **Jack-enabled Core Audio applications**. These applications use the JackRouter driver, or JAR, to communicate with the Jack Server. This is as simple as selecting the JackRouter driver in the application's audio I/O setup window (or for applications that just use the default OS X audio device, by selecting the JackRouter as that default "device"). Examples might include Cubase SX, Logic, Digital Performer, Reason, Live, Max/MSP, Peak, Spark, iTunes, RealPlayer, etc…

3. **Jack plugins used in any AU- or VST-compatible audio application**. You can use the Jack AU and/or VST plugins within any application that supports these plugin formats, to communicate with the Jack Server. This greatly expands the routing capabilities within Jack on OS X – for example, one could route the main audio from one application to a second application using Jack, but then route an insert feed on one channel to a third application, using a Jack plugin.

See the figure below for a graphical illustration of how these three types of Jack-enabled applications interact with the Jack Server and the JackRouter (JAR):



Understanding how Jack works at an even more fundamental level may be helpful for some users. ***Warning, the next three paragraphs are fairly technical and low level, and are not necessary to understand in order to use Jack OS X! They're provided purely for those who are interested.***

The Jack system is built around several components: a server, a driver and several clients. Since Jack clients will typically be separate applications, the system has to be

able to transfer data (like audio buffers) between different processes, activate them when needed, and possibly notify them when global state changes occur.

The Jack server is the center of the system. It interacts with the driver, and communicates with all registered clients. Triggered by the driver, the server activates the client graph, a set of connected "nodes", each of which must be "executed" on a periodic basis. In the case of Jack, the graph is made up of Jack clients, and each one has its own "audio process" function to be called in a specific order. The connections between each node may take any configuration whatsoever. Jack has to serialize the execution of each client so that the connections represented by the graph are honored (e.g. client A sends data to client B, so client A should execute before client B). In the event of feedback loops, there is no "correct" ordering of the graph, so Jack just picks one of the legal possibilities.

The whole graph is executed synchronously by a driver that interacts with the hardware, waking the server at regular intervals that are determined by its buffer size. The server then "distributes" this audio interrupt to all running clients. The basic requirement for the system's proper functioning is that the server and all clients do their jobs, including server / client communications, audio data transfer and processing, between two consecutive audio interrupts.

Beginning with version 0.75, Jack OS X uses a new version of the Jack server, called Jackdmp. With the emergence of multi-core machines, optimizing the use of available processors becomes more and more important. When several audio applications are running together in a Jack-like system, data dependencies between applications impose an activation order between them. If A is connected to B, and B connected to C, then A must be run first, then B and C. If on the contrary A is connected to B and C, A must be run first but then B and C could be run at the same time on two available processors. Depending on the connection topologies between applications, Jackdmp allows this kind of optimization by using a "data-flow" model that guarantees that applications are activated as soon as possible on any available processor. The global result is that available CPU is better used. Jackdmp also improves other parts of the system like avoiding audio glitches when connecting/disconnecting applications.

System Requirements

- OS X 10.4 or greater (including Snow Leopard, 10.6.x).

Installing Jack OS X

If you have already have a previous version of Jack OS X installed on your computer, the version 0.85 installer will automatically remove the older version before installing the new one.

- Download the Jack OS X version 0.85 compressed file to your hard drive.
- Double clicking this file should automatically decompress, and you should see a resulting file called **JackOSX.pkg.**

- Double click this file to start the installer.
- As part of the install process, you will be presented with a password dialog, in which you will need to provide the Administrator password for your Mac – enter it and click OK.
- Follow the instructions in the installer to install Jack OS X.
- You will need to restart your computer after the installer finishes its work, to complete the installation.

What gets installed where?

- /usr/local/bin
  - jack_connect
  - jack_disconnect
  - jack_load
  - jack_unload
  - jack_lsp
  - jack_metro
  - jack_netsource
  - jackd
  - jackdmp
- /usr/local/include/jack
  - intclient.h
  - jlist.h
  - jack.h
  - midiport.h
  - ringbuffer.h
  - session.h
  - statistics.h
  - thread.h
  - transport.h
  - types.h
  - control.h
  - systemdeps.h
  - weakjack.h
  - weakmacro.h
- /usr/local/lib
  - libjack.0.dylib
  - libjack.dylib
  - libjackserver.0.dylib
  - libjackserver.dylib
- /usr/local/lib/jackmp
  - jack_coreaudio.so
  - jack_coremidi.so
  - jack_net.so
  - jack_netone.so
  - netmanager.so
  - netadapter.so
  - audioadapter.so
- /usr/local/lib/pkgconfig
  - jack.pc
- /Library/Application Support/JackPilot/Modules
  - NetJack.jpmodule ***(this file temporarily not installed)***
- /Library/Audio/Plug-ins/HAL
  - JackRouter.plugin
- /Library/Audio/Plug-ins/Components
  - JACK-insert.component

- /Library/Audio/Plug-ins/VST
  - JACK-insert.vst
- /Library/Frameworks
  - Jackservermp.framework
  - Jackmp.framework
  - Jacknet.framework
  - Panda.framework
- /Applications/Jack
  - JackPilot.app
  - Documentation.pdf (this document)
  - Uninstall JackOSX.command
- /Applications/Jack/Extras/Developer
  - ReadMe.rtf
  - example-clients.xcodeproj
- /Applications/Jack/Extras/Developer/example-clients
  - connect.c
  - lsp.c
  - makefile
  - metro.c

Uninstalling Jack OS X

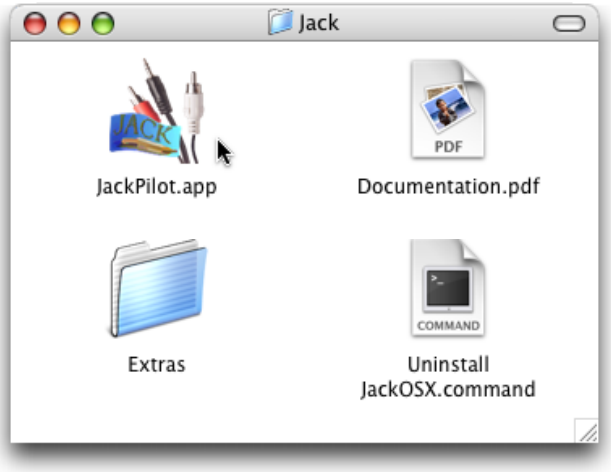
To uninstall Jack OS X, double click the “Uninstall JackOSX” file located in the Jack folder, in your main Applications folder. A Terminal window will be displayed, and it will prompt you for your Mac’s Administrator password. Type it, press Return, and all components of Jack OS X will be removed from your system. Please note that when typing your password into the Terminal window, it will not seem as though your password is being acknowledged; this is not the case – just keep typing and press Return, and it will work.

If you’d like to uninstall a previous version of Jack OS X, this is done automatically for you when installing the new version.

How do I use Jack?

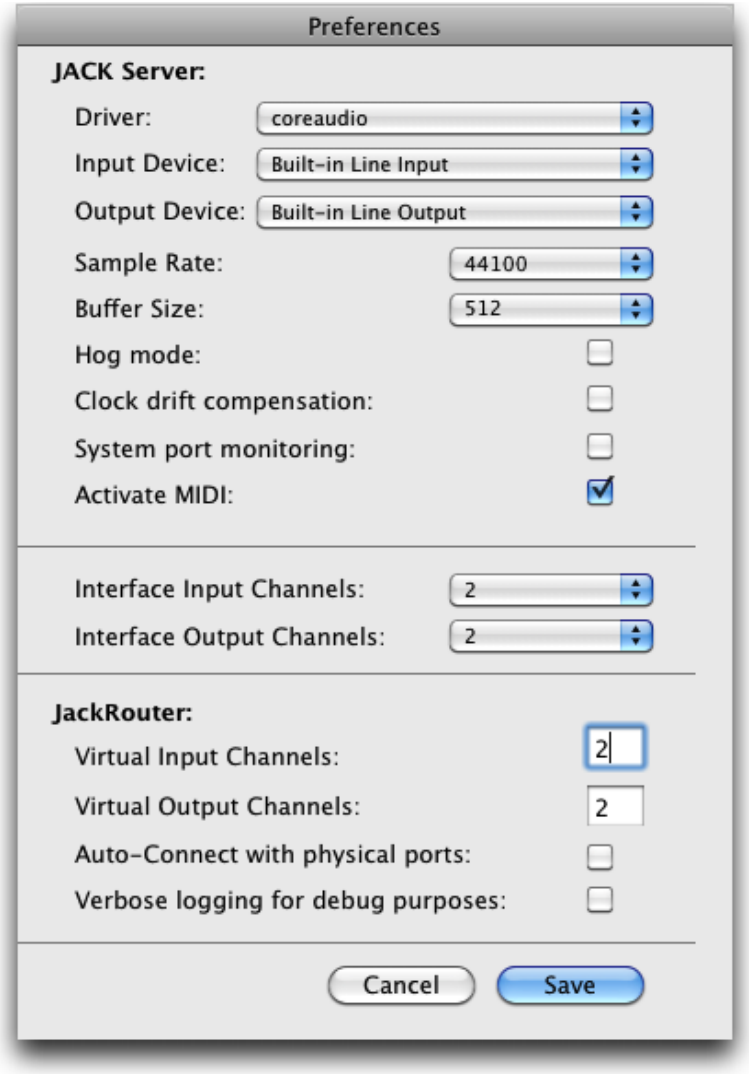
Starting JackPilot

You startup and control Jack using the JackPilot application. Navigate to the Jack folder in your Applications directory, and double click the **JackPilot** application:



Setting Preferences

If it's the first time you've run JackPilot, and/or if you've not yet saved your Jack preferences, the Preferences dialog will open automatically, so that you can set up your Jack environment:



The settings in the Preferences window are:

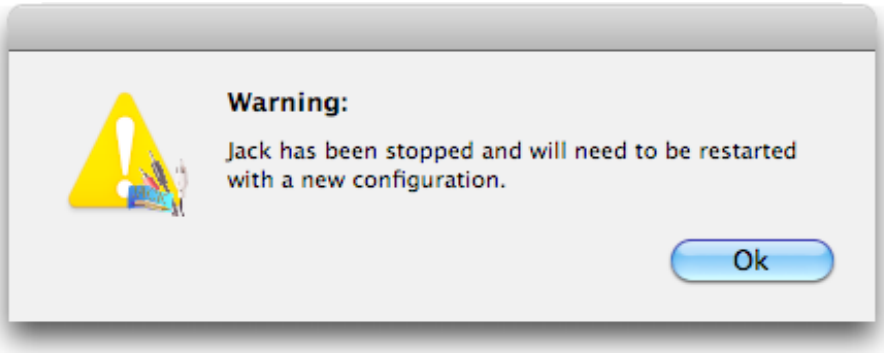
- **Jack Server**

  ○ Driver – the audio driver. Choose “coreaudio”, the only option currently available.
  ○ Input Device – select the physical audio input device that you would like the Jack server to communicate with.
  ○ Output Device – select the physical audio output device that you would like the Jack server to communicate with.

  *Please note that input and output devices can be selected independently if they are input **only** or output **only** devices. In other words, when a duplex device (that is a device with both inputs and outputs) is selected as one of the devices (Input or Output), then the other device must also be the same device. If you want to define a more complex setup (like selecting 2 inputs devices for the input and one for the output, then a more powerful tool like Audio Midi Setup, with*

creation of a dedicated Aggregate Device, has to be used.

- Sample Rate – choose your desired sample rate for the Jack server. Only sample rates supported by the selected Input and Output Devices will be displayed.
- Buffer Size – choose your desired audio buffer size (in samples). Only buffer sizes supported by the selected Input and Output Devices will be displayed.
- Hog Mode – checking this option will prevent other audio applications from adjusting the sample rate of the selected Input and Output Devices. If this option is *not* checked, and another application changes the sample rate of either the Input or Output Device while the Jack server is running, the Jack server will stop, and a dialog will be displayed alerting the user:



- Clock Drift Compensation – if selecting Input and Output Devices that are not controlled by the same master clock (e.g. a USB microphone as input, and the Mac's Built-In Audio for output), then checking this option will enable Jack OS X to provide compensation for any clock drift that may exist between the two devices.
- System Port Monitoring – checking this option enables use of looped back audio from the main system output, via the system input monitor ports. See the Monitor Ports section below.
- Activate MIDI – checking this option enables the Jack MIDI / CoreMIDI bridge. See Jack OS X and MIDI section below.
- Interface Output / Input Channels – choose your desired number of Output / Input audio channels for Jack to communicate with your physical audio interface. The maximum number of channels is limited by the number of output / input channels supported by your physical audio device(s).

- **JackRouter**

  - Virtual Input Channels – the number of virtual input channels Jack will provide for each application. Typically this should be set to 2, for stereo operation. However, using values greater than 2 is useful if your applications can manage multiple concurrent input channels (as most digital audio workstation applications can). In this case, you could set this field as high as you'd like for your particular purpose. See the second example in the Step-By-Step tutorial section below for

an illustration of how this can be useful.

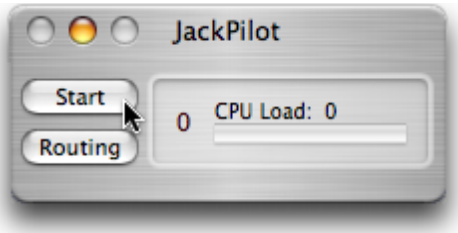
- Virtual Output Channels – the number of virtual output channels Jack will provide for each application. Same explanation as above for Virtual Input Channels.
- Auto-Connect with physical ports – check to allow JackPilot to automatically connect the Virtual Input and Output Channels to your physical audio device input and output ports, respectively.
- Verbose logging for debug purposes – check to allow Jack OS X to write more complete log messages to the Console, useful for debugging in problem cases

Please note that the Default Input, Default Output, and System Default Output options are no longer available in the JackRouter Preferences as of Jack OS X version 0.75. This was required because of changes in Mac OS X 10.5 and forward. You must manually change the Mac OS's default Input/Output/System Output options to JackRouter (using either the Sound Preferences pane or the Audio MIDI Setup application) if you want to use applications that use the Mac's default input/output device together with Jack.

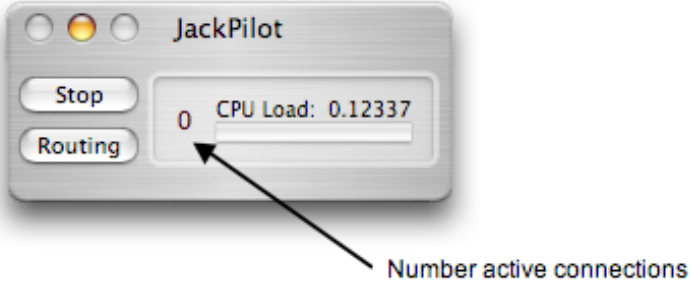
If you don't choose to save your Preferences at this time, or if you need to change them later, you can always return to the Preferences window by selecting the **Preferences…** option in the JackPilot menu. Please note that you cannot make changes to the Jack Server preferences when the Jack Server is running.

Starting the Jack Server

Click the **Start Jack** button to start the Jack server:



You'll briefly see a dialog telling you that the Jack server is starting up, and then you'll see the CPU Load display show a slight increase. The "Start" button also converts to a "Stop" button. The Jack Server is now running! The number of current Jack connections is also displayed (more on connections later).

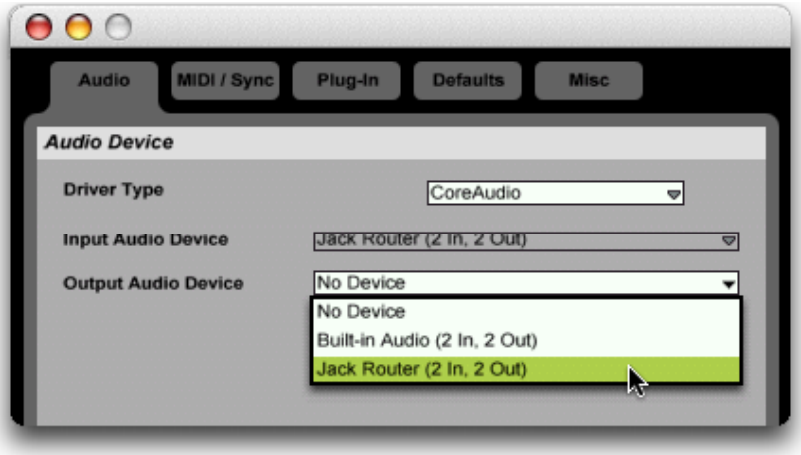


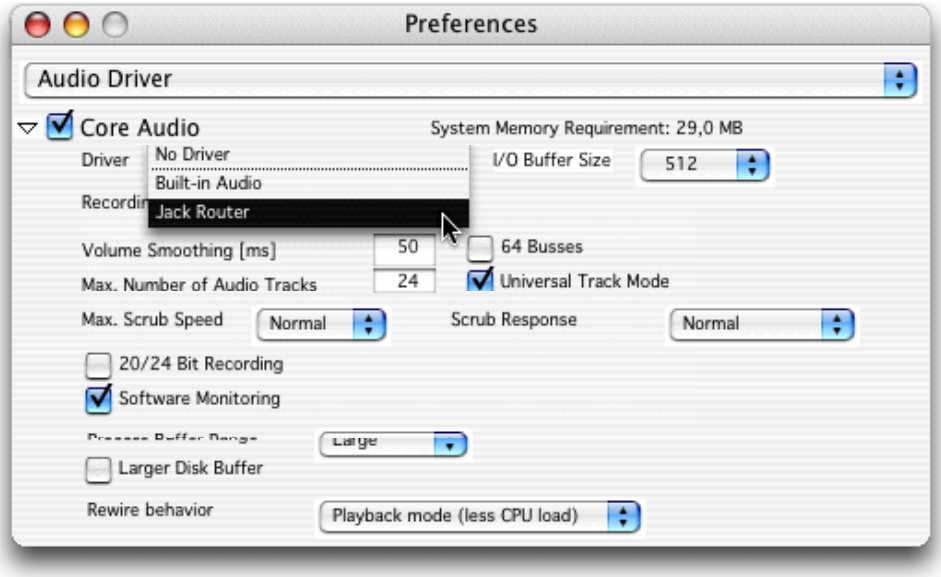
Jack-enabling Applications

Some applications, which always use the “default” Mac OS X audio device, will be “Jack-enabled” when you select Jack Router as the Default Input and/or Output device in the Audio MIDI Setup application or Sound Preferences pane. Examples of this type of application are Apple's iTunes, Quicktime and DVD Player, Real Networks' RealOne Player, or Five 12's Numerology.

Other applications allow you to select which CoreAudio device to connect to – examples include Steinberg's Cubase SX, Apple's Logic, MOTU's Digital Performer, Ableton's Live, BIAS's Peak, Cycling ‘74's Max/MSP, etc… In most instances, this will be specified in the application’s audio driver or I/O setup screen. To use Jack, select the **Jack Router** option on this setup screen. If you don’t see this option, be sure that the Jack Server is on; if it wasn’t, you might need to restart your CoreAudio compliant application before you will see the Jack Router option available to select.

The following are two visual examples, from Live 4.01 and Logic Audio Platinum 6.3.3:

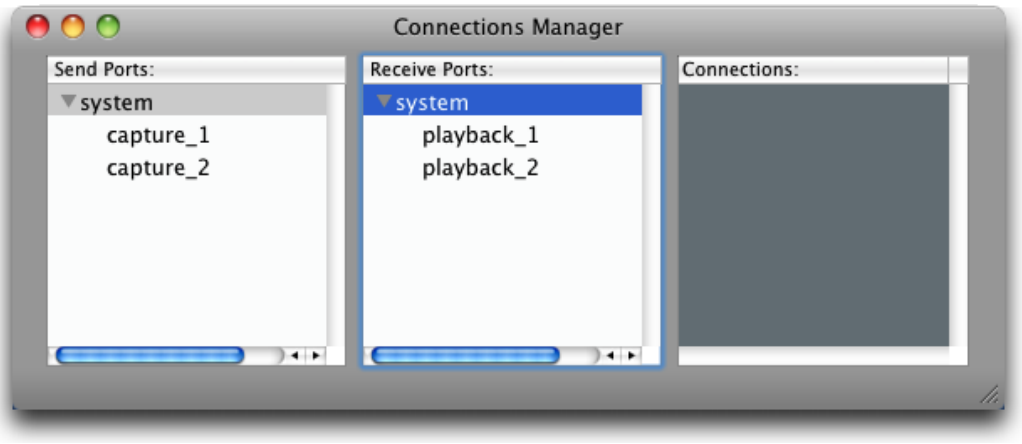




Fully CoreAudio compliant applications are *supposed* to only display the sample rate and audio buffer sizes supported by the currently selected CoreAudio driver. So when using the Jack Router driver, these applications should only offer ***one*** possible choice for sample rate and audio buffer size values, the ones chosen in the JackPilot Preferences window. This is not always the case however, as many applications offer sample rates and audio buffer values that are not supported by the driver. ***Therefore, it is best to confirm that the CoreAudio compliant application has the same sample rate and audio buffer size values selected as those you specified in the JackPilot Preferences***. Failure to do so may degrade overall Jack performance, and may cause crashes as well.

Making Connections

Once you have Jack-enabled your various applications, Jack allows you to essentially connect *any* input to *any* output, and vice versa. With the Jack server on, click the **Routing** button in JackPilot, to bring up the Connections Manager window. The following is an example of what you might see if you had not yet Jack-enabled any applications – all you see is the physical audio device (generically called "**system"**), and its send ("**capture"**) and receive ("**playback"**) ports:

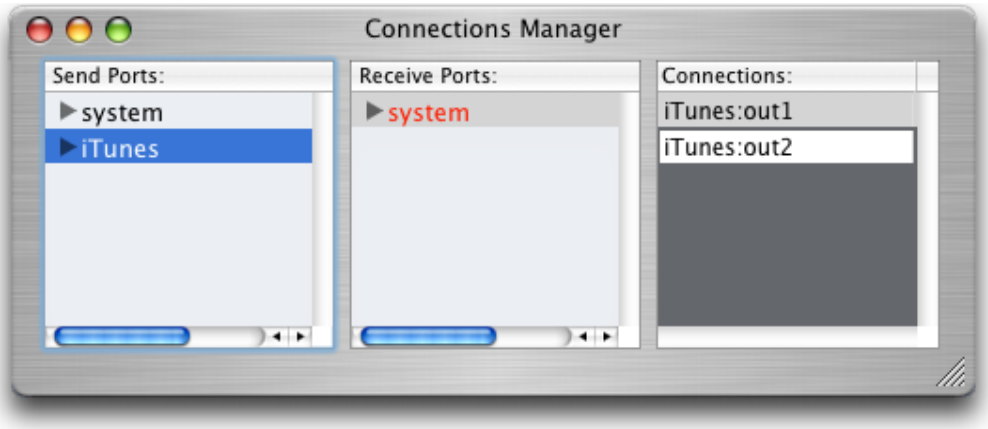


It's easy to make or break connections between Jack-aware applications and an audio device using this window:

- A list of currently active Jack-aware applications and devices will be displayed in the Send Ports and Receive Ports columns. ***Please note: some applications will not appear in the Send Ports column until they are actively sending audio.***

- A single click on an application name or device will show its connection state, with connected applications/devices turning red. A list of the individual ports it is connected to (if any) will be displayed in the Connections column.

In the following example, iTunes (which was selected with a single click) is connected to the system output. The system output is in red, because that's what iTunes is connected to. The specific ports that iTunes is connected to appear in the Connections column.



- After having selected an application or device in either the Send Ports or Receive Ports column, a double-click on a second, or target, application or device will either connect or disconnect the target application. If the target application **was not previously connected**, a double-click **connects** it to the 1st application or device; if the target application **was previously connected**, a double-click **disconnects** it from the 1st application or device.

- A double-click on a connection listed in the Connections column will disconnect it.

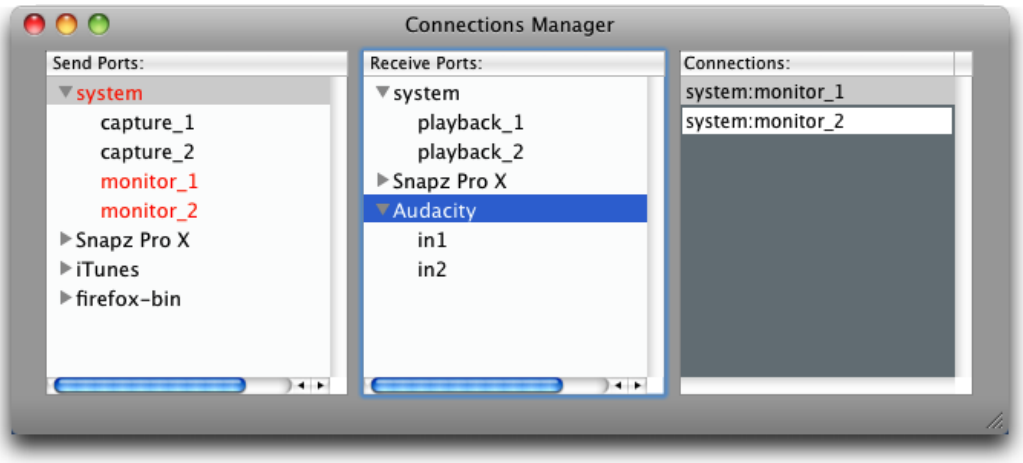
- “Opening” an application or device (using the disclosure triangle) allows one to make individual port connections and disconnections, using the same method as above.

- **Please note**: there are no restrictions on setting up potential feedback loops (e.g. connecting the **system** Send ports to the **system** Receive ports), ***so be careful*!**

Monitor Ports

Starting in Jack OS X version 0.86, **monitor ports** are available within the system device in the send column, if the System Port Monitoring option is checked in JackPilot Preferences. These ports represent audio **input** that is currently being sent to the audio **outputs** (to the system device in the receive column). In essence, the audio being sent to the output is being “looped back” to the audio input. This is convenient for example if you’ve got a complex set of interconnections within Jack, and you want to send the final resulting audio output to another application to record.

In the example shown below, both iTunes audio and audio from Firefox are being routed to the audio output device (the system playback ports). At the same time, that combined iTunes and Firefox audio is being looped back in, via the monitor ports, to be recorded using Audacity:

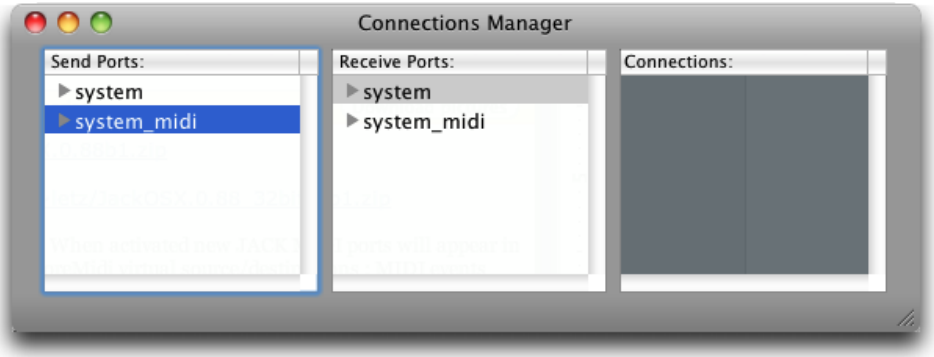


- **Please note**: there are no restrictions on setting up potential feedback loops (e.g. connecting the **system** Monitor ports to the **system** Playback ports), ***so be careful*!**

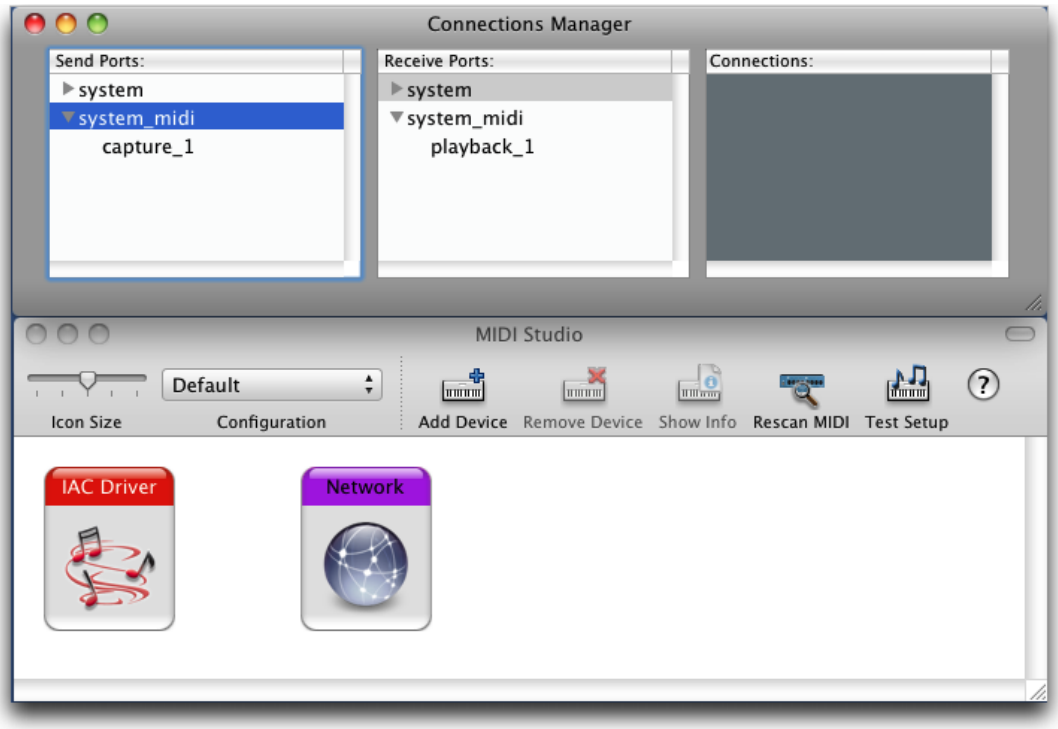
Jack OS X and MIDI

Starting in Jack OS X version 0.88, a Jack MIDI / CoreMIDI bridge has been implemented. This allows CoreMIDI information to be routed to and from Jack-enabled devices, in addition to audio information.

Once the "Activate MIDI" setting has been enabled in the Preferences (see Setting Preferences section) and the Jack server started, you will see "system_midi" ports within the Connections Manager:



Within the system_midi ports, you will see representations of all the MIDI devices available in the MIDI section of your Mac's Audio MIDI Setup application (with the exception of the Mac's "Network" MIDI device). In the example below, only one port is available, the IAC device, and so only one device is listed within system_midi (and is represented by "capture_1" and "playback_1". If you had additional MIDI devices, they would be listed as "capture_2", "capture_3", etc...

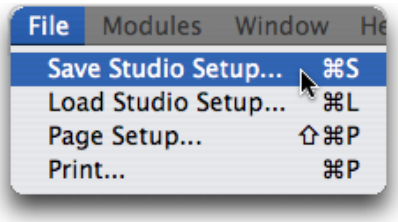


Routing MIDI from device to device is done in exactly the same way as audio connections are made – see the Making Connections section above. MIDI connections can only be made between MIDI ports, and likewise, audio connections can only be made between audio ports.

The advantage to using Jack OS X to route MIDI information is that the MIDI information is routed with sample accuracy. It should be noted that Jack OS X does not currently route SysEx information reliably however.

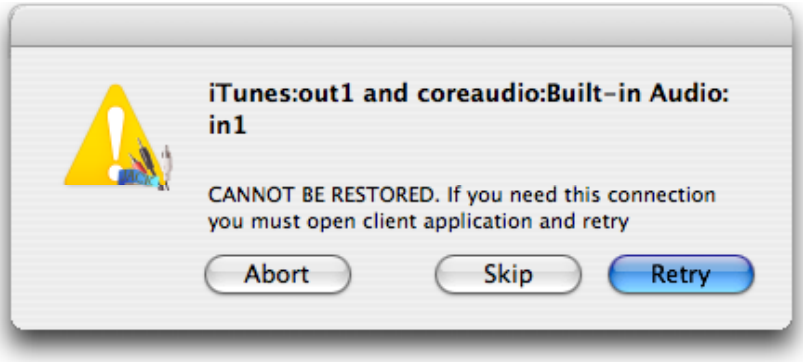
Saving & Loading Jack Studio Setups

To facilitate the use and reuse of a complex set of Jack connections, you can save a Jack "studio" setup. Within Jack Pilot's File menu, simply select **Save Studio Setup** (key combination *Command-S*) or **Load Studio Setup** (key combination *Command-L*) to Save or Load a studio setup, respectively.



Please note that Jack must be on, and the Connections window must be open, before loading or saving a studio setup. Also note that double clicking on a studio setup in the Mac's finder will **not** load it – you must load it using the Load Sudio Setup command within JackPilot.

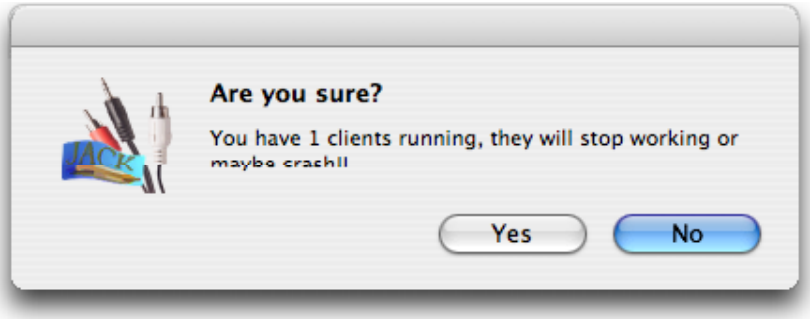
If an attempt is made to load a studio setup that contains an instrument that is not currently active and/or Jack-enabled, you will receive a warning, and given an opportunity to start the application and/or Jack-enable it:



Stopping the Jack Server

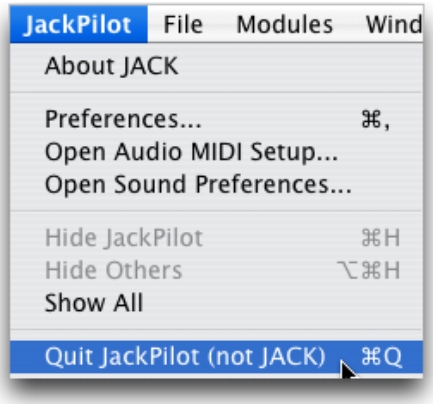
To stop the Jack Server and the JackRouter, click the **Stop** button in JackPilot:

A warning dialog may be displayed if you have clients that are still connected to Jack. Stopping Jack while clients are still connected can potentially cause them to crash, so best to disconnect them from Jack prior to stopping Jack.



Quitting JackPilot

Quitting JackPilot is done as with most Mac OS X applications, via the JackPilot menu, or using the key combination *Command-Q*:

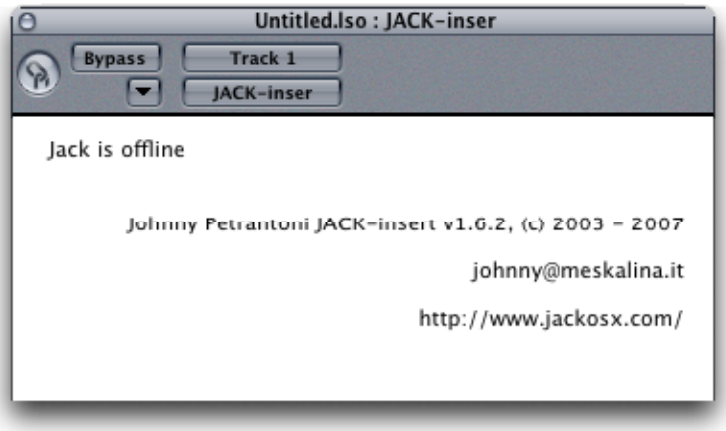


It is important to note however, that quitting JackPilot will not stop the Jack Server and/or the JackRouter – these must be stopped manually using the Stop Jack button in JackPilot.

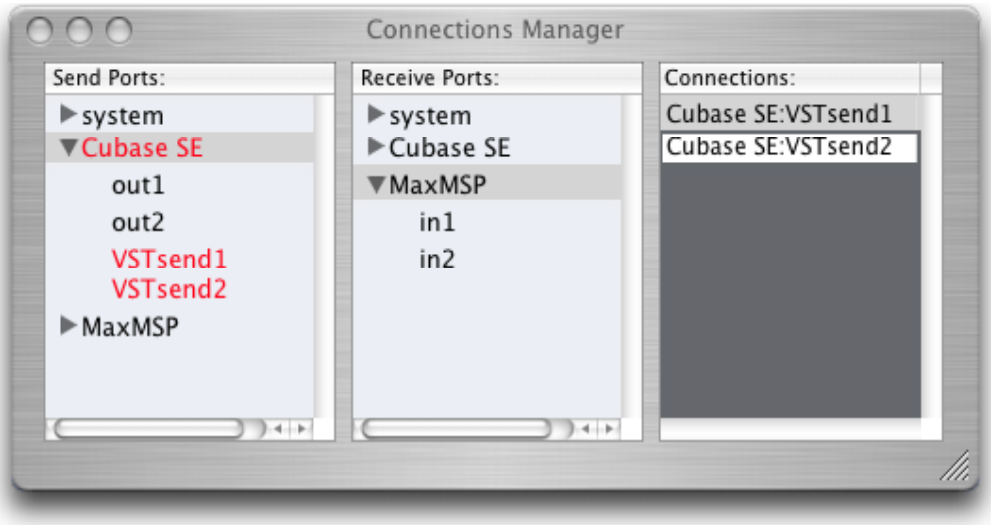
Using the Jack Plugins

Jack OS X comes complete with **JACK-insert** plugins in both VST and AU formats. This enables you to route streams of audio into Jack via the host application's plugin routing scheme. So for instance, one could route the main audio output from an application to one destination application, but then route an audio insert feed, using an AU or VST plugin, to a completely different application for processing. This greatly expands the routing flexibility of Jack when used in concert with the many VST and AU-compatible applications available today.

The image below shows the Jack AU insert being loaded within Logic 7:



In the example below, Cubase SE is sending audio via the Jack VST plugin to input channels 1 and 2 of Cycling '74 Max/MSP Live.



NetJack [Temporarily not installed]

NetJack has been temporarily removed from the Jack OS X package, until it is fixed.

Overview
NetJack is the first Jack OS X module created for use within JackPilot. It allows for low latency transmission of audio streams over a local network, or even over the Internet, provided you have a fast and reliable connection between the computers you are linking.

NetJack uses the UDP (User Datagram Protocol) transport layer protocol. Although in some respects similar to the more familiar TCP (Transmission Control Protocol) protocol

(they both utilize the IP (Internet Protocol) layer, and send data in “packets”, for example), UDP is better suited for real time audio transmission over a reliable network because it is a simpler protocol with less overhead, and therefore, less latency. If you’re interested in reading more about UDP, a good (albeit technical) overview is available at the following site: http://en.wikipedia.org/wiki/User_Datagram_Protocol .

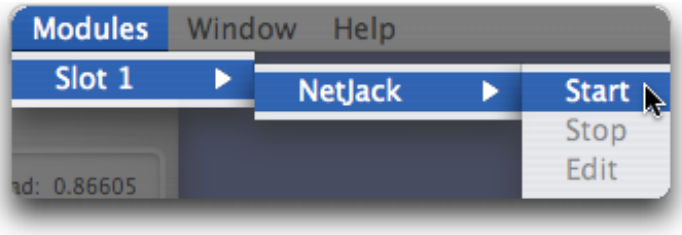
You can use any Mac OS X network interface with NetJack, including Ethernet, Airport, or IP over Firewire connections. Furthermore you can just connect two Macs to each other using an Ethernet cable, and let them discover themselves using Rendezvous.

Using NetJack

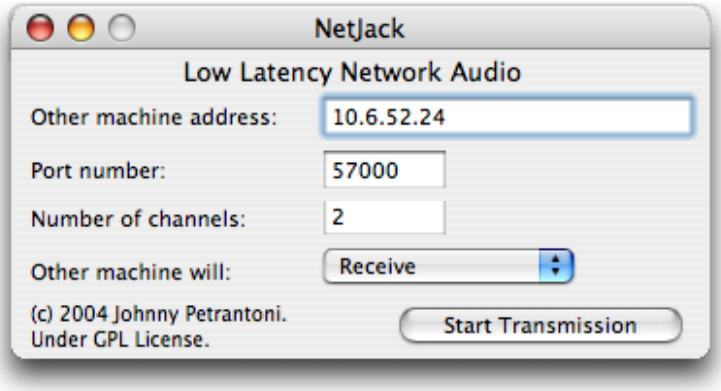
We’ll use an example in which we want to sent audio from one computer (A) to another computer (B) to illustrate how to use NetJack; rest assured that you can connect as many computers as you like.

NetJack requires Jack OS X to be installed on both computers A and B, and the Jack server should be started on both, as described in the previous sections. Also it is assumed that the two computers are accessible to each other over a network; be sure that firewalls, routers, etc. are configured to allow traffic through the selected NetJack port (more on this below).

On computer A, start a new instance of the NetJack module, by choosing the Start menu option within Slot 1 of the JackPilot Modules menu:



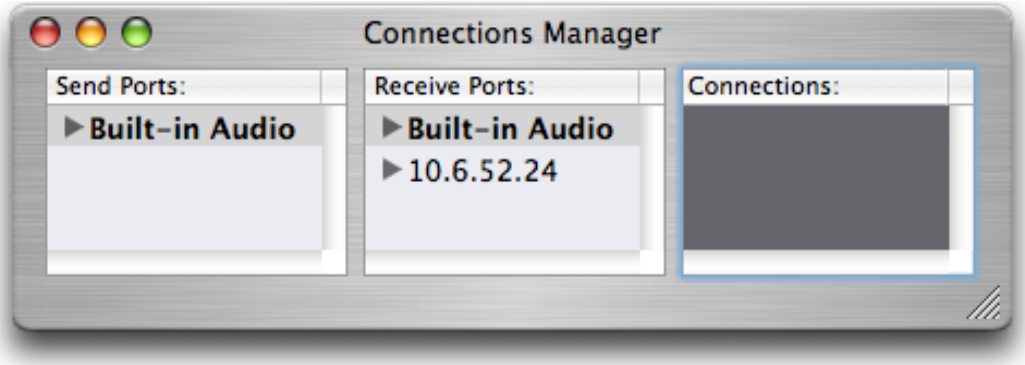
You’ll then see a new window that has the details of how and to where NetJack will connect:



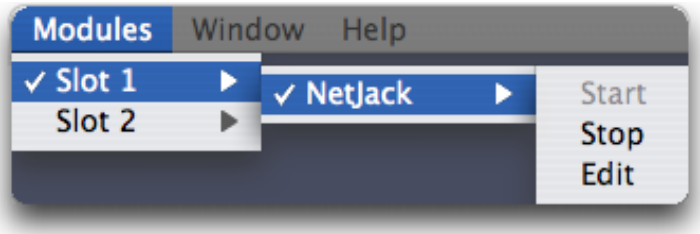
For now, we will focus on just some of these settings:

- Other machine address: This is the IP address (e.g. 10.6.52.24) or DNS name (e.g. cool.stuff.com) of the machine to which you are connecting; either will work. You can also type “localhost” (without the quotes) if you want to connect back to the same machine. If you don’t know how to determine the IP address of your Mac, please see the FAQ section below.
- Other machine will: This defines what mode the other machine will be in: Receive audio, Send audio, or Send-Receive audio. Use Send-Receive mode if the other machine will be both receiving audio as well as sending audio back – this is more efficient than using two separate NetJack instances, one to Send, the other to Receive. In our example, the other machine (B) will be receiving audio, so we will set the drop down to Receive mode.

To begin audio transmission, press the Start Transmission button; you will then see the NetJack entry in the appropriate Connections Manager column. To stop transmission, press the button again (which at that point will be labeled Stop Transmission).



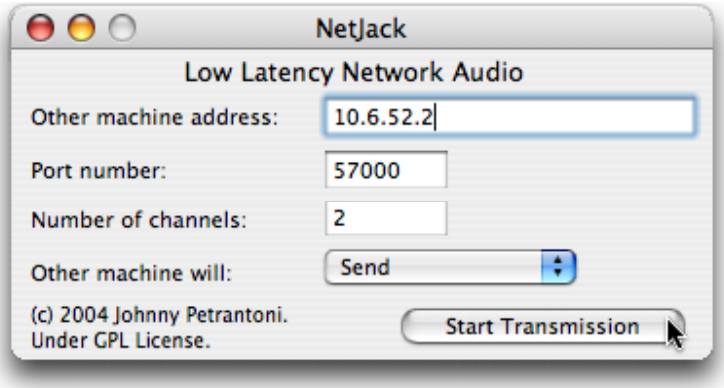
Note that if you go back up to the Modules menu, you’ll now see a second slot, Slot 2; JackPilot will keep adding or removing slots sequentially as you start or stop them. Also note that the Stop and Edit options are no longer grey in the first Slot 1 NetJack section:

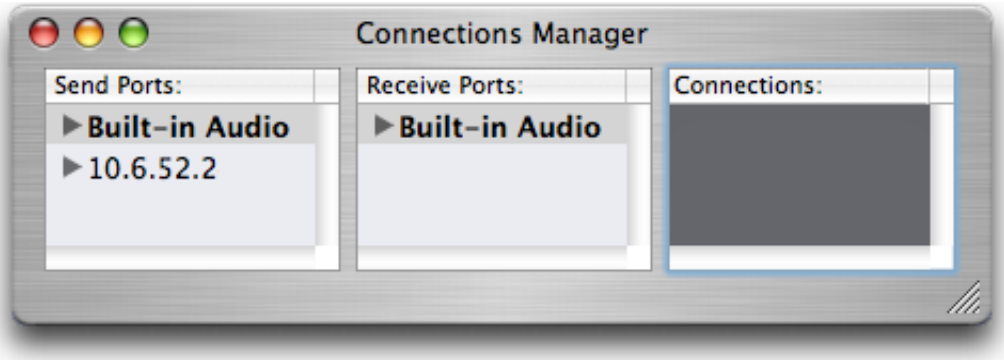


If you close a NetJack settings window using the red close button in the top left corner of its window, you can always reopen the window by selecting Edit from this menu. If you are ready to stop using a NetJack instance, choose Stop from this menu. This will stop the NetJack transmission, and remove the slot instance (unless the NetJack instance you’re using is in Slot 1; Slot 1 is never removed).

Now let’s set up the other computer. Using JackPilot on Computer B, start a new

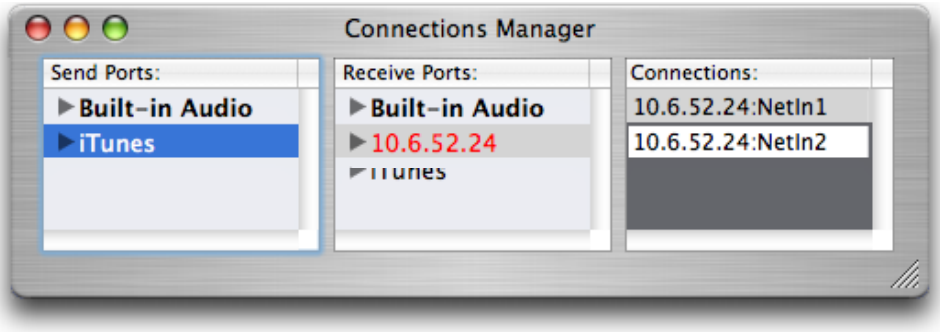
instance of the NetJack module just as you did it on Computer A, by choosing the Start menu option within Slot 1 of the JackPilot Modules menu. In the NetJack settings window, enter the IP address (or DNS name) of Computer A, be sure it is set to Send mode, and press Start Transmission.



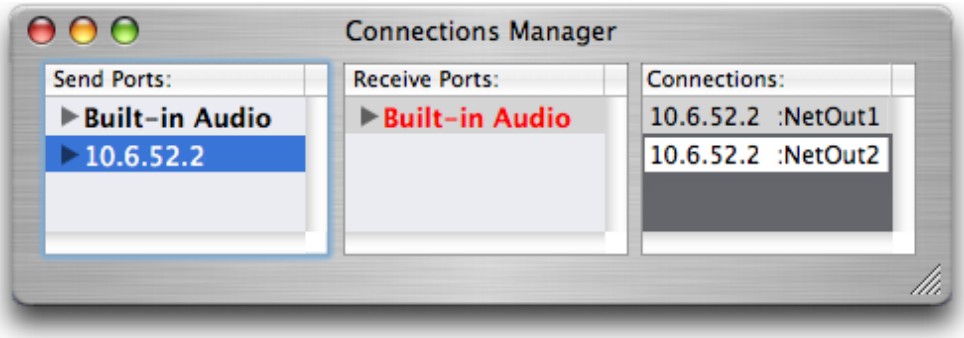


There – we've now established the connection between the two computers, each running Jack OS X. For purposes of illustration, let's say we want to send iTunes audio running on Computer A to Computer B, and we want to hear the output on Computer B's built in speakers.

On Computer A, set up iTunes to send its audio to the NetJack ports of Computer B (be sure you have audio playing in iTunes, in order to see it in the Connections Manager!) (also, if you don't know how to connect things in the Jack Pilot Connections Manager, refer back to "How Do I Use Jack?" section above):



On Computer B, connect the NetJack ports to the Built-in Audio ports:



You're done! You should be hearing your iTunes audio, which originated on Computer A, playing back on Computer B!

Now, let's return to the NetJack settings window, for an explanation of the other settings not previously described:

- Port number: The UDP port on which NetJack will send the audio streams over the network. The default port number is 57000, but you can change this to any port you desire. Just be sure that both computers that are to communicate are set to the same port. Also, be sure that any firewalls or routers are set appropriately, to allow traffic through your selected port number.
- Number of channels: This setting defines the number of channels that the particular NetJack instance will transmit. In general, do not select more channels than are necessary for your particular application, as latency will increase with increasing number of channels. In general, if you desire the lowest possible latency between connected computers, you should use the lowest number of channels per NetJack instance. In other words, in our example above, we sent a stereo iTunes stream over the network using one NetJack instance, set up for two channels. We could have accomplished the same thing using two NetJack instances, but configured with only one channel each. This setup would have yielded less latency in getting the audio from one computer to the other. The tradeoff is that this configuration requires more of your CPU, as well as more RAM.

NetJack works best over reliable, fast networks. That said, there's nothing preventing you from using it over the Internet, but just understand that it may not be reliable enough to deliver glitch-free audio back and forth.

Miscellaneous Features

Automatic Connection Restoration

Some applications are not recognized by JackPilot's Connections Manager when they are not actively sending audio. This caused problems in earlier versions of Jack OS X, as these applications would "disappear" from the Connections Manager window when

they stopped sending audio, and any connections that were already made with the application would be lost, requiring manual re-establishment of the connections when the audio was restarted.

Beginning with version 0.5 of Jack OS X, a feature in Jack OS X *restores* the connection state of the application in question once its audio stream is resumed.

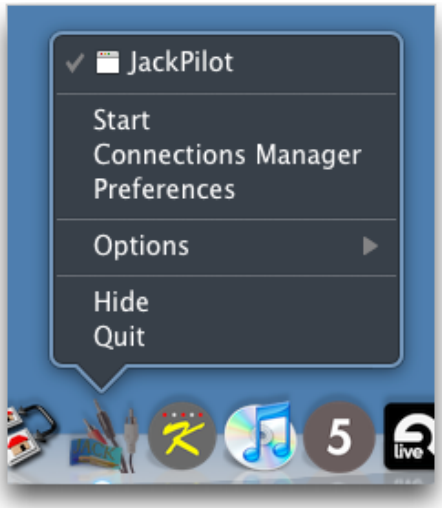
Practically this means that for these types of applications, you can start, stop, and restart the application's audio playback, with audio connections that were in place prior to stopping being restored automatically when the audio is restarted.

About Jack

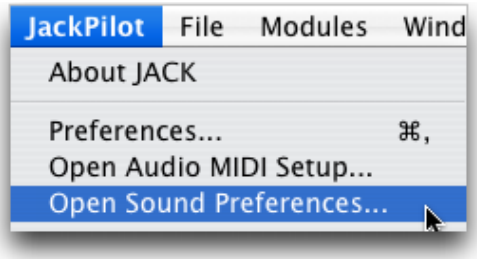
You can quickly and easily find out what version of Jack OS X you have installed (as well as its sub-components: Jack server, Jack Router and JackPilot application), by selecting the About JACK menu item in JackPilot to see the About Jack window. In addition to giving details about the currently installed Jack OS X version, it contains a link to the Jack OS X homepage at ***http://www.jackosx.com***. Click the Jack logo to close the window.

Dock Menu

JackPilot also provides a quick way to access most of its functions, by using the Dock menu; control-click on the Jack icon in the Dock to see a list of possible functions, including starting or stopping the Jack Server, opening the Connections Manager, and opening the Preferences window:



Audio MIDI Setup and Sound Preferences quick access
Quick access to the Audio MIDI Setup utility and the Sound Preferences pane is available within the JackPilot menu:



Step-by-Step Examples

In this section, we will walk you through some common Jack OS X setups in a step-by-step manner; hopefully this will familiarize you with the process sufficiently so that you can extrapolate from these examples to your particular setup. All of the examples below will use the Mac's Built-in Audio, though you obviously can the audio interface of your preference.

We'll discuss three different setups, with increasing complexity as we progress through them:

- iTunes to Logic Express
- iTunes and Real Player to Logic Pro (using four virtual inputs)
- Cubase SX, Jack VST plugin, and Max/MSP as "insert effect"

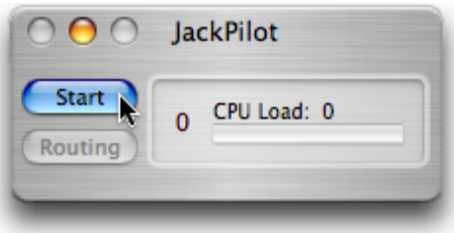
Here we go!

iTunes to Logic Express

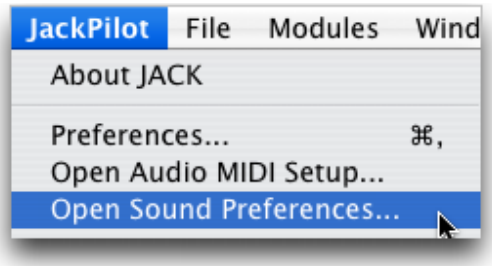
- Step 1. Start JackPilot

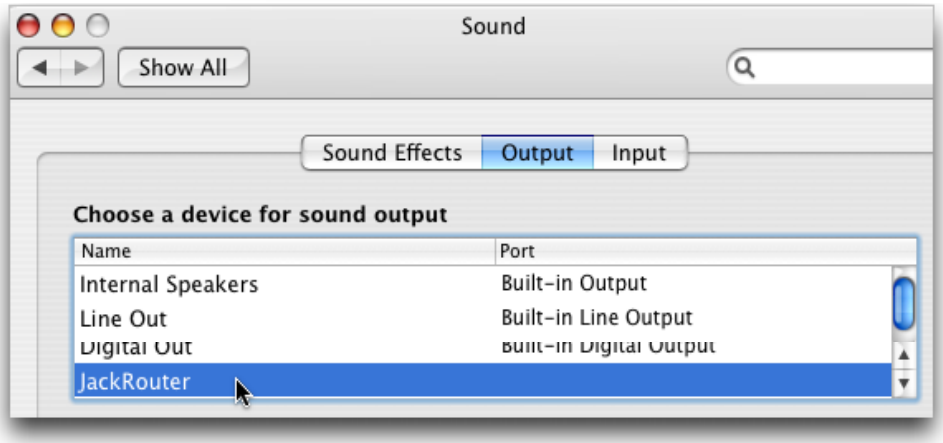
- Step 2. Be sure that Virtual Input and Output Channels are set to 2 in the JackPilot Preferences. It is simplest if you do not check Auto-Connect with physical ports:

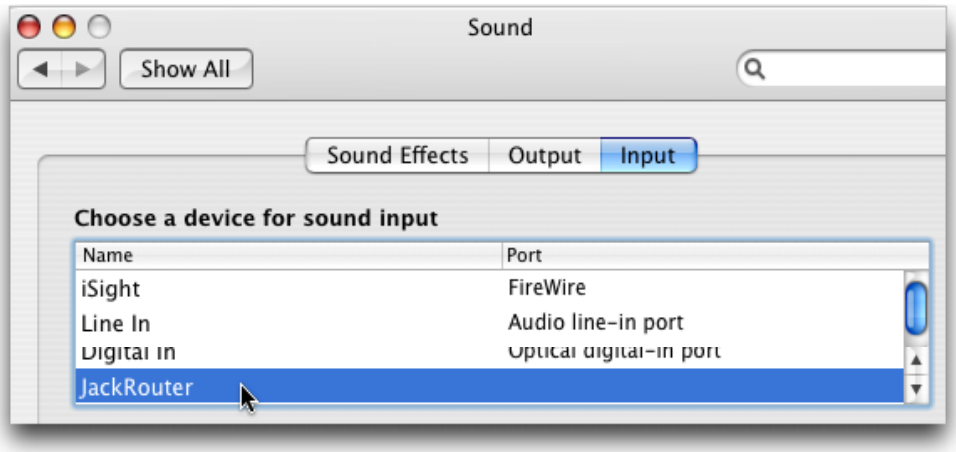
- Step 3. Start the Jack server:



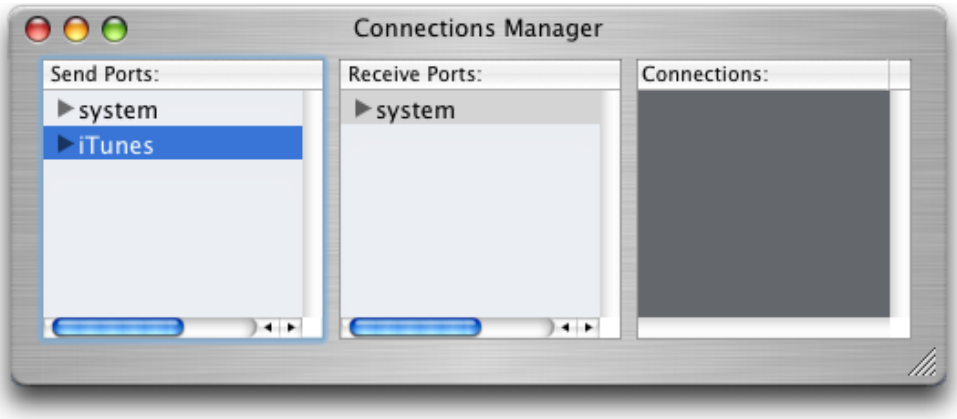
- Step 4. Open System Preferences, then select the Sound Preferences pane (or if you prefer you can use the Audio MIDI Setup application), then select JackRouter as both Output and Input device:



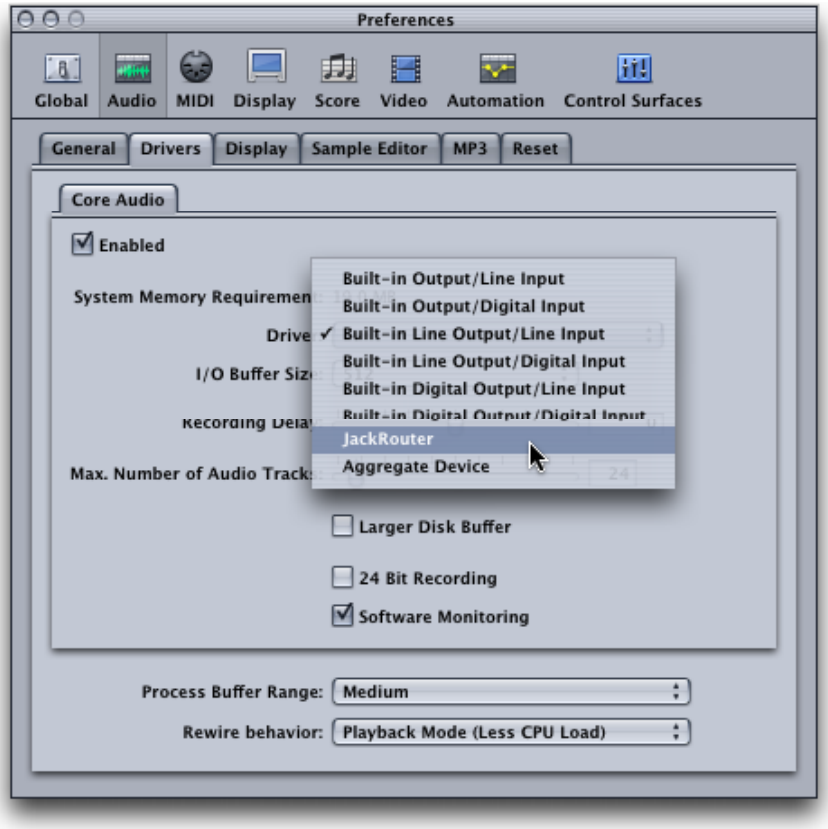




- Step 5. Start iTunes, and be sure that it is playing audio (choose either a very long audio file, or connect to an Internet radio station). You will NOT hear any audio yet – don't be discouraged! By clicking the Routing button in JackPilot, you should now see the following in your Connections Manager window:

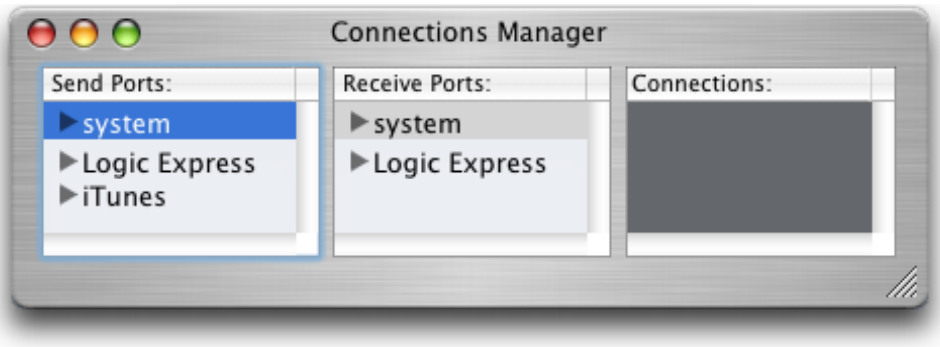


- Step 6. Start Logic, and in the Audio menu, choose "Audio Hardware & Drivers…". In the window that is displayed, in the Audio section, choose Jack Router in the Core Audio Driver section:

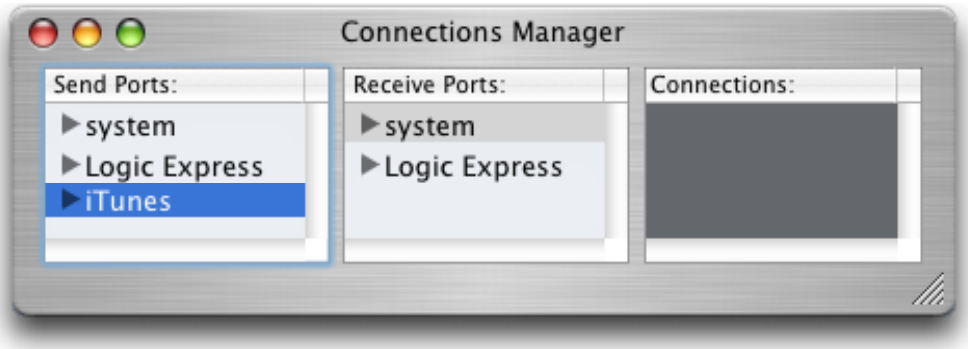


- Step 7. You may be able to just Re-Launch Core Audio, otherwise reboot Logic:

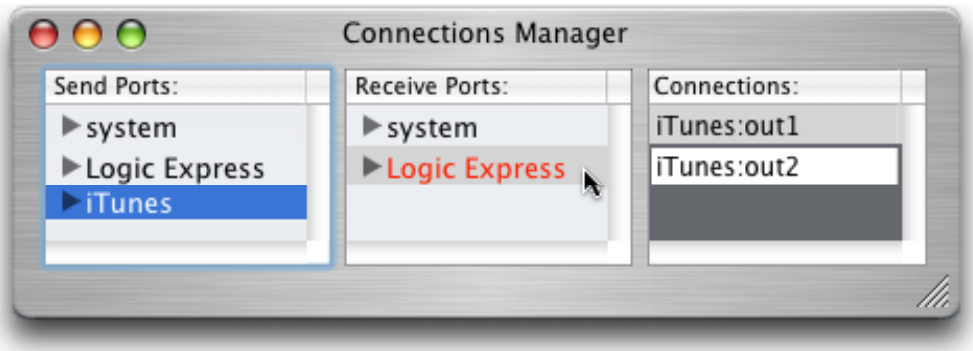
- Step 8. Your Connections Manager window should now look like this:



- Step 9. Now it's time to route the iTunes audio into Logic. Click **once** on iTunes in the Send Ports column, to highlight it blue:

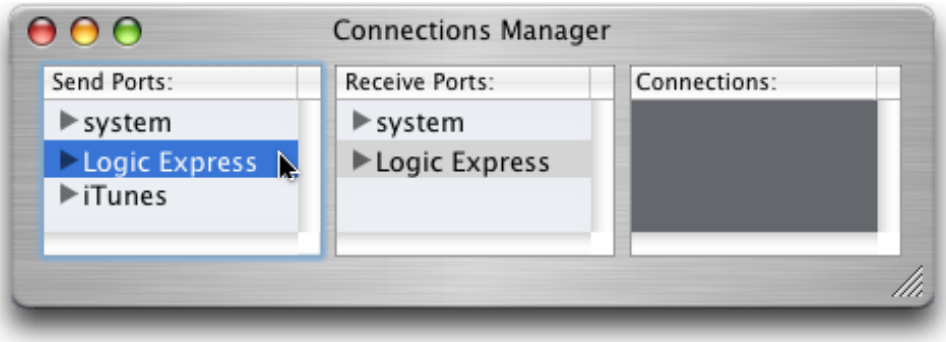


- Step 10. Now **double click** on Logic Express in the Receive Ports column. It will turn it red, and you will see the connections listed in the Connections column. You've now routed the audio from iTunes to Logic Express:

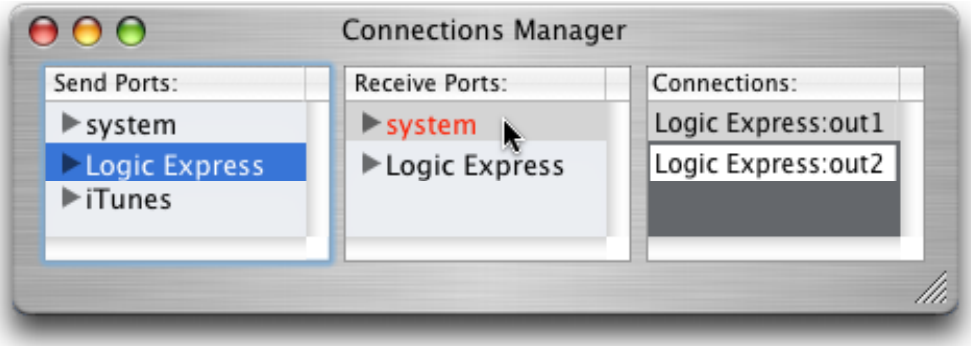


- Step 11. In Logic, create a stereo audio track, and set its input to Inputs 1 and 2. If you turn on monitoring for this track, you will see the iTunes audio streaming in:

- Step 12. All that we have left to do is to connect Logic Express to the system device within the JackPilot Connections Manager. **Single click** Logic Express in the Send Ports column, to highlight it blue:



- Step 13. Then **double click** system in the Receive Ports column – you should see the connections listed in the Connections column, and more importantly, you should hear your iTunes audio coming through your Mac's speakers, via Logic Express!



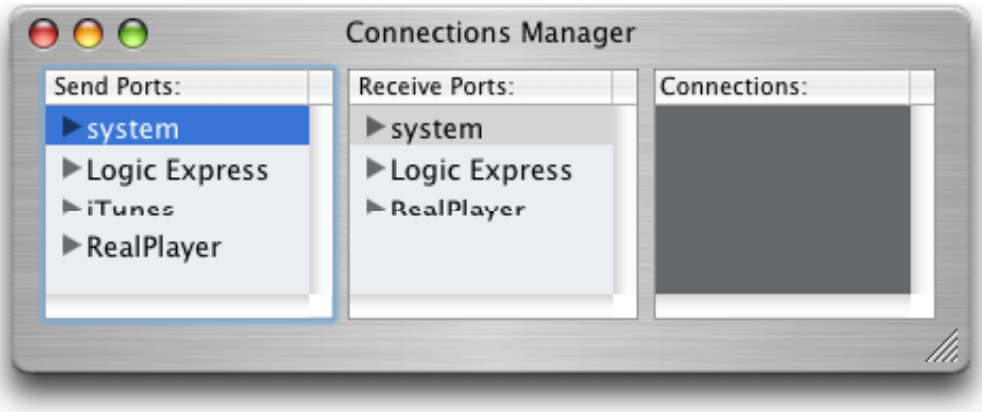
You've now got iTunes audio routed to a stereo audio track in Logic Express, and you can do anything with that audio that you could with any other Logic audio track, including processing it with a plugin, sending it to a bus, recording it, etc.

iTunes and Real Player to Logic Express (using four virtual inputs)

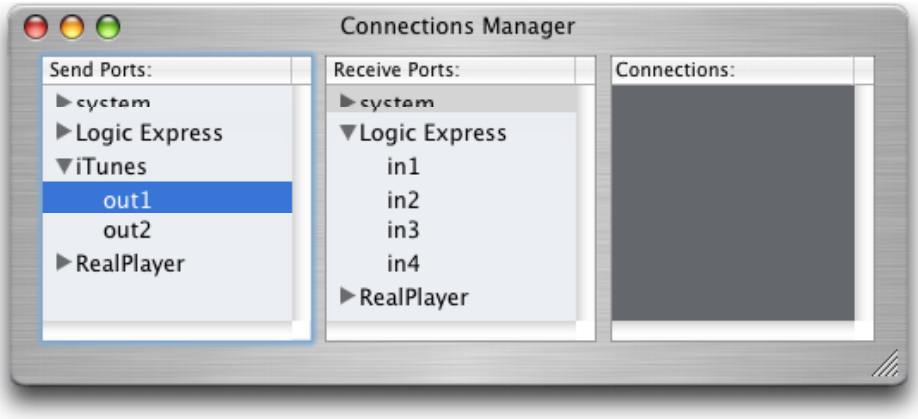
Our next example is very similar to the previous one, except that we want to route two different applications into Logic Express, and on separate channels. To do this, we will need to create more virtual inputs into Logic.

- Step 1. Start JackPilot
- Step 2. Be sure that Virtual Output Channels are set to 2, Virtual Input Channels are set to 4, and Auto-Connect with physical ports is unchecked in the JackPilot Preferences:

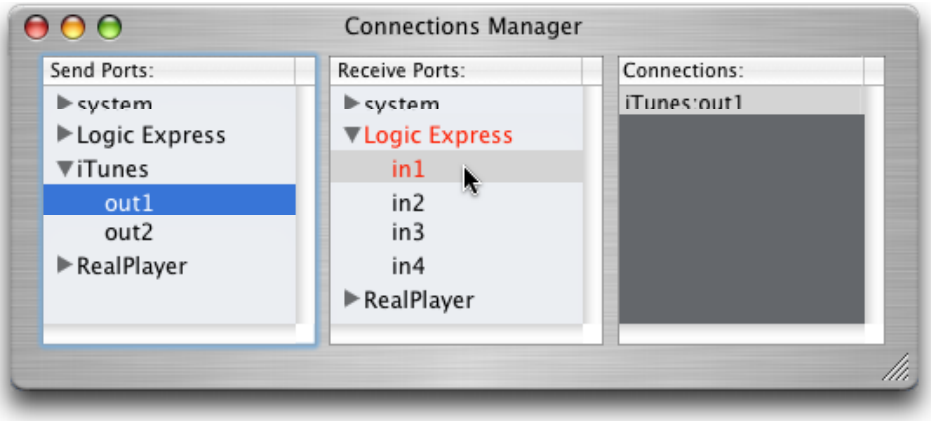
- Steps 3 – 8. Same as previous example.
- Step 9. Start Real Player, and start it playing a long audio file or Internet audio stream. Your JackPilot Connections Manager should look like this:
-



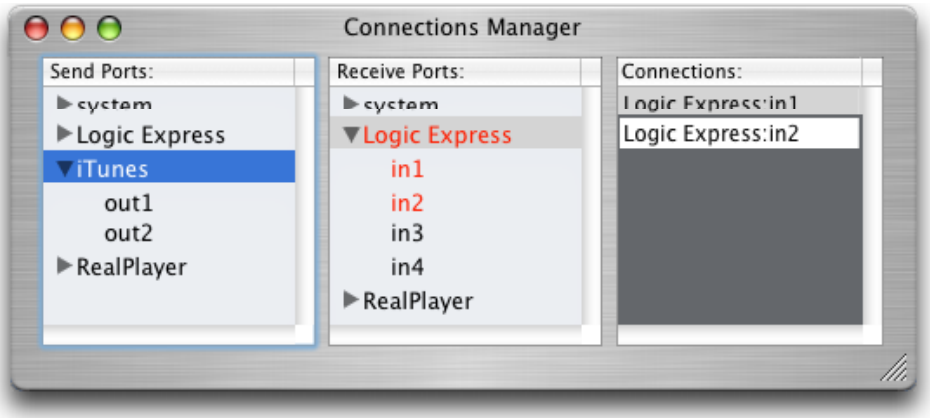
- Step 10. We'll now connect iTunes to Logic Express's first two input channels. Click the disclosure triangles next to iTunes in the Send Ports column, and Logic Express in the Receive Ports column. Click **once** on "out1" under iTunes, to highlight it blue:



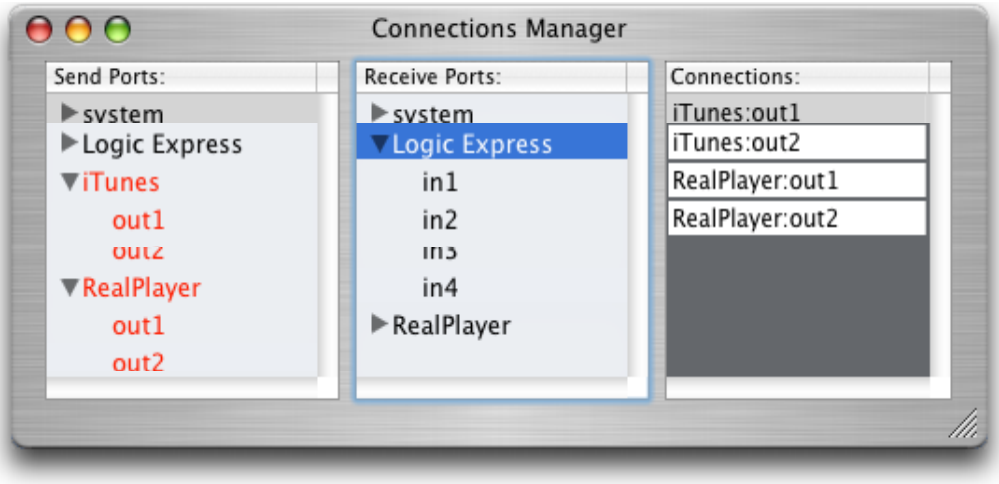
- Step 11. Now **double click** “in1” under Logic Express to connect it. It will turn red, and you will see the connection in the Connections column:



- Step 12. Repeat the last two steps, but substituting iTunes out2 and Logic Express’s in2. You’ve now connected the two iTunes outputs to the first two Logic inputs. If you now **single click** on iTunes in the Send Ports column, you should see the two connected Logic Express ports turn red in the Receive Ports column, and you should see both connections displayed in the Connections column:

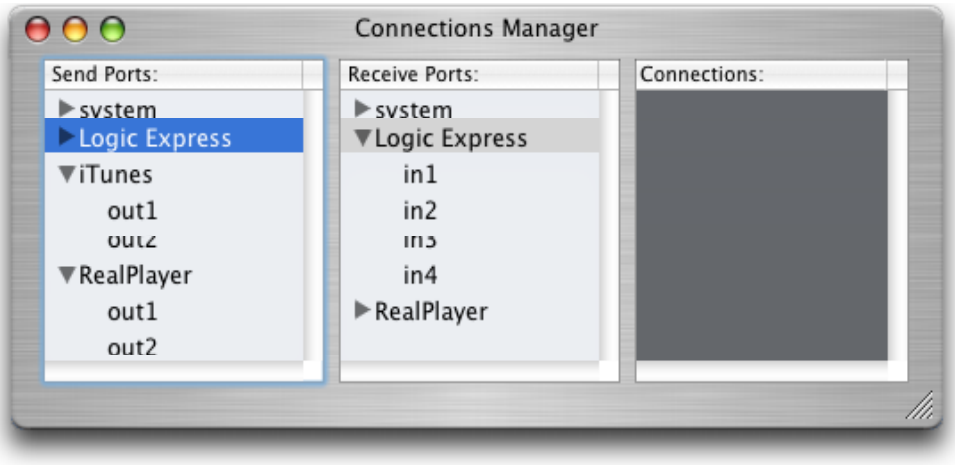


- Step 13. Repeat steps 10 – 12, but substituting RealPlayer for iTunes, and Logic Express's in3 and in4. If you then **single click** on RealPlayer in the Send Ports column, you should see the two connected Logic Express ports (in3 and in4) turn red in the Receive Ports column, and you should see both connections displayed in the Connections column. You can also try **single clicking** Logic Pro in the Receive Ports column, to see all four connections to it (two from iTunes, two from RealPlayer) turn red:

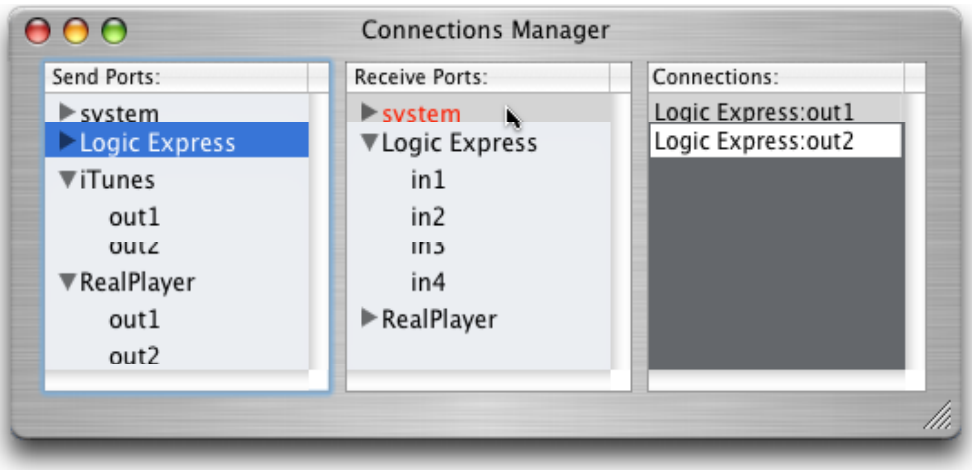


- Step 14. Now head back to Logic, and set up two stereo audio tracks, one set to get its input from channels 1 and 2, the other set to get its input from channels 3 and 4. Put both tracks in monitoring mode, and you should see audio streaming in to each: iTunes on channels 1 and 2, RealPlayer on channels 3 and 4:

- Step 15. All that we have left to do is to connect Logic Express to the system device within the JackPilot Connections Manager. **Single click** Logic Express in the Send Ports column, to highlight it blue:



- Step 16. Then **double click** system in the Receive Ports column – you should see the connections listed in the Connections column, and more importantly, you should hear your both iTunes and RealPlayer audio coming through your Mac's speakers, via Logic Express!



Cubase SE, Jack VST plugin, and Max/MSP as "insert effect"

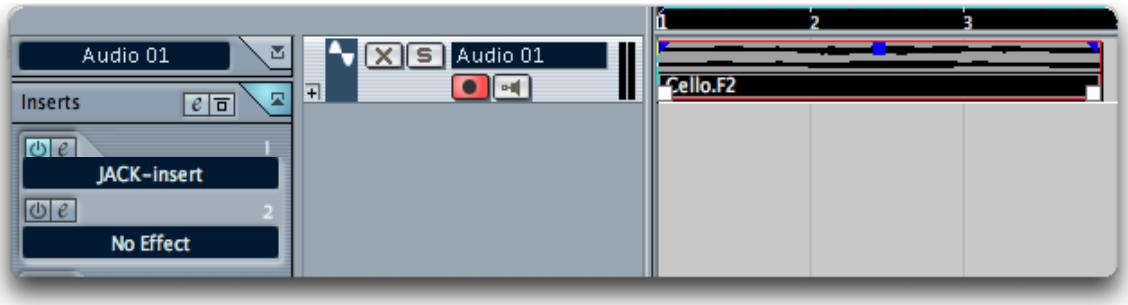
Sometimes, routing a whole application's audio output through to another application is not what's desired; sometimes, you might want to just route a single audio track into another application for processing. The Jack plugins are one way of doing this.

In this example, we will take a single stereo audio track out of Cubase SE, and using Jack's VST insert effect, will process that audio with Max/MSP, and then return it back to Cubase SE.

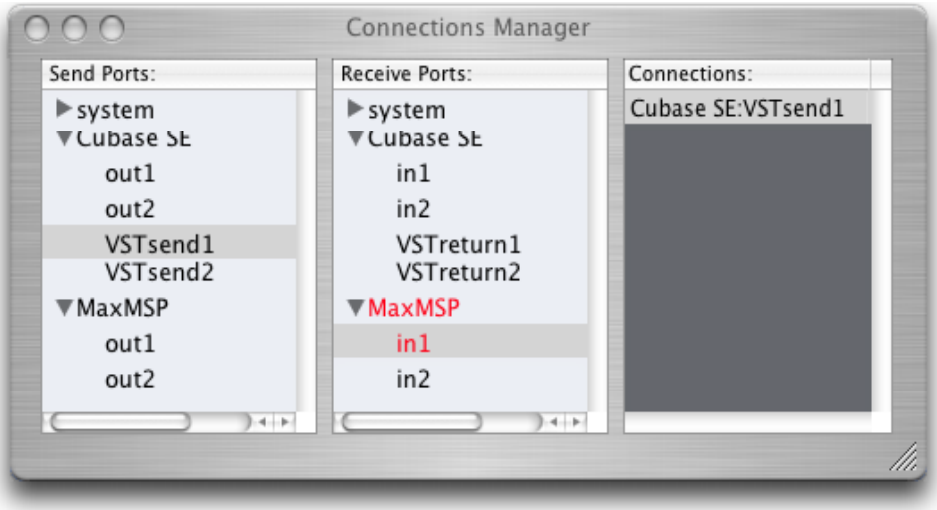
Since this is the last example, we'll assume you already know how to do the basics,

including starting JackPilot and the Jack server, making connections within the Connections Manager, and how to “Jack enable” both Cubase and Max/MSP.

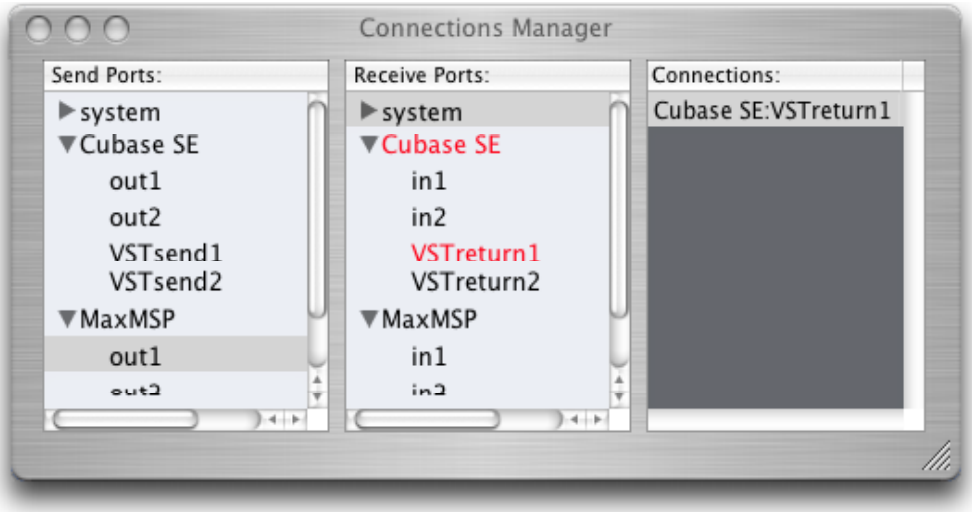
- Step 1. Start Jack, using two Virtual Input and Output channels (see first example for more details).
- Step 2. Start Cubase SE, and select the JackRouter as its audio driver.
- Step 3. Start Max/MSP, and select the JackRouter as its audio driver.
- Step 4. In Cubase SE, create an audio track (could be a sample, a software synth, whatever you want), and add Jack as an insert effect on that track:



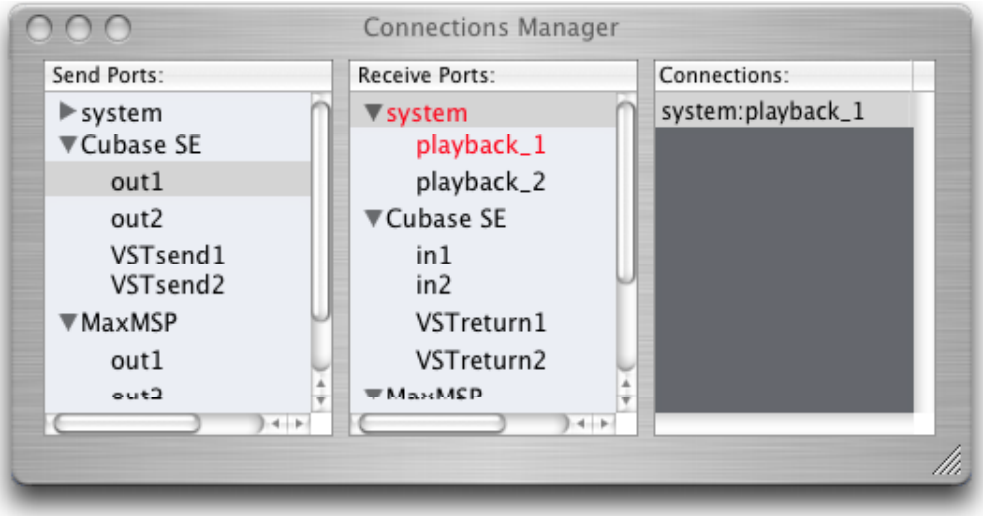
- Step 5. Open a Max/MSP patch; we’ll use a pitch shifting patch that takes an audio input stream, shifts its pitch, and streams the modified audio back out.
- Step 6. Now let’s connect things in the JackPilot Connections Manager window. Open the disclosure triangles next to Cubase SE, Max/MSP, and system. Connect Cubase SE VSTSend1 and VSTSend2 in the Send Ports column to Max/MSP in1 and in2 in the Receive Ports column (we illustrate just the VSTSend1 to in1 connection below):



- Step 7. Connect Max/MSP out1 and out2 in the Send Ports column to Cubase SE VSTReturn1 and VSTReturn2 in the Receive Ports column (we illustrate just the out1 to VSTReturn1 connection below):



- Step 8. Connect Cubase SE out1 and out2 in the Send Ports column to the system playback_1 and playback_2 (we illustrate just the out1 to playback_1 connection below):



You've now set up a Max/MSP as an "insert effect" on your audio track in Cubase; if you play your audio track in Cubase, you should be able to manipulate that audio in real time within your Max/MSP patch, and have the resulting audio come right back into your original Cubase audio track. Congratulations!

Developer Information

For developers who wish to use the Jack API, the Jack OS X library is distributed in two ways:

- the Apple approach: a Jack framework that contains the library and the associated headers
- the Unix approach: a libjack.dylib library installed in usr/local/lib, the associated

headers installed in usr/local/include, and a corresponding jack.pc file

Developers can use either the Apple or Unix approaches, depending on their need. The Unix approach is especially helpful when porting applications from Linux. The "example-clients" folder contains an XCode project and a makefile to illustrate the two ways of developing Jack native applications on Mac OS X.

FAQ

Q: How is Jack different than Propellerhead's ReWire?

A: Propellerhead's ReWire technology is similar to Jack in that it allows inter-application audio exchange. Jack however, works with ***all*** CoreAudio compliant OS X audio applications, not just the ones in which their manufacturers have included inter-application audio support, as with ReWire. Jack also provides for both AU and VST plugin support, further allowing for more interesting audio routing possibilities. Furthermore, Jack is ***free*** under the GPL and LGPL licensing models, to both end users and developers, unlike ReWire, which is proprietary and requires licensing from Propellerheads.

Jack does *not* support inter-application MIDI communication however, unlike ReWire. This is lessened in importance in OS X however, as OS X's CoreMIDI allows for implementation of inter-application MIDI communication natively. (See related FAQ question below).

Q: How is Jack different than Cycling 74's Soundflower?

A: Soundflower is a free utility from Cycling '74 that provides 2 and 16 channel CoreAudio ports that are available to all CoreAudio applications. It is a kernel driver. Although very effective for straightforward setups, it is less flexible than Jack in its routing abilities. Like Jack, it is open source.

Q: Does Jack OS X work with "Aggregate Devices"?

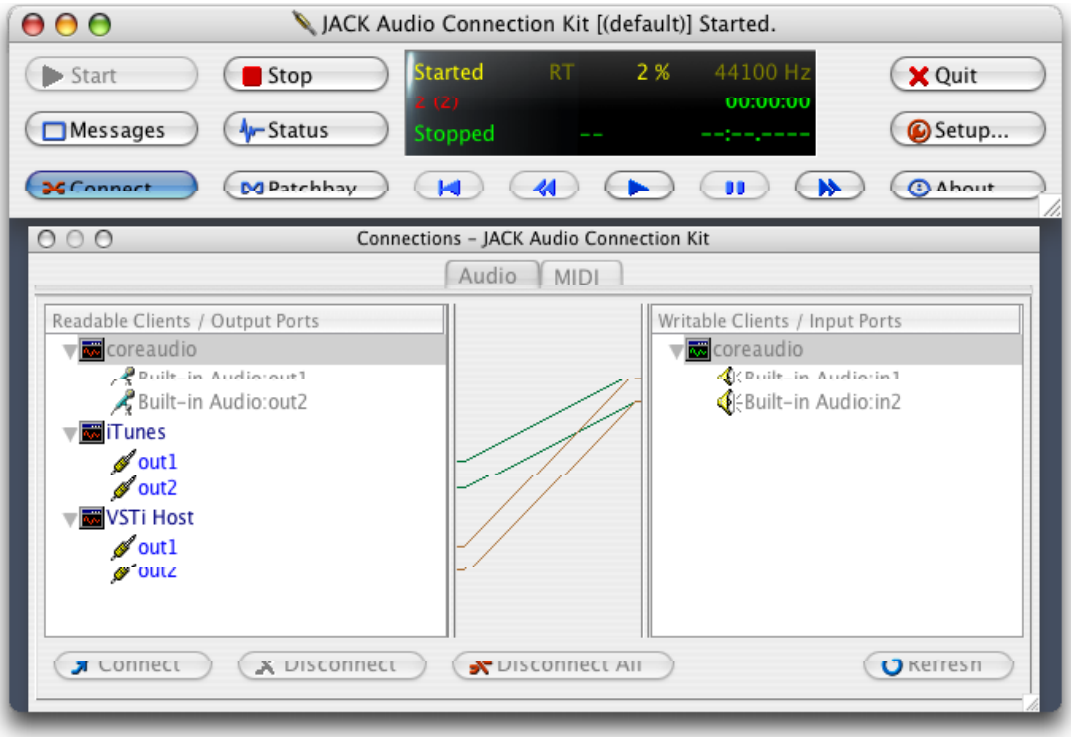
A: Aggregate Devices (created and managed from within the Audio MIDI Setup application) can be used like any "native" device in Jack OS X; the device will appear in the JackPilot Preferences Input and Output Device selection dropdowns, and if selected there, in the Connections Manager, where the set of all audio ports included in the aggregate device will appear. **Starting with Jack OS X 0.84, it is no longer necessary to create and use Aggregate Devices on Intel Macs, which used to be required in order to create a "single device" from the separate input and output half duplex devices that are present on Intel Macs.** Aggregate Devices will not work properly if created using a non-Administrator account. For a more complete overview of how to create an Aggregate Device using the Audio MIDI Setup application, see Apple's documentation at: http://www.apple.com/ca/pro/techniques/aggregateaudio/

Q: Can Jack OS X work with Qjackctl, instead of JackPilot?

A: Qjackctl (http://qjackctl.sourceforge.net) is an open source application, developed by Rui Nuno Capela, that is similar to JackPilot, in that it controls the Jack server. Unlike JackPilot though, it supports many of the additional Jack features available on Linux,

but not yet on OS X (e.g. transport control). One often-requested enhancement to the current Jack OS X package is the ability to more easily make or break connections between devices/applications, and Qjackctl does support a more intuitive graphical connection manager (see below).

Rui and several other open source developers have been able to get Qjackctl and Jack OS X to work together; a disk image is available at the following location: http://www.ardour.org/requirements_osx



Q: Why doesn't iTunes appear in my JackPilot Connections Manager?
A: The Default Output setting, in either the Sound Preferences pane or within the Audio MIDI Setup application, must be set to Jack Router. This is because iTunes uses the default OS X output device when sending its audio (also similar for Quicktime Player, and other apps that use the default OS X output device).

Q: Does NetJack send MIDI data over a network?
A: No. But there are a few utilities available that do, that you could use in conjunction with Jack OS X and NetJack:

- MIDIoverLAN CP
  - http://www.musiclab.com/products/rpl_info.htm
- iMIDI (beta)
  - http://www.grantedsw.com/imidi

Also, starting in OS X 10.4.x, the ability to send MIDI data over the network is built into the operating system – for more information and step by step procedures, open the **Audio MIDI Setup** application in Applications/Utilities, select **Audio MIDI Setup Help**

from the Help menu, click the **See all Audio MIDI Setup topics** link, and then click the **Sending MIDI device information over a network** link**.**

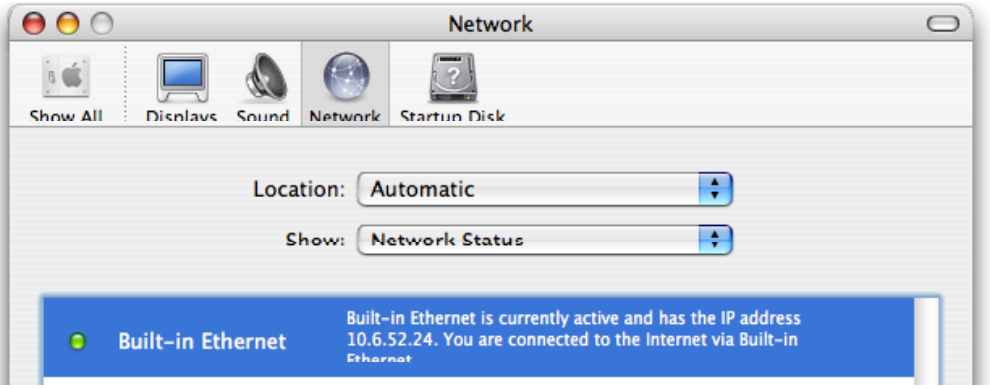
Q: How is NetJack different than other network-enabled audio tools?
A: apulsoft's WormHole is an AU and VST plugin-based system that allows for audio transmission over a network. It uses TCP/IP however, which theoretically will yield higher latency. It also does not work with applications that do not support VST or AU plugins.

FX-Max's FX Teleport allows you to route audio from/to VST plugins on another machine that is network accessible. It is intended to allow for reduction of processing load on a single machine by the distribution of work to multiple machines that are connected over a network. It is not yet available for Mac, nor is it free or open source. It only works with VST plugins and instruments.

Steinberg's System Link allows machines to be networked over physical connections; it is not IP-network based. It is only supported by Steinberg applications. It does transmit MIDI over the connection however.

Apple's Logic provides for "distributed audio processing". This allows for the built in Logic plugins to be run on connected (via TCP) "slave" computers, whose audio is then routed back to the main Logic application on the "master" computer. Like Steinberg's SystemLink, this is an application-specific technology that likely works very well for users of Logic, however affords less flexibility than Jack OS X.

Q: How do I find a machine's IP address?
A: Open the Mac's System Preferences, choose "Network", and choose "Network Status" next to the "Show" drop down menu. You will then see a summary of your Network connectivity, which will include your current IP address if connected to a network:



Q: How should one select the buffer size within JackPilot?
A: Small buffer sizes are needed when using audio applications that require real-time interaction, when a small latency is required between the physical audio input and output. The drawback is that CPU consumption for the system is higher with smaller buffer sizes. Larger buffers (like 512 or 1024) can be used when there is no such

requirement.

Q: What is the difference between the Jack Server and the JackRouter?
A: The Jack Server is the core infrastructure of Jack, and is necessary to use Jack in any way. It routes audio between both applications and audio hardware. The JackRouter is an OS X-specific development. It is a Core Audio "user space" driver, that allows for any Core Audio compliant audio application to become "Jack enabled".

*Q: How do I use inter-application **MIDI** communication, when I'm using Jack for inter-application **audio** communication?*
A: CoreMIDI services in Mac OS X provide similar functionality to OMS's IAC (inter-application communication) MIDI bus, used in OS 9 and earlier. This allows for the utilization of "virtual" MIDI busses, between applications. Currently, some applications provide their own virtual MIDI busses in OS X – examples include Cycling 74's Max/MSP, Pete Yandell's Simple Synth, and Five 12's Numerology. Starting with version 10.3 of OS X (Panther), an IAC MIDI bus comes as part of the operating system for *any* application to use – take a look at the **Audio MIDI Setup** application in Applications/Utilities, in the MIDI Devices section:

Q: What does the CPU meter in JackPilot represent?
A: The CPU meter in JackPilot represents the sum of the *real-time audio thread* CPU use of **all** Jack clients. Thus it gives a partial picture of total CPU use, since it does not take into account any GUI impact on CPU use of either JackPilot or of the Jack clients.

Known Issues & Limitations

- RealPlayer (http://www.real.com) has some problems when running within Jack OS X – some files are played on only one channel and its pitch is lowered. This has been identified as a bug in RealPlayer.

- Two specific Logic limitations:
  - The JackPilot buffer size must be set to 1024 for the Jack AU plugin to work correctly. This is because Jack's buffer size must be greater than or equal to the plugin buffer size, and Logic seems to keep it fixed at 1024.
  - When the JackPilot Connections Manager is used, the Jack client graph is stopped when the connection state is read or new connections are made; this can cause an error message in Logic ("Error while trying to synchronize Audio and MIDI") if Logic was currently playing. Stopping and

starting Logic playback solves the problem.

- The Apple DVD Player application only works when the Jack buffer size is set to 1024; this seems to be a limitation of the application itself, and a bug report has been filed with Apple.

- The contents of the JackPilot Connections Manager columns, as well as the number of connections displayed in the main JackPilot window, do not refresh immediately when there is a change in state.

- Powering an audio interface on or off, or hot-plugging an audio interface, will cause errors in JackPilot if it is already running. To remedy the problem, quit JackPilot, and start it up again after making your audio device changes.

- When using a Jack OS X on a Mac laptop with built in volume adjust keys, these keys may not work when using an Aggregate Device with Jack OS X. This is not a Jack OS X problem, but rather a limitation of the Core Audio infrastructure (you can try to see if the keys work without Jack OS X at all, and just using an Aggregate Device).

Contacts

- Stephane Letz – letz @ grame.fr
    - OS X Jack port, Jackdmp multi-processor optimized version of Jackd, JackRouter

- Dan Nigrin – dan @ defectiverecords.com
    - Documentation, Testing, Web Site, Public Relations

- Johnny Petrantoni – johnny @ lato-b.com
    - JackPilot, Jack-aware plugins, NetJack, JackRouter

- Paul Davis – paul @ linuxaudiosystem.com
    - Initial Jack concept, design and main developer

Links

- Jack OS X - http://www.jackosx.com

- Jack OS X discussion group – http://groups.yahoo.com/group/jackosx

- Jack – http://jackaudio.org

License

License (LGPL), version 2.1, as published by the Free Software Foundation. The identification of which parts of Jack OS X are covered by either the GPL or the LGPL is specified below:

- Jack – GPL, except for the jack.h file (used when an application needs to access the Jack API), which is LGPL.
- JackRouter – GPL
- Jack AU and VST plugins – GPL
- JackPilot – GPL
- NetJack – GPL
- Panda - LGPL

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

The GPL and LGPL are available from the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA, or online at http://www.gnu.org/licenses/license-list.html#GNUGPL

Copyrights and Trademarks

Version History [Updated]

- 0.90 – December 22, 2011
  - On OSX Lio, JackRouter component can be selected as default device.
  - Jackdmp / Jack2 – 1.9.8
  - JackRouter – 0.9.6
  - JackPilot – 1.7.4
  -
- 0.89 – November 28, 2011
  - Flash timing bug fixed
  - Improved support of Avid/Digidesign soundcards
  - ProTools 9 support added
  - Jackdmp / Jack2 – 1.9.8
  - JackRouter – 0.9.5
  - JackPilot – 1.7.3

- 0.88 – April 15, 2011

  - CoreMIDI / Jack MIDI bridge implemented
  - Jackdmp / Jack2 – 1.9.7.1
  - JackRouter – 0.9.4
  - JackPilot – 1.7.2

- 0.87 – September 28, 2010
  - Fixed a problem with saving studio setups within JackPilot.
  - Jackdmp / Jack2 – 1.9.6 (SVN 4047)
  - JackRouter – 0.9.3
  - JackPilot – 1.7.0

- 0.86 – September 3, 2010
  - Fixed an issue in CoreAudio driver, especially with Apogee cards
  - Added monitor ports feature
  - Jackdmp / Jack2 – 1.9.6 (SVN 4047)
  - JackRouter – 0.9.3 : Internal cleanup
  - JackPilot – 1.6.9 : Fixed a bug in English “About” box.

- 0.85 – December 9, 2009
  - Fixed an issue when Jack OS X VST or AU plugins are closed after the internal Jack client
  - Updated to latest Jackdmp version, which addresses problem with some Digidesign cards
  - Jackdmp / Jack2 – 1.9.5 (SVN 3854)
  - JackRouter – 0.9.2
  - JackPilot – 1.6.8

- 0.84 – November 19, 2009
  - Changed requirement to have single duplex audio device specified in preferences; now specify Input and Output devices separately (eliminates Aggregate Device requirement for Intel Macs)
  - Added optional clock drift compensation mode when using Input and Output devices controlled by independent clocks
  - Added hog mode, to prevent other applications from changing sample rate of devices that Jack server is already using
  - Added notification that Jack server stopped when not using hog mode and another application changed sample rate on device(s) Jack was using
  - Fixed sleep/wake bug on Snow Leopard (10.6)
  - Fixed monitor port naming in JackAudioDriver and JackCoreAudioDriver
  - Fixed bug with potential race condition at startup
  - Jackdmp / Jack2 – 1.9.4
  - JackRouter – 0.9.2
  - JackPilot – 1.6.7

- 0.83 – September 22, 2009
  - Added support for OS 10.6 (Snow Leopard) with “64/32 bit” version

  - "32 bit" version for pre-10.6 systems, as well as 10.6 systems using audio devices with 32 bit driver
  - Jackdmp / Jack2 – 1.9.4
  - JackRouter – 0.9.1
  - JackPilot – 1.6.6

- 0.82 – March 23, 2009
  - Fixed "deadlock" issue on some machines
  - Increased the maximum number of Jack ports to 1024 (from 512)
  - Updated to the latest Jack server version
  - Jackdmp / Jack2 – 1.9.2
  - JackRouter – 0.9.1
  - JackPilot – 1.6.5

- 0.81 – February 05, 2009
  - Fixed issue in JackRouter that was causing "dirty buffers" to continue playing when audio was stopped in Max/MSP
  - Fixed "deadlock issue in Jackdmp server that could cause audio applications to hang
  - Updated to latest Jackdmp SVN version
  - Jackdmp – 1.9.1
  - JackRouter – 0.9.1
  - JackPilot – 1.6.5

- 0.80 – January 16, 2009
  - Fixed issue in JackRouter that arose with OS X 10.5.6; JackRouter can be default system audio device again
  - Fixed issue with JackPilot where available sample rates for each device are now retrieved more reliably
  - Updated to latest Jackdmp SVN version
  - Jackdmp – 1.9.1
  - JackRouter – 0.9.0
  - JackPilot – 1.6.5

- 0.79 – November 4, 2008
  - Updated to latest SVN version of Jackdmp, to fix CoreAudio driver to address a conflict with Audio Hijack application, and a problem with input-only devices
  - Updated JackRouter to fix an issue with iTunes when using jack with more than 2 virtual channels
  - Jackdmp – 1.90
  - JackRouter – 0.8.9
  - JackPilot – 1.6.4

- 0.78 – July 24, 2008
  - Updated to latest version of Jackdmp, to address problem with Ardour

  - Jackdmp – 1.90
  - JackRouter – 0.8.7
  - JackPilot – 1.6.4

- 0.77 – April 8, 2008
  - Many internal improvements in Jackdmp, including fix for bug that was causing JackPilot to crash when accessing certain ports
  - Fixed JackRouter to correct crashes with AudioFile Engineering's Rax and Apple iMovie
  - Fixed JackRouter to correct dropped samples when connecting to Quicktime Player
  - Updated documentation to mention that Aggregate Devices must be created while logged in with an Administrator account
  - Jackdmp – 0.71
  - JackRouter – 0.8.7
  - JackPilot – 1.6.3

- 0.76 – Feb 7, 2008
  - Corrected a flaw in how certain specific audio device state changes are handled by the Jack server
  - Added menu items for Audio MIDI Setup and Sound preferences pane
  - Jackdmp – 0.691
  - JackRouter – 0.8.6
  - JackPilot – 1.6.3

- 0.75 – Jan 23, 2008
  - Now using Jackdmp Jack infrastructure, instead of Jackd
  - Now compatible with OS X 10.5.x (Leopard)
  - No longer automatically selects Jack as Default Input/Output device if selected in JackPilot Preferences – must manually do so in Sound Preferences pane or in Audio MIDI Setup application
  - Jack VST plugin now VST 2.4-compatible
  - Previous iSight incompatibility fixed
  - Jackdmp – 0.69
  - JackRouter – 0.8.6
  - JackPilot – 1.6.2

- 0.74 – May 4, 2006
  - Updated Jack AU plugin to pass Apple AU validation

- 0.73 – Apr 13, 2006
  - Entire Jack OS X package, including plugins, now Universal Binary
  - Updated Jack server to most current version
  - Removed NetJack component from package pending critical bug fixes
  - Removed "portaudio" driver from package
  - Jack – 0.101.3

    - Jack Audio Router (JAR) – 0.8.3
    - JackPilot – 1.6.0

- 0.72 – Jan 27, 2006
    - Updated Jack server to most current version
    - Updated Core Audio driver to solve problems with some audio interfaces
    - Updated JackPilot to solve a problem with latest 17 inch PowerBooks
    - Updated Jack Audio Router (JAR) to support VideoLan player (VLC), and addressing various bugs, including memory leak
    - Jack – 0.100.6
    - Jack Audio Router (JAR) – 0.8.0
    - JackPilot – 1.5.9

- 0.71 – Jul 12, 2005
    - Fixed bug in JAR buffer management code that prevented Rax from working properly. Fixed a bug in JAR stream property management that prevented Plogue Bidule from working properly.
    - Jack Audio Router (JAR) – 0.7.7

- 0.7 – Jun 23, 2005
    - Now compatible with OS X 10.4.x (Tiger), including Aggregate Audio Devices
    - Only streams that are actually used now appear as Jack ports (e.g. iTunes only has outputs now, and no inputs)
    - Fixed JackPilot port scanning bug, so that ZynAddSubFX OS X port works properly.
    - Updated Jack server to most current version, 0.100.1
    - Documentation updated
    - Jack – 0.100.1
    - Jack Audio Router (JAR) – 0.7.6
    - JackPilot – 1.5.8

- 0.61 – Mar 1, 2005
    - Fixed Jack Audio Router to work with Skype, Rax, and other applications it was having trouble with. Related to a problem starting the JAR because of some C++ static variable initialization issues.
    - Added Arvid Tomayko-Peters' Autostart Applescript, to standard distribution, in Extras/Autostart folder

- 0.6 – Feb 21, 2005
    - Updated Jack server to version 0.99.49
    - Fixed Jack Audio Router to work with Native Instruments applications
    - Fixed Jack Audio Router to work with Final Cut Pro
    - Fixed Jack Audio Router to allow working with both Jack AU and Jack VST plugins at the same time
    - Added "blacklist" functionality to Jack Audio Router, allowing for some

clients to be rejected from being added. Currently, XQuartz (opened by X11) is the only client on the list. This is mostly for the upcoming Ardour OS X port
    - Fixed CoreAudio driver to fix a problem with multi-stream interfaces
    - New management of Jack plugin audio buffers (they are now allocated and managed in the JAR, not in the plugins), to address “dirty buffer” problem, especially in Logic
    - Documentation updated
    - Jack - 0.99.49
    - Jack Audio Router (JAR) – 0.7.2
    - JackPilot – 1.5.4

- 0.51 – Oct 7, 2004
    - Fixed Jack Audio Router to fix crash when using “Band-in-a-Box™” application; JAR did not deal with special characters nicely
    - Fixed AU and VST plugs to fix crashes when Jack buffer size less than hosting app buffer size
    - Fixed default Jack server buffer size to 512 – was set to 4096 by default in 0.5 version, which caused problems with iTunes not playing correctly
    - Jack Audio Router (JAR) – 0.5.8
    - Jack AU plugin – 1.5.1
    - Jack VST plugin - 1.5.1

- 0.5 – Sep 30, 2004
    - New CoreAudio driver (in addition to existing PortAudio driver)
    - New plugin architecture included in JackPilot
    - New NetJack UDP plugin for JackPilot
    - Workaround for “iTunes problem” – restores state of application’s connections if audio stream into Jack is stopped, then restarted
    - Various Jack Audio Router bug fixes
    - New Jack VST and AU plugin design – AU plugin now passes current Apple AU validation test (version 1.1.1b11 of validation tool)
    - CFM VST plugin no longer supplied in package – essentially all VST hosts now support Mach-O
    - Developer toolset added
    - Step-by-step illustrated examples added to documentation
    - Documentation updated and revised overall
    - Italian and French localization of JackPilot
    - Jack - 0.82.0
    - Jack Audio Router (JAR) – 0.5.7
    - Jack AU plugin – 1.5
    - Jack VST plugin – 1.5
    - JackPilot – 1.5

- 0.4 – initial public release – Jan 7, 2004 (originally called Jack Tools)
    - Jack - 0.82.0

- Jack Audio Server (JAS) – 0.4.5
- Jack AU plugin – 1.0
- Jack VST plugin (Mach-O) – 1.0
- Jack VST plugin (CFM) – 1.0
- JackPilot – 1.4