How to start

- Read the first page, then read about **VAC** principles and features.
- Be aware of the compatibility issues.
- Install the software.
- Read about simple usage ways.
- Try some examples.
- Read about advanced usage issues.

If you experience troubles, first try to find solutions in Troubleshooting, FAQ and How To sections.

Reading the manual, don't forget that there is the "Search" tab. Having a problem, enter some words related to the problem and appropriate pages will be displayed. Of course, search feature will not help if you enter a question like "how can I use it?". It only finds pages containing given keywords.

Please note that **VAC** is not a self-sufficient, easy-to-use application with an intuitive interface. **VAC** is a wide-range and powerful tool that works efficiently only if used properly. In particular, it is not for beginners. A beginner can successfully use **VAC**, especially having detailed instructions, but there is a risk of unexpected behavior due to improper usage. **VAC** is not a harmful tool; using it improperly, you simply will not get what you need. So please take care by reading instructions and following them.

If you want to use several cables at the same time, it is highly recommended to make a drawing graphically representing signal routing scheme you want to implement. The drawing should contain all signal sources and destinations, as well as transfer directions between them.

If this manual does not cover your issue, please contact the support.

What is Virtual Audio Cable 4

The **Virtual Audio Cable** (**VAC**) software implements an idea of a physical interconnection cable applied to Windows audio applications. Like traditional audio devices (Disk player, FM receiver, equalizer, amplifier etc.) are connected together by the electric (analog or digital) cables, Windows audio applications can be connected together by the **Virtual Cables** created by **VAC**.

VAC has Windows WDM/KS virtual device driver that creates a set of virtual audio devices named **Virtual Cables**. Each cable has a pair of audio ports, input and output. These ports are internally connected within each cable so all digital audio data sent (played) to the output port by "source" application are implicitly transferred to the input port and can be retrieved (recorded) by another (destination) application.

A **Virtual Cable** is similar to an ordinary audio adapter (a sound card) with its input and output externally connected between each other. Since a real adapter usually has a DAC (digital-to-analog converter) and ADC (analog-to-digital converter) in its signal path, an unneeded double digital-to-analog and analog-to-digital conversions are performed in case of such connection. Only rare and/or expensive adapters have a digital input and output that can be connected to transfer a signal clearly. **VAC** does such transfer itself, without any additional audio hardware.

Each port of each **Virtual Cable** is multi-client, allowing multiple applications to open the port at the same time. In the WDM/KS terms, it means each cable pin can be instantiated several times. All sounds (audio streams) coming to the playback port are mixed together, and each client connected to the recording port will get an individual copy of a signal via its recording/capture stream. Mixing and distributing are performed on the cable basis only, different cables are completely independent from each other.

As a WDM/KS filter driver, **VAC** represents a lowest-level audio layer that can be effectively used by any upper-level layer (KS-aware applications, DirectSound and MME subsystems, ASIO wrappers and so on).

A small utility, Audio Repeater, is included into the package to perform various useful tasks.

What you can do with VAC

- **Connect** two or more audio applications into a chain where each next application receives an audio signal produced by a previous application. For example, you can connect a player application to a sound processors and then connect a processor to an analyzer or a meter application to investigate the audio signal.
- **Intercept** the digital audio signal from applications playing it to MME/Wave, DirectSound or WDM/KS Audio ports. For example, you can connect Real Audio Player to Audacity and record any portion of played sound in real time, without a quality loss. Or you can use it to route input and output signals to/from Skype from/to some recording/playback applications.
- Record **bitperfect** digital audio data produced by applications that don't create WAV files, sending audio only to MME/Wave, WASAPI, DirectSound or WDM/KS Audio device in real time.
- Digitally **mix** several audio sources together and route resulting audio stream to a recording application.
- Digitally **distribute** (share) an audio stream among several recording applications.
- **Convert** audio data from one format to another in real time.
- Scatter some audio channels to a stream or gather some channels from a stream.
- Bring multi-client feature to any audio device that has not such feature.

VAC allows to interconnect almost all audio applications: audio/video players, software synthesizers, sound processors, audio editors, music sequencers, VoIP applications, instant messengers, speech recognition software etc. There are almost no limitations to use a **Virtual Cable** port instead of another audio device.

What you cannot do with VAC

- Render a **MIDI** sequence to an audio signal.
- **Secretly intercept** an audio signal coming from/to any another audio device unless a **Virtual Cable** is explicitly inserted into the signal path.
- Transfer an audio data over a **network**.

However, **VAC** can help you with these tasks if you use some third-party applications/drivers. For example, you can use a software synthesizer (for example, built-in Windows GS Synthesizer) that renders a MIDI sequence to an audio stream, and then use a **Virtual Cable** to receive this stream and route it to a recording application. And you can use a network application to transfer an audio stream over a network using a **Virtual Cable** to supply this application with a recorded data and/or to route its playback data.

Principles

It is like an electric cable

A **Virtual Cable** behaves like a real electric cable. In a traditional modular audio system, you need to connect the output of each sound **source** (deck, pickup, guitar or synth) to the input of each sound **destination/target** (amplifier, equalizer, gate, processor etc.) by its own electric cable. If one component's input is connected to other component's output, the first can receive an audio signal from the second. Otherwise, it cannot.

Computer sound system works very similar. There are two kinds of objects that can be sound sources or destinations: **applications** that generate, play or record various sounds (WinAMP, Skype, Audacity, Cakewalk, Sound Forge, etc.) and audio devices (sound adapters/cards). Applications can transfer sounds using device audio ports; typical sound adapter has one recording (capture/input) and one playback (render/output) port. From a recording port an application can record sounds, and can play sounds to a playback port.

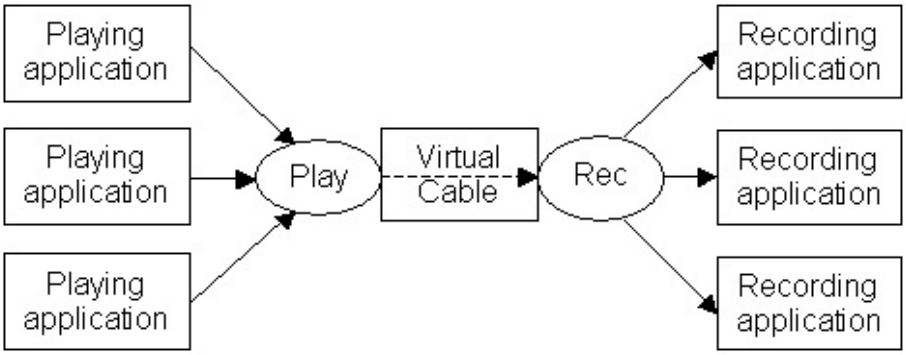
Some audio devices have multiple recording/playback ports, or a single recording/playback port only. For example, USB microphone adapter has a single recording port and USB speakers adapter has a single playback port.

In native Windows audio subsystem, there is no way to internally connect an application producing a sound to another application recording and/or processing this sound. For example, you cannot directly transfer a sound from Skype to Audacity. You need a full-duplex sound card with its line output jack connected to its own input jack, and there will be two digital/analog transforms, affecting sound quality and volume level, and there will be high latency (recorded sound arrives about 100-300 ms later than original sound was played).

Cable and application interconnection

VAC looks like a set of real full-duplex sound cards with their digital

outputs hardwired to their digital inputs. Like a typical sound card, each **Virtual Cable** has two ports: playback and recording. Source application produces (plays) the sound, sends it to **Virtual Cable** playback port, **VAC** driver immediately transfers it to the recording port of the same **Virtual Cable**, and destination application receives the sound from this port:



Recording and playback sides

Keep in mind that a **Virtual Cable** is **unidirectional**. Only the playback side can accept audio data, and only the recording side can produce it. The playback side is a "**data sink**" and the recording side is a "**data source**".

At a beginning level, you can imagine a **Virtual Cable** as a kind of a **pipe**. When you direct a sound into your pipe end, other party can hear it from another end. But due to Windows audio principles, the pipe is **unidirectional**. That is, you can send a sound only to a playback end, and receive it only from a recording end.

Full independency

You can use **VAC** even without a real sound card in your system; but there will be no real sound to your ears, only digital sound in your computer.

Remember that **VAC** does not **produce** any sounds (except of trial voice reminder), it only **transfers** sounds produced by applications. With **VAC** only, as with simple electric cable alone, you will hear nothing. You need a **complete sound signal path**, i.e. source and target applications to

transfer sounds, and a real sound adapter, either internal or external, to hear it or record from the real world. You also must make proper **connections** between your sound system components, otherwise your goals will not be achieved.

Possible usage methods

Using **VAC** and Audio Repeater, you can route an audio stream coming from any audio port to many applications at the same time, sharing the real audio port. You can also mix audio streams from several applications together, routing the result to any audio port. But **VAC** is not primarily intended for such purpose; it is better to use the Wave Clone software.

Features

- Windows 5.x and 6.x platforms (32-bit and 64-bit). May work in Win2000 but not tested.
- Up to 256 virtual cable devices (some systems limit number of MME devices).
- 1..20 milliseconds per interrupt.
- 1..100 pin instances.
- Almost any of **fixed point** PCM audio formats (1000..1000000 samples per second, 8..32 bits per sample, 1..8 channels). Floating point formats are not supported.
- Almost no sound latency with maximal interrupt frequency.
- Unlimited number of clients connected to each port.
- Signal mixing (with saturation) between output port clients.
- PCM format conversion (sampling rate, bits per sample, number of channels).
- Volume control features (both attenuation and boost).
- Channel scattering/gathering mode.
- Watermark control technique to improve stream stability with unstable applications.
- Control Panel application to configure cables and watch their state.
- Audio Repeater application that transfers from any recording to any playback device.

Compatibility issues

There are some compatibility issues preventing VAC from working in particular cases.

Trial version limitations

- Supports no more than three cables.
- A female voice repeatedly reminds that it is a trial.

Trial version of **VAC** allow you to see all features of the full version. To prevent professional usage, it adds a voice reminder (-6 dB level) to an audio signal. Voice reminder is not applied during first several minutes after driver startup (Windows boot or driver restart without rebooting), then it is applied at a regular basis, each several seconds. If you need additional several reminder-free minutes to perform some testing, restart the driver and run your tests immediately.

Please note that voice reminder is just **added** to cable audio signals, not replaces them. If you are recording from a cable and hear nothing except voice reminder, it definitely means there is no useful signal in the cable because it is not supplied with a signal, and you should check audio application's settings. Vice versa, if you have installed trial version, recorded from a cable within several minutes and didn't hear the reminder, it definitely means that you have recorded from another device, not from a cable.

Compatibility issues

Remote access

Windows has only a limited audio device support in native Remote Desktop connection so **Virtual Cable** devices may not be accessible via RDP connections.

If target (remote) system is a workstation (Windows 2000 Pro, XP, Vista, Win 7), Remote Desktop session is established as a direct/console session too. A local user, if any, is forcibly logged out and you are logged in instead, having his/her local desktop. To access **Virtual Cable** devices via MME/DirectSound interfaces and control **VAC** driver from the Remote Desktop connection, you should specify "Leave at remote computer" for the "Remote computer sound" at the "Local resources" tab in the "Remote Desktop connection" dialog. When you request a connection, a simplified form of this dialog is present, containing only computer and user names. To view the "Local resources" tab, expand the dialog by clicking the "Options" button.

If target system is a server (Windows 2000/2003/2008 Server etc.), it creates a **separate** logon session by default. A local user is not disturbed by a remote connection. Only a default local audio device is redirected to the new session. There is no way to use and control other local audio devices in such connection. To access them, you should connect to a "console" session like workstation systems do. It is possible using **mstsc** command with the "**/console**" option. In XP SP3, and Windows 6.x, the **/console** option is replaced with **/admin**.

If the host runs 2003 Server system and there is no "RDP Audio" playback device in a remote session, please check Terminal Services configuration at the server (Start - Services - Control Panel - Administrative Tools - Terminal Services Configuration). At left pane, click **Connections**, right-click **RDP-TCP** at right pane and click **Properties**. Open **Client Settings** tab and make sure that **Audio mapping** in **Disable the following** group is **not** checked.

In Server 2008, there is **no way** to connect to a console session via RDP.

To see local audio devices, you should use desktop sharing software.

See Windows Help for details.

The restrictions above are related only to most popular MME/DirectSound/WASAPI interfaces. Using Kernel Streaming interface, you can access remote WDM audio devices from any session type.

Please note that in most cases there will be no audio signals passed over the network between local and remote computers. Even when you use a KS application that sees **Virtual Cable** devices, all audio operations are performed on a remote computer, no audio data are transmitted to or from a local computer. It is similar to a situation if you are using **VAC** on a computer without a hardware audio adapter.

Virtualized environment

As a kernel-mode driver, **VAC** may not work in some virtualized environments.

Partial virtualization is enough to use most user-mode software (Web and other network servers, remote desktops, distributed calculations) in a virtualized environment. For user-mode applications, virtualization software maintains several separate environments. But there is only a single system kernel so in most cases you cannot install a kernel-mode driver like **VAC** on such environment. In most cases, the system warns you but in some cases the installation might be successful but the driver will not be loaded.

If the environment is virtualized partially (at user-mode level), you cannot install **VAC** under it. But if you have full access to the host machine, you can install **VAC** on the entire host, and then all virtualized environments will be able to use **VAC**.

In a fully virtualized environment, you have a completely separate OS, with its own kernel, so you can separately install and use **VAC** on each virtual machine of this type.

System requirements

VAC system requirements are very simple:

- Any hardware configuration (CPU, memory size etc.) enough for a supported operating system.
- Microsoft Windows 5.x or 6.x 32-bit or 64-bit operating system.
- For a virtualized environment, please see compatibility issues.
- For a remote connection, please see compatibility issues.

Installation

If distribution files are packed, first unpack the archive to an empty folder on your local hard disk. The original package is a ZIP archive and you can unpack it using Windows internal ZIP pseudo-folder support, simply "entering" into the package, selecting all files and folders, and then copying them to an empty folder. Please don't try to start installation right inside a ZIP pseudo-folder, it will not work.

Alternatively, you can unpack the package using 7-Zip, IZArc or similar unpackers.

Please take care to preserve **subfolder structure** of the package. It is not recommended to unpack the package to an encrypted and/or network folder and/or disk, use a plain disk/folder is possible.

Creating an empty folder to unpack the installation package, use a simply named folder that does not contain unusual characters in the path.

- Uninstall all installed versions of **VAC 4**. It is **very important**. Previous major versions of the **VAC** (1.xx, 2.xx, 3.xx) will not interfere with **VAC 4** and you don't need to uninstall them if you want to use several different versions.
- Read the system default device issues section.
- If UAC is not enabled in your Windows 6.x system, explicitly log to the Administrator account.
- If you need more than 16 **Virtual Cable** devices, read the get more than 16 virtual cables section.
- Run the **setup.exe** file from the package directory and follow instructions displayed.

If you have some previous major (2.xx or 3.xx) version of **VAC** installed, make sure you have chosen different installation folder and **Start Menu** folder. Different major versions of **VAC** can coexist on the same system but they need to be installed to **different** places.

If you are installing under Windows 5.x system and there is at least one WDM audio device in your system, **VAC** installer uses a lightweight

installation mode that is silent and you will see no system warning messages. But if your system has no WDM audio devices or you are running Windows 6.x, installer performs a compatible-mode installation. In such case, the system warns you that the driver has no WHQL signature. The system also will warn that product publisher is NTONYX and ask whether you trust it. Let the system install the driver.

Under 64-bit Windows 6.x, kernel-mode drivers require a publisher's digital signature to be loaded. **VAC** driver is signed by a certificate issued to NTONYX, one of **VAC** resellers. The author cannot sign the driver personally because Microsoft introduced a strange security policy that does not allow an individual developer to obtain a publisher certificate valid for driver signing.

After successful installation, you will receive a confirmation dialog. If there were no errors, Windows restart is not required. **VAC** becomes functional immediately after the installation completes. If Windows cannot safely replace system files during installation, a reboot is required to make **VAC** active.

Performing the installation, **Setup** creates a log file in Windows temporary directory. This file reflects all operations performed by the installer. If installation succeeds, log file is moved to an installation directory and renamed to "**install.log**". Saved log file may be needed if some problems will appear further. If installation fails, **Setup** prompts you to keep the log file in a temporary directory. If you experience a problem with installation, please keep the file and attach it to your support request.

As a result of the successful installation, a new system-wide device named **Virtual Audio Cable** appears in the Sound/Multimedia device group of the Device Manager.

In Windows 5.x, the driver exposes a set of audio (waveform) ports named **Virtual Cable N** (**N** is cable number). Unlike previous versions of the **VAC**, **In** and **Out** ports have the same names, as usual in modern Windows versions. It is easy to distinct them because all applications place input and output device names to different lists.

In Windows 6.x, due to endpoint mechanism, **Virtual Cable** devices are

named differently: "**XXX N (Virtual Audio Cable)**" where "**XXX**" stands for a source line name and "**N**" stands for a cable number. The endpoint idea is useful for real audio devices having multiple input lines but is meaningless for a **Virtual Cable** device that has only a single audio path. But some applications like **MSN Messenger** try to locate a particular source line (for example, microphone or line input) input so **VAC** supports them, exposing **fake** "inputs lines". By default, only a single source line is exposed, named "**Line**" (an "analog line input"). At a cable playback end, **VAC** also exposes the "**Line**" endpoint (an "analog line output"). So, by default, there will be a single input and a single output waveform audio port for each **Virtual Cable**.

Totally, **VAC** can expose up to three source lines from each cable: Microphone input ("Mic"), line input ("Line") and S/PDIF input ("S/PDIF"). See the Configuration chapter for details.

If you don't see **Virtual Cable** devices in audio device menus and are absolutely sure that it is not related to a known issue (for example, an inappropriate remote connection or virtualized environment), please try the manual installation.

The setup process also creates **Virtual Audio Cable** (if you have not modified this name) folder in your **Start/Programs Menu**. It contain a shortcuts to the Audio Repeater, VAC Control Panel and other useful links. If the **All Users** menu (**Start Menu** folders for all users at this computer) is available, the **VAC** folder created here. Otherwise, the folder is created for the currently logged user.

Now you can begin using the **VAC**.

Manual driver installation

Manual driver installation may be required if **VAC** installer was not capable to install all the driver functions automatically. In such cases, the installer reports successful installation and you see a working **Virtual Audio Cable** device in Device Manager but you don't see **Virtual Cable** devices in audio application device lists or cannot get them to function properly.

Manual installation should be performed only if **VAC** proprietary installer reported that the installation was successful but **Virtual Cable** devices are not visible or do not function. If **VAC** installer reports an error, don't perform a manual installation but ask for a support.

Also please make sure that there is no other known issues (for example, an inappropriate remote connection or virtualized environment).

To install **VAC** manually, perform the following actions:

- Start **Windows Hardware Wizard** (the **Add Hardware** item in the Windows Control Panel).
- Specify that the hardware is already connected.
- In a device list, select "**Add a new hardware device**".
- In a next form, choose "**manually select from a list**".
- In a device type list, choose "**Sound, video and game controllers**".
- In a manufacturer list form, click "**Have Disk**" button.
- Browse for the folder where **VAC** installation files are unpacked.
- In a device list, choose "**Virtual Audio Cable**".

You will be warned that the driver is unsigned. Proceed the installation.

When manual installation is finished, you should see **Virtual Cable** devices. If they still are not visible, please ask for a support.

Do not perform manual installation before installing **VAC** using its setup utility. During manual installation, only driver files are installed while **VAC** setup utility installs all **VAC** files, such as VAC Control Panel application. Manual installation can be performed after running **VAC** setup utility but not vice versa. If you have performed manual installation before running **VAC** setup utility, you must use Advanced Setup Mode to overwrite driver files and over-install the driver. Such sequence is not recommended.

Advanced setup mode

Setup utility works for an **ordinary** user by default. In this mode, any abnormal situation causes abortion of installation process. If you are an **experienced** user and can understand some abnormal situations (for example, when file to be created is actually present), you can select the "**I am an advanced user**" checkbox. In such case, you will be prompted if some files, devices and/or services already exists and may be overwritten. Use this feature with extreme care. In general, there are no need to overwrite anything during normal installation process. You need an advanced mode only if you have incorrectly uninstalled **VAC** or tried to install it over the already installed one.

When the general installation mode fails, you can try the advanced mode.

First, locate a directory and a **Start Menu** folder where **VAC** files/links are installed, and delete them. Start the setup utility and check the "I am an advanced user" item. When prompted to overwrite existing files and replace service and/or device descriptors, answer Yes/Ok.

At first phase, the installer may be unable to replace all objects needed, and installation process may abort again. In such case, reboot the system and run the installation in advanced mode again. If prompted for overwriting some files, answer **Yes**. At this second phase, installation must succeed. If it aborts again, request for a support.

Simple cable usage

To use a cable simplest possible way, first run VAC Control Panel application and make sure that there are 1-3 cables and they are configured with typical parameters:

- Maximum instances - 20
- Milliseconds per interrupt - 5..7
- Sampling rate range - 22050..48000
- Bits per sample range - 8..16
- Number of channels range - 1..2
- Stream format limiting - Cable range
- Volume control - disabled
- Channel mixing - enabled
- PortCls usage - disabled

Don't close VAC Control Panel, it will help you later.

Run any audio producing application (a player, a tone generator, an audio editor) that allows you to specify a playback/render device directly and choose **Virtual Cable 1**.

Start playback and make sure that the application really plays to **Virtual Cable 1**. You should hear nothing but VAC Control Panel must show 1 or 2 new playback streams for **Virtual Cable 1** and cable signal level indicator should be shown. If you hear a played sound, it means the application directs it to your hardware audio device, not to **Virtual Cable 1**.

If a sound is really played to the **Virtual Cable 1**, run Audio Repeater application and choose **Virtual Cable 1** as a recording (Wave In) device and your hardware audio device as a playback (Wave Out) device.

Activate audio transfer clicking **Start** button in Audio Repeater. Now you must hear a sound played by an application started first and the Control Panel application must show a new recording/capture stream. It means that audio data are transferred from a playing application to **Virtual Cable 1** and then Audio Repeater transfers them further to a hardware

audio device. You have created a full audio path containing **Virtual Cable 1**.

You may hear some quality loss in comparison with a direct playback to the audio card. It may occur due to a format conversion. To improve sound quality, you need to play with client and cable audio formats. See advances usage topics section for details.

To record a signal transferred via the cable, run any recorder application. Choose **Virtual Cable 1** as a recording device and start recording. VAC Control Panel may show a new recording/capture stream indicating that a recorder application has connected to **Virtual Cable 1**. If recording is performed via System Audio Engine, no additional recording streams will be shown.

To hear a signal during recording, you need to keep Audio Repeater active. Otherwise, a signal is transferred from a player application to a recorder application only. Starting audio transfer in the Audio Repeater, you create another connection to the **recording** side of the **Virtual Cable 1** that splits an audio signal into two identical streams.

You can connect any number of playing/recording applications to each end of each **Virtual Cable**.

Remember that **VAC** itself only transfers audio data internally from its **playback** side to its **recording** side within each **Virtual Cable**. **VAC** does not route any sound to/from your hardware audio devices and don't intercept any signals that are not explicitly routed to **Virtual Cables** because **VAC** has no relation to other audio devices at all.

To configure driver and cable parameters, use the VAC Control Panel application.

If you have troubles using **VAC**, try to look at examples/troubleshooting sections and FAQ/How to... lists.

More complicated tasks

Virtual Cables can be used much more complicated ways. See the

advances usage topics section.

Advanced cable usage

This section describes **advanced** usage methods. You need to understand **VAC** principles and simple usage rules before attempting to use advanced features.

Choosing most appropriate interface

As a WDM/KS Audio filter, **VAC** driver presents a pair of render and capture pins for each cable. Using these pins, Windows audio subsystem builds higher-level audio layers (see Audio layering issues). So, each **Virtual Cable** presents at least three audio interfaces: WDM/KS, DirectSound and MME (Wave), with a pair of recording/playback ports in each layer. In Windows 6.x, there will be a fourth interface, WASAPI.

The following list shows audio interfaces in the order of increasing efficiency:

- WDM/KS
- DirectSound with hardware acceleration (Windows 5.x) or WASAPI in the exclusive mode
- MME or WASAPI in the shared mode

To get higher performance and less latency, use a lowest-level possible interface type. To get higher compatibility and less problems, use a highest-level possible interface type.

Using WDM/KS or WASAPI in the exclusive mode, keep in mind that they does not support pin instance sharing. More than one connection to a specific pin is available only if a driver supports multiple pin instances. **VAC** supports them but not all other audio drivers do.

Using WDM/KS interface, please note that it contains a huge set of features (properties) but some audio drivers, especially third-party ones, are tested only with a limited set of them. Some properties are not documented well so driver developers may not be able to meet all the requirements. Incorrect driver behavior may cause application malfunction or even system crash. Trying a new application, new audio

device or new driver version, first test them a lot in a minimal environment, prior to performing an important work.

Adjusting DirectSound hardware acceleration level

In Windows 5.x, **VAC** can be used via the DirectSound interface much more effectively if its hardware acceleration feature is utilized. To enable Windows to utilize it, you must adjust the hardware acceleration level for the **VAC** device.

In Windows 6.x, DirectSound hardware acceleration is not supported.

Stream and cable format limiting

Due to audio layering specifics, audio applications have no full control over audio formats actually used by devices for playback or recording, except of exclusive mode access. In Windows 5.x, recording operation always requests a specified format from the driver, but playback operations in Windows 5.x try to request a most wide format supported by the device for System Audio Engine.

For example, if a DirectSound application requests playback at 48000/24/2, DirectSound subsystem first requests 44100/16/2 from System Audio Engine. If a **Virtual Cable** is free and this format is within the cable format range, 44100/16/2 will be chosen for the cable format. A second render stream will be created with 48000/24/2 but the cable format cannot be changed when a cable is used. Therefore, **VAC** will perform a format conversion from 48000/24/2 to 44100/16/2 and signal quality will degrade. To minimize format conversion, you could limit a cable format, or a stream format, or both of them.

Cable format limiting is useful to specify most convenient format range for both cable sides. For example, if you want to use 48000/16/2 at one side and 44100/24/2 at another side, you could set cable format range to 44100-48000/16-24/2-2. It prevents the cable format from locking at low sampling rate and/or bitness due to implicit pin instance allocation for System Audio Engine.

By default, cable format range is restricted to very common range for

better compatibility. If you want to use wider range, you can extend the format range using the VAC Control Panel.

DirectSound and MME subsystems often can try some consecutive formats if the driver does not allow them to use their preferred format parameters. A driver cannot suggest to higher-level layers to use a particular parameters but it can reject some formats and allow some others. So you could use various stream format limiting modes:

- **None** - there is no stream format limiting over the common driver features. So a new stream can be created using any format supported by the driver. A format conversion will be performed in most cases.

- **Cable range** (default) - a new stream can be created only if its format is inside the cable format range. A format conversion is performed in some cases.

- **Cable format** - a new stream can be created only if its format exactly matches the current cable format. If a cable is free, the "**cable range"** mode is used for a first stream (see format selection rules for details). Therefore, no format conversions are performed because all cable streams have the same format. This mode is similar to previous (2.x, 3.x, 4.00 betas) versions behavior.

As an extreme case, you can set the same lower and upper ranges for each cable format parameter and specify stream format limiting mode to the "cable format". Therefore, only a particular format (for example, 96000/24/2) will be allowed for client streams. If both side clients are able to choose such format, no conversion will be performed. All audio data will be transferred through the cable unchanged, with no quality loss.

But don't forget that layered scheme prevents applications from having full control over their audio activity. Some implicit format conversion may occur in System Audio Engine and there is no way to easily check it.

Format conversion issues

VAC 4 supports a format conversion. But due to mixing purposes, **VAC**

cannot directly convert from a playback client's format to a recording client's format. Instead, **VAC** first converts data from various playback client formats to the cable format then mixes client data using the cable format. Finally, **VAC** converts mixed data from the cable format to various recording client formats.

The cable format parameters are selected when a first playback or recording client connects to a given **Virtual Cable**. Sampling rate, bits per sample and number of channels are chosen from the cable format range to be possibly close to a client's format parameters.

For example, if a client requests 44100/16/2 and the cable format range is 22050-48000/8-32/1-8, the cable format is set to 44100/16/2 (equal to client's format). If the cable format range limits sampling rate to 22050-32000, cable sampling rate will be set to 32000. If it is limited to 48000-96000, resulting value will be 48000.

If you don't know which format is used by a particular application for playback or recording , run the VAC Control Panel. It shows a current cable format for each cable.

For multichannel (4.1, 5.1, 7.1 etc.) formats, don't forget to adjust the **Virtual Cable**'s default format for shared-mode access (Windows 6.x) or speaker configuration (Windows 5.x).

If cable channel mixing is enabled and cable and client formats have different number of channels, only typical channel conversion schemes for 1, 2, 4, 6 and 8 channels are implemented. In other combinations, all source channels are mixed together and then distributed to all destination channels.

To extract particular channels from a stream or add them, use channel scatter/gather feature.

Try to minimize format conversion chances. A conversion is performed by the main CPU and takes significant CPU resources and it may noticeably slow your applications. When used by many clients at a same time, it may even hang your computer because the conversion is performed in kernel-mode which has higher priority than every application in the system.

Volume control issues

VAC supports **volume control**. For each playback and recording stream, volume and pan can be controlled via appropriate interface methods. For the entire cable, a main volume, pan and mute can be controlled, either via an audio mixer interface or the **Windows Audio Mixer Control** application.

To use cable volume control features, you must enable **Volume Control** in the cable configuration parameters.

Volume control limits are -40..+12 dB.

To control cable volume parameters, use Windows Volume/Mixer Control tool:

- In Windows 6.x, right-click speaker icon in the system tray, select **Open Volume Mixer** then click the arrow under device name/icon and select a desired device. Clicking on device icon opens device properties applet; advanced controls can be found on the **Levels** tab. Another way is to customize a fast volume control window by selecting **Volume Control Options** from the speaker icon context menu and then use a single click to open the fast control window.
- For Windows 5.x: click **Start** - **Programs** - **Accessories** - **Entertainment** - **Volume control**. Another way is right-clicking the speaker icon and choosing **Open Volume Control**. In **Windows Audio Mixer Control** window, open **Options** menu and select **Properties**. In **Mixer Device** item, select a proper **Virtual Cable** device and click **OK**. The Mixer will switch to show **Virtual Cable N** volume controls. The first group, "**Volume Control**" represents the main cable controls. "**Wave**" group adjusts waveform stream volume and "**SW Synth**" group adjusts a system-created software MIDI synthesizer channel volume. Other controls are shown for compatibility purposes and don't affect cable operations.

VAC supports only playback (output) volume controls because recording (input) cable side is connected immediately to output side and there is no difference where to change volume level. So recording volume controls for **Virtual Cable** devices have no effect.

Like format conversion, volume control takes some CPU resources and affects sound quality. Use it only when really needed.

Clock correction feature

When a **Virtual Cable** is used in a path containing another audio device (a hardware device, a software device other than **Virtual Cable** and even another **Virtual Cable** device), clock difference effect occurs. To minimize it, change cable clock correction multiplier until cable internal clock speed becomes closest to another device clock.

Mixing, format conversion and sound quality

Mixing of output streams is implemented with a saturation (clipping). If the amplitude value exceeds a maximal range defined by the sample bitness, it is clipped to the maximal value.

VAC uses relatively simple linear resampling algorithm and does not use dithering or other advanced smoothing features. Therefore, conversion results may sound worse than an original signal. To prevent quality degradation, you need to pay attention to a format matching. In an ideal case, all three formats (the playback client's format, the cable format and the recording client's format) are the same.

If all formats are equal and cable volume control is disabled, **VAC** performs no conversion, all samples are copied directly, and only mixing is performed if there are more than one playback clients. In such case, no quality degradation occurs (a bitperfect transfer is performed). If you play a WAV file to the playback side and simultaneously record data from the recording side, recorded data will be equal to played data, except of possible leading and/or trailing silence.

But such perfect results can be achieved only if all used formats are **the same**, volume control is **disabled** and audio data path is clearly understandable. In Windows 6.x, it is hard to clearly understand audio data path unless WDM/KS interface is used.

Start recording first

Due to layering issues, Windows 5.x audio subsystem in shared mode propagates only recording audio format on a first device connection, when a common pin instance is created for System Audio Engine. If a first connection issued to a **Virtual Cable** device is a playback request, shared format selection will be more complex and less predictable. So in Windows 5.x, it is better to start a recording operation with **Virtual Cable** before you start a playback operation on the same cable. It will fix the cable format and further playback operations will not change it.

An "operation" stands for an **actual** recording/playback process, not running an application. Some applications have a special button and/or menu command to start an operation. But some other applications can start an operation when they want. For example, Skype starts them when a peer connection is established.

In Windows 6.x, you can explicitly choose a default format for shared mode operations.

To watch the cable format evolutions, use the VAC Control Panel.

Speaker/channel configuration for multichannel formats

To properly transfer multichannel audio data under Windows 5.x, you must configure the Speaker Configuration for a particular **Virtual Cable** first. **Virtual Cable**'s Speaker Configuration must match the channel distribution of the audio format. Adjusting the Speaker Configuration is similar to adjusting the DirectSound Hardware Acceleration Level:

- Under Administrator account, open Audio Properties applet.
- Open the **Audio** tab, select an appropriate **Virtual Cable** device in **Sound Playback** listbox and click **Advanced** button.
- On the **Advanced Audio Properties** page, open **Speakers** tab, select a configuration and then click **OK**.
- On the **Sounds and Audio Properties** page, click **Cancel** (if you will click **OK** here, your default device will be changed).

In Windows 6.x, the **Configure** button is not available for cables with disabled speaker-type pins (default setting). Instead, select a default format at the **Advanced** tab in the device properties dialog. You can

enable speaker pin type for some cables but all these cables will have the same "**Speakers**" playback endpoint name so it would be hard to distinguish between them.

Scattering/gathering channels in multichannel streams

Mixing cable client streams to cable sound and distributing cable sound among client streams, **VAC** processes audio channels by three possible ways:

- **Convert** a source channel set to a destination channel set using common summation/distribution schemes. For example, to convert two stereo channels to a single mono channel, left and right channels are summed together. This mode is used if the Channel Mixing is enabled for the cable.
- **Scatter** a set of less number of channels **to** a set of more channels. Channels packed adjacently in the source stream frame are distributed (scattered) as suggested by source channel distribution mask.
- **Gather** a set of less number of channels **from** a set of more channels. Channels placed in source stream frame as specified by destination channel distribution mask are compacted (gathered) to be adjacently packed in the destination stream frame.

Scatter/gather modes are used if the Channel Mixing is disabled for the cable. Internal cable mixing buffer in the cable format always has all channels placed sequentially. So scattering can be performed either from a playback stream having less channels than in cable format, or to a recording stream having more channels than in cable format. Vice versa, gathering can be performed either from a playback stream having more channels than in cable format, or to a recording stream having less channels than in cable format.

For example, the stream has stereo format (two channels) and mask 0x410 (BL+SR) while cable format has 7.1 format (eight channels). If the stream is a playback one, two stream channels will be scattered to eight cable channels: left stream channel will be mixed to **BL** cable channel, and right stream channel will be mixed to **SR** cable channel. If the stream is a recording one, two stream channels will be gathered from eight cable

channels: left stream channel will be taken from **BL** cable channel, and right stream channel will be taken from **SR** cable channel.

In other words, there can be two data transfer paths: from audio application (source) to Virtual Cable (destination), or from Virtual Cable (source) to audio application (destination). If source stream has less channels than destination stream, scattering occurs. If source stream has more channels than destination stream, gathering occurs.

Please note that channel reordering is not supported. Scattered/gathered channels are always packed in the same order as source channels.

Scattering/gathering is performed only if cable and stream channel masks are overlapped (bitwise AND is non-zero). Otherwise, direct transfer is performed (first source channel copied to first destination channel and so on).

Most applications don't allow to explicitly specify channel distribution masks; instead, they assign a standard mask based on the number of channels. To use scatter/gather feature, you need to use KS version of Audio Repeater or develop your own application accessing **Virtual Cable** pins via WDM/KS interface.

See the example demonstrating scatter/gather feature usage.

Using stream buffer watermark technique

Watermark control technique is intended to partially compensate application buffering problems and, in some cases, clock difference problems.

By default, **VAC** driver processes stream audio data continuously at a constant speed, transferring a fixed amount of data on each timer event. If a client fails to provide data/room at a good time, an overflow/underflow occurs. If the client discovers this situation and quickly restores data flow, providing amount of data/room with a reserve, data processing continues normally. But if the client continues being late with providing data/room for the stream, overflows/underflows occur repeatedly and the only way to restore a stable flow is to stop and restart client application activity.

If watermark control is enabled, **VAC** does not perform audio data transfer from/to client buffer until the client provides an amount of data (for rendering streams) or empty space (for capture streams) equal to or more than the high watermark value specified (a reserve of data/space is created). Then **VAC** starts data transfer that consumes this reserve. If the client provides additional data/space in time, the stream goes smoothly. If amount of data/space available drops down to the low watermark value specified, **VAC** suspends data transfer until the high watermark is reached again.

Both low and high watermark values are specified in milliseconds so the "amount of data/room" means an amount of audio data that is played/recorded within the specified time. There is no need to calculate number of bytes, samples or frames.

The "Max wait" parameter prevents **VAC** from waiting for data/room very long. If data processing is suspended, **VAC** waits for data/room no more than specified by this parameter (also in milliseconds) then resumes data processing regardless of actual amount of data/room.

Using stream data buffer

For each stream, **VAC** driver creates an internal data buffer used for some technical purposes. In some cases, this buffer can be used to compensate bad data buffering algorithm used by client application or System Audio Engine, holding some audio data while a client fails to provide data buffers in time. Please note that this buffer cannot compensate clock difference.

You can control buffer length by Control Panel application, separately for each cable.

By default, length of this buffer is set to several dozens of milliseconds that is enough for most cases. To get a smallest possible latency and overhead, you can set it to **Auto**; **VAC** driver will use a smallest possible size and bypass this buffer when possible. But please note that this buffer is not directly linked to latency time, there is only a small correlation. Latency time is much more sensitive to event period time.

In very rare cases, you may get a profit from increasing buffer length up to hundreds of milliseconds.

Please note that this buffer can compensate delays and buffering failures only for immediate driver clients, not via intermediate layers. For example, an application using KS is always an immediate client while applications using MME, DS or WASAPI are clients of System Audio Engine, not of **VAC** driver. But in some cases, System Audio Engine itself uses small buffer sizes and/of amount of buffers so its stability can be improved too.

Using PortCls data processing engine

For performance and stability reasons, **VAC** uses its own stream data processing engine, partially bypassing the code provided by the standard Windows PortCls driver. But **VAC** internal data processing engine is not exhaustive tested yet so there may be some problems with it. In case of such problems, you can disable internal **VAC** engine and use PortCls' one. In such case, you may need to restrict system threads affinity to partially avoid known PortCls problems.

Get more than 16 Virtual Cables

By Windows design, number and names of available audio devices must be specified at device installation stage. The driver cannot create more devices than were described during the installation. In **VAC** installation, number of described devices is limited to 16 by default for two reasons:

- Installation of more possible devices takes more time but most users don't need many cables.
- Creation of many (more than 10-15) audio subdevices in the driver causes very high system load and the system may almost hang for several minutes.

If you need more than 16 cables, replace the device/driver description file (**vrtaucbl.inf**) and catalog file (**vrtaucbl.cat**) with files contained in "**256cables**" package subfolder before the installation. If **VAC** is already installed, uninstall it, unpack the distribution package, replace the files and install **VAC** again.

Increasing number of cables, be aware of system limitation and overhead issues.

Driver and cable configuration

VAC has several configuration parameters. Some of them concern the entire driver and others are cable-specific, which can be different for each cable.

Full version of **VAC** can be configured to achieve any valid number of cable devices. Trial version is restricted.

Use the VAC Control Panel to easily configure driver and cable parameters.

VAC parameter tree is stored in the registry under the following key:

HKEY_LOCAL_MACHINE\SOFTWARE\EuMus Design\Virtual Audio Cable\4

Driver parameter values are stored immediately under this key, cable-specific parameter values are stored under "**\Cable N**" subkeys. All parameters are specified as **DWORD** values unless explicitly specified. Value names are the following (literally):

Driver parameters

- **Number of cables** (1..256).
- **Number of cables to restrict KMixer thread affinity** - number of cables beginning from which **VAC** will restrict affinity of System Audio Engine worker threads. If actual number of cables is greater or equal than this value, thread affinity will be restricted to run only on a first (number zero) CPU/core. It eliminates concurrent PortCls request processing and will solve some problems like this. This parameter is ignored if PortCls engine is not used.
- **Number of cables to restrict client thread affinity** - number of cables beginning from which **VAC** will restrict thread affinity of any client process (including System Audio Engine). It may be useful to solve problems like this. This parameter is ignored if PortCls engine is not used.
- **Maximum worker threads** (0..32 in 32-bit systems, 0..64 in 64-bit

systems).

- **Maximum worker thread priority** (0..31).

Cable parameters

- **Maximum instances** (1..100).
- **Milliseconds per interrupt** (1..20).
- **Minimum sampling rate** (1000..384000).
- **Maximum sampling rate** (1000..384000).
- **Minimum bits per sample** (8..32).
- **Maximum bits per sample** (8..32).
- **Minimum number of channels** (1..8).
- **Maximum number of channels** (1..8).
- **Stream format limiting** (0 - "**None**", 1 - "**Cable range**", 2 - "**Cable format**").
- **Volume control** (0 or not present - disabled, nonzero - enabled).
- **Source line types mask** - a bit mask that represents cable source line set. Bit 0 (value 1) means a microphone line, bit 1 (value 2) means an analog line connector, and bit 2 (value 4) means a digital line connector. You can combine them by the addition (1+2=3, Mic and Line inputs). This source line set means which lines are exposed by the cable's topology filter. Default value is 7 (all three line types). Don't change this parameter unless really needed.
  Under Windows 6.x, you need to restart System Audio Engine or even reboot the system after changing this parameter. Driver restart is not enough to propagate it.
- **Connected source lines mask** - a bit mask that represents currently connected source lines. Unlike the "**Source line types mask**" parameter, this bit mask means which exposed lines are reported as "connected" (plugged in) and must be a subset of the line types mask (all bits set in this value must be also set in the "**Source line types mask**" value). Default value is 2 (only the Line Input is connected).
- **Clock correction** - cable clock correction ratio multiplied by 1000000000 (10E9). Default value is 1000000000 that represents 1.0 ratio. **Note**: **VAC** versions prior to 4.11 use a different multiplier format that is not accepted by later versions so a default value is loaded if an old-format multiplier is found.
- **Stream data buffer ms** (0..500, 0 - auto, not present - default).

- **Stream buffer watermark control enabled** (0 or not present - disabled, nonzero - enabled).
- **Stream buffer watermark low ms** (5..100).
- **Stream buffer watermark high ms** (10..500).
- **Stream buffer watermark max wait for high ms** (20..1000).
- **Restrict client thread affinity** (0 or not present - disabled, nonzero - enabled).
- **Restrict KMixer thread affinity** (0 or not present - disabled, nonzero - enabled).
- **Enable channel mixing** (0 - disabled, nonzero or not present - enabled).
- **Use standard PortCls engine** (0 or not present - disabled, nonzero - enabled).
- **Enable speaker pin type** (0 or not present - disabled, nonzero - enabled).

Almost all of these parameters can be controlled by VAC Control Panel and are explained there. If you create/modify some parameters manually, you need to restart **VAC** driver to propagate them.

When parameter value is changed by VAC Control Panel, its value is automatically copied to the **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EuMu** key to allow driver boot-time access. If you create/change a parameter under "Software" branch manually, you should ensure its creation/changing under "System" branch. Otherwise, the driver will be incorrectly configured on a next boot because there is no access to the "Software" branch at a boot time.

However you could propagate parameter change to "System" branch by hand, it's better to simply restart the driver that propagates parameter values automatically.

Saving and restoring VAC configuration

You can save **VAC** configuration by simply exporting the "Software" registry subtree (main key with all subkeys) mentioned above, using interactive **RegEdit** utility or the following console command:

reg export "HKEY_LOCAL_MACHINE\SOFTWARE\EuMus

Design\Virtual Audio Cable\4"
"%HOMEPATH%\documents\vaccfg.reg"

This command exports **VAC** configuration subtree to the "vaccfg.reg" file located in your "My Documents" folder.

To restore **VAC** configuration, just delete both "Software" and "System" settings subtrees, then import a previously saved .reg file, using **RegEdit** or the following console commands executed under an administrative account:

reg delete "HKEY_LOCAL_MACHINE\SOFTWARE\EuMus
Design\Virtual Audio Cable\4"

reg delete
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EuM

reg import "%HOMEPATH%\documents\vaccfg.reg"

After that, open VAC Control Panel and restart the driver to propagate boot-time parameters.

You can create as many configurations as you want, keeping them in different files.

Please be careful with registry manipulations, any mistake would cause serious system mailfunction.

Please don't confuse **VAC** settings shown in VAC Control Panel with Audio Repeater application settings that can be specified in a command line.

System default (preferred) device issues

Why system default device may be arbitrarily changed

Please note that your system default audio device can be changed in result of **VAC** installation if you had not previously assigned this device explicitly. If a default device was assigned explicitly, it remains unchanged. If it wasn't, Windows may use any of present devices, and there is no easy way to predict which device will be used. **VAC** installer itself never changes default device settings; it may be done by Windows if default device wasn't set explicitly.

To set a default device, open Audio Properties applet. Then find a property page (tab) that displays **Playback** and **Recording** settings and select particular devices from the lists.

Note you must **always** select each device from a list and click **OK** or **Apply**, even the selection already shows this device. Windows creates default device registry record only if the device is set **explicitly**. If a device record is not present, Windows picks a device by its own rules and displays it in **Audio Properties** page. If you will install/uninstall another audio device, Windows may pick some other device in the future.

Using a Virtual Cable as a system default device

If playing and/or recording application does not allow to specify a recording/playback device directly, try to set **Virtual Cable N** as a system default device (Wave Mapper) using Audio Properties Applet.

Keep in mind that a system default playback device is used by Windows to play system event sounds when you open/close a menu, run/exit an application, when message boxes appear on the screen. If you set a system default playback device to the **Virtual Cable**, these sounds are directed to this cable. You will not hear them but they will be passed via the **Virtual Cable** to its recording clients. If you don't want these sounds to be passed via the **Virtual Cable**, disable them in the **Sounds** tab by setting a sound scheme to "**No sounds**". You will be prompted to save a current scheme to re-enable it in the future.

Some poorly designed applications always use a first available device and cannot be configured at all. In Windows 2000, to ensure that **Virtual Cable 1** is a first device, you should disable all other audio devices. Don't remove audio cards or uninstall their drivers, just disable their functions in the Device Manager. This may allow you to achieve your goal but you would not hear any sound until you re-enable hardware audio.

In Windows XP and later, a device assigned as a system default always becomes a first-enumerated in MME, DirectSound and WASAPI interfaces.

Using different default devices for different applications

In Windows 5.x, default audio device is set at per-user basis so you can assign different **Virtual Cable** devices as default devices for different user accounts. For all applications started under a certain account, default
recording/playback devices will be mapped to appropriate **Virtual Cables**.

In Windows 6.x, default devices are set at per-system basis so you cannot use this technique.

To create additional user accounts at your computer, log in to the Administrator account, open the "User accounts" applet in Windows Control Panel and create one or more user accounts. If you usually work under a limited (non-administrative) account, create limited accounts too, otherwise create administrative accounts because not all applications, especially old, can work properly under a limited account. Once each account is created, set a password for it.

Another way to manage user accounts is using **Administrative Tools** - **Computer Management** item in Windows Control Panel. User accounts can be controlled in **User Accounts and Groups** - **Users** item. Creating a new account, specify a password, uncheck "User must change password at next logon" and check "Password never expires".

Once additional accounts are created, log in under them using **Fast User Switching** feature or logoff/logon sequence, and set different **Virtual**

Cable devices as playback and/or recording devices, as needed.

To start an application under a given account, use "**Run as...**" item from a shortcut context menu. Open the **Start Menu**, locate application shortcut as usual, but use right-click instead of left-click. A context menu will appear. Left-click at "**Run as...**" item, select "The following user" and enter account name and password. If both were entered correctly, application will be started under given account and use different preferences than it uses under your usual working account.

To start applications automatically from a command (batch) file, use the **runas** command. See **Windows Help and Support** for details.

To avoid unneeded complexity, use additional accounts only for applications that don't allow to select any available audio device and use only a default one.

Please note that each user account has its own preferences: desktop themes, sound settings, application working directories, temporary files etc. For example, if an application uses a working directory, creating project files in it, such directory may be different under different user accounts. So you may need to specify these preferences manually if the application works under an account different from your usual working account.

Audio layering issues

WDM/KS driver is a lowest-level layer in Windows audio subsystem. Most applications don't connect directly to it. Instead, they use higher-level DirectSound, MME and WASAPI interfaces provided by the System Audio Engine. Using such intermediate layers, an application cannot have a full control over all audio parameters. So there are some special issues and limitations. At a glance, they might look quite complex but they are easy to understand in practice.

Driver's pin allocation

WDM/KS drivers, like **DirectShow** filters, are represented by several numbers of pins. If an application or the system needs to record or play audio data, it creates an instance of the specified pin. A pin instance is associated with a data stream, and instance owner (the application or System Audio Engine) becomes to be a client of audio driver.

Applications that use WDM/KS or DirectSound/WASAPI in exclusive mode, request a separate pin instance. If there is an available pin instance, such request succeeds, otherwise it fails. Most modern drivers provide only a single pin instance. Therefore, several WDM/KS applications can record/play simultaneously only if there are more than one available pin instances at a particular device driver.

To allow simultaneous playback/recording possibility, System Audio Engine uses its own pin instance when a device is accessed via DirectSound, MME or WASAPI interface in shared mode. When an application connects to a device in shared mode, its connection request is implicitly processed by System Audio Engine. If the engine is not connected to the appropriate pin, a pin instance is created. Otherwise, no additional pin instances are requested from the device driver.

If an exclusive-mode access is requested by an application through System Audio Engine but no shared-mode connections for the appropriate pin exist (no common pin instance allocated to System Audio Engine), the engine creates its common pin instance first to allow further shared-mode connections (there is a known bug here). If the driver is

single-client and supports only a single pin instance, there will be no available instances to satisfy the exclusive-mode access request. In such case, the request is either rejected or satisfied in shared-mode, depending of the parameters.

Therefore, if there are not enough available pin instances, Windows prefers shared-mode access for compatibility purposes. But direct WDM/KS access requests don't go through System Audio Engine so they can allocate the last (or even single) possible pin instance. If such direct low-level request is made before a first higher-level request, System Audio Engine will be unable to create its common pin instance and all further higher-level requests, regardless of shared or exclusive mode, will fail.

Vice versa, if the last (or single) possible pin instance has been allocated for System Audio Engine, further direct WDM/KS access requests will fail.

Format selection for System Audio Engine

Creating a pin instance for System Audio Engine in shared mode, Windows audio subsystem must choose an audio format for it. Format selection is based upon application's format specified in its connection request, set of formats supported the device driver and some other rules. There are following common rules of System Audio Engine audio format selection in Windows 5.x:

- **MME** playback: sampling rate is equal to requested, bits per sample is preferably 16 and number of channels is preferably 2 (stereo).
- **DirectSound** playback: sampling rate is equal to a first matching from driver's set, bits per sample is preferably 16 and number of channels is preferably 2 (stereo).
- **MME** recording and **DirectSound** capture: format is the same as requested except some incompatible cases (weird sampling rates, number of channels is greater than 8 etc.).

If possible pin format parameters are strongly restricted and System

Audio Engine cannot use 16 bits and 2 channels, other bitness and/or number of channels can be used.

In Windows 6.x, a separate default format can be specified for shared mode connections.

Stream and client counting

All shared-mode connections to a recording or playback device are established through System Audio Engine using a single common pin instance. So if your applications use shared-mode access, you will see only a single playback and a single recording stream in VAC Control Panel because all of these applications are not **VAC** clients. They are clients of System Audio Engine. **VAC** even knows nothing about their existence and streaming activity; all their requests are serverd by System Audio Engine.

A pin instance created for System Audio Engine is often "sticky" and persists several seconds after all applications have closed their connections to a **Virtual Cable**. You can see it in the VAC Control Panel.

Only exclusive-mode connection allows the application to become **VAC** client, create its own pin instance and an associated data stream. For each exclusive-mode connection, an additional stream is counted by the recording/playback stream counter.

You can experiment with various applications and formats and watch number of clients/streams in the VAC Control Panel.

Windows audio subsystem and some applications may access **VAC** driver and its pins for property requests, creating no pin instance. Such connections are counted in the Driver Clients field of VAC Control Panel.

How cable format is selected

For each **Virtual Cable**, **VAC** maintains the cable format. The cable format is a common format for all cable clients (data streams). All render (output) data are converted to the cable format, mixed and then distributed to all capture (input) clients, being converted to their particular

formats.

VAC selects the cable format when a **Virtual Cable** becomes active (gets its first client stream). As described above, there may be implicit, hidden clients like System Audio Engine whose might request a connection with a format different from the format specified by the application.

For example, default playback format for **Virtual Cable 1** is set to 48000/16/2. The application requests an exclusive-mode connection through System Audio Engine, specifying 96000/24/2. Prior to satisfy this request, System Audio Engine requests a pin instance to be used for shared-mode access, specifying the default device format (48000/16/2) which is fixed by **VAC** as the cable format. Then the engine requests another pin instance for the application, specifying 96000/24/2. But the cable format remains to be 48000/16/2 and **VAC** will downsample 96000 to 48000 and cut 24-bit samples to 16-bit ones. Therefore, system default device format assigned to a **Virtual Cable** for shared-mode access can affect exclusive-mode access to this **Virtual Cable**.

Similarly, improper cable format range may cause quality problems in Windows 5.x. For example, is the range is set to 22050-96000/8-32/1-8, System Audio Engine creates its own common pin instance with 96000/32/8 and cable format is set to the same. If most applications use formats like 44100/16/2, 48000/16/6 or like them, there will be many unnecessary format conversions.

To prevent such issues, choose cable format range to be possibly close to your favorite format set. See format limiting rules for details.

Other issues

Windows 6.x systems have a common bug that prevents System Audio Engine from creating its common capture (recording) pin instance for shared-mode access if there are some active (allocated) pin instances (streams). For example, it occurs if you have started recording using KS interface (KS version of Audio Repeater, ASIO4ALL or something like) and then try to start recording using a higher-level interface (MME, DS, WASAPI) in shared mode. As a workaround, start the higher-level interface first then start KS recording..

DirectSound issues in Windows 5.x

Target device testing

Before using a target device, an application or DirectSound subsystem itself often performs some testing, repeatedly opening the device with various audio formats to see which of them are supported. Such testing is usually performed at Windows boot, at DirectSound application startup and at DirectSound device selection.

Most devices can be successfully opened in any time with any audio format they support. But a **Virtual Cable** can restrict possible stream format range and reject open requests that specify "bad" formats. Due to that, some DirectSound applications may fail to select this cable or may unable to start at all. You might need to stop all cable activity to allow such applications to initialize properly. Or you might even need to set less restrictive format range and/or stream format limiting mode before starting this application and restore appropriate values again when application finishes its initialization.

Using hardware acceleration

In Windows 5.x, DirectSound offers a **hardware acceleration** feature that allows to utilize some internal audio device capabilities. One of these capabilities allow a multistream playback and recording. It means that a device can process multiple audio streams at a same time and its driver can create multiple pin instances in parallel. Multistream feature allows several applications to use a device simultaneously without a special sharing support in software. Each application creates one or more streams and a device processes all of them appropriately. For example, if an internal hardware DSP can resample and mix several streams, it reduces main CPU load and allow applications to be more flexible in audio format selection.

Although **VAC** has no proprietary hardware and uses only main CPU resources, it reports a hardware acceleration support to allow Windows audio subsystem to create multiple pin instances (streams). When an

application creates a hardware-accelerated DirectSound buffer, Windows audio subsystem requests a new pin instance from the driver. At a driver side, a new audio stream is created (number of active streams is displayed by the VAC Control Panel). Having several audio streams, **VAC** converts their formats to the cable format and then mixes together.

Each cable can create as many pin instances as specified by the maximum instances configuration parameter. If there are no available instances, DirectSound fails to create a hardware-accelerated buffer. For an ordinary buffer, DirectSound can create a new stream in System Audio Engine layer. In such case, System Audio Engine will convert and mix these streams instead of the **VAC**. So you can control how application's audio data will be processed.

Sometimes you can control application's ability to use hardware acceleration features. For example, WinAMP and Foobar2000 players have checkboxes for it.

Hardware acceleration level

Windows 5.x DirectSound subsystem uses each hardware acceleration feature only if a current maximum hardware acceleration level allows it. There are four acceleration levels: **None**, **Basic**, **Standard** and **Full**. Each next level allows DirectSound subsystem to utilize more of device/driver internal capabilities and process audio data more efficiently. So higher acceleration levels are better but they may cause some incompatibilities so Windows provides lower levels to achieve better compatibility.

A maximum allowed hardware acceleration level can be set to any of these four values for each DirectSound device independently. Applications using DirectSound interface can utilize no more acceleration features than the maximal device level allows.

A multistream feature is available only on a highest (**Full)** acceleration level. For each new device, maximal level is set to the **Standard** value. Therefore, to utilize all device capabilities, you must **manually** set it to the **Full** value.

You can change maximal acceleration level for **VAC** devices using Audio Properties applet. Open the **Audio** tab, select any **Virtual Cable N** device in **Sound Playback** listbox and click **Advanced** button. On the **Advanced Audio Properties** page, open **Performance** tab, select desired acceleration level and then click **OK**. On the **Sounds and Audio Properties** page, click **Cancel** (if you will click **OK** here, your default device will be changed).

Windows 6.x and higher implements no DirectSound hardware acceleration. All audio processing is performed by main CPU.

Working if channel mixing disabled

If channel mixing is disabled, hardware-accelerated playback may produce undesirable effects. For example, if you play a mono stream to a two-channel cable, DirectSound first creates a shared stereo stream with channel mask 3 (**FL**+**FR**), then creates a separate hardware-accelerated mono stream with channel mask 4 (**FC**). Since these masks don't overlap, **VAC** does not perform scatter/gather conversion and copies mono channel directly to cable's left channel.

Control Panel application

VAC Control Panel is a standalone application that allows to view cable operating modes and to configure their parameters. All configurable **VAC** parameters can be set with the **VAC Control Panel**.

Don't confuse **VAC Control Panel** with the Windows Control Panel, they are independent at all.

You can start **VAC Control Panel** from the **Start menu**, by opening the program folder where you have installed **VAC**.

VAC Control Panel shows all available **Virtual Cables** and their parameters/statuses, together with global driver parameters and status.

Driver parameter section

This section combines global driver parameters not specific to a particular cable.

- **Cables** - number of cables created by the driver. Has its own **Set** button. Changing this value causes driver restart.

- The **Worker threads** group controls driver worker thread usage policy. Has its own **Set** button. Number of cables cannot be changed while at least one cable is in use.

  Up to - maximum number of worker threads running by **VAC** driver. In **Auto** mode, one working thread per physical CPU/core is created (a single thread on a single-core CPU, two threads on a dual-core CPU or a two-CPU board, and so on). You may need to limit number of worker threads if you create many (several dozens of) streams and want to lower CPU load but don't need to minimize latency as possible. In such case, you can limit number of worker threads by n-1 or n-2 (**n** is a number of physical CPUs/cores) to free 1-2 CPUs/cores for other tasks. In hyper-threading systems, you can set number of worker threads larger than a number of physical CPUs/cores, up to a number of logical CPUs. Some systems can

work more efficiently in such case but please make sure that your system really can. Otherwise, you may lower the performance instead of raising it.

Prio - worker threads scheduling priority. In **Auto** mode, they have a priority between a normal thread priority in the system (15) and a maximum possible priority (31). The higher is the priority, the more aggressive **VAC** uses CPU resources, preempting other user and system threads carrying out other tasks. Use priority value 31 with an extreme care: if the interrupt interval is very low (1-2 ms), number of worker threads is not limited and several threads intensively communicate with **VAC** driver using short (several milliseconds) data buffers, it may produce very high load on all available CPUs/cores and even freeze your system totally.

- **Clients** - global driver client counter. Counts all driver clients, not only audio stream owners. Each time as the driver interface is accessed (opened) by the system and/or an application, a new client arrival is registered. Some applications and/or system components may hold driver interface instances without stream creation; you could see client counter change as they start or exit.

- **Streams** - global audio stream counter. Counts all audio streams associated with all cables.

- **Restart** - driver restart button. Restarts the driver, causing the driver to terminate, unload, reload and reinitialize. See this for details.

- **Reset counters** - reset overflow/underflow counters for all cables.

- **Restart Audio Engine** - a button to restart System Audio Engine.

Cable parameter section

This section combines individual cable parameters.

- **Format range (SR, BPS, NC)** - sampling rate, bits per sample and number of channels ranges for the cable format range.

- **Connected source lines** - currently connected source lines.

Checked source lines are exposed lines are reported as "connected" (plugged in). Connected lines must be a subset of the "**Source line types mask**" configuration parameter.

- **Max inst** - maximum number of pin instances (number of times the pin can be instantiated). A first pin instance is always reserved for System Audio Engine. Other instances are available for client applications. A driver that provides more than one pin instance is considered "hardware-accelerated" and can be used in extended DirectSound or Exclusive Mode WASAPI playback. See the DirectSound issues chapter for details. Changing this value causes driver restart.

- **Ms per int** - a number of milliseconds per software timer events (interrupts). 20 ms per interrupt means 50 interrupts per second, 10 ms per interrupt means 100 interrupts per second, and so on. It represents a frequency with which audio data portions are transferred from playback cable side to its recording side (many applications use the "latency time" term). The more events per second, the less latency time, the more system overhead. Typical audio cards use 5-15 milliseconds per interrupt; DirectSound subsystem uses 10 ms as a default.

- **Stream fmt limit** - stream format limiting mode.

- **Volume control** - state of the cable volume control mode. Enable to allow to change cable and stream volume levels.

- **Clock corr ratio** - clock correction ratio value, in percents. If the value is 100%, cable clock rate exactly matches a sampling rate value of the cable format. If the correction value is more than 100%, cable clock goes faster, otherwise it goes slower. For example, if the value is 100.25, cable clock goes 0.25% faster; if the value is 99.98, cable clock goes 0.02% slower. Clock correction precision is 0.0000001%.

  You can change clock correction value even if a cable is active (has streams), changes will be reflected immediately.

- **Enable channel mixing** - controls channel mixing mode. Leave enabled to use standard channel conversion, disable to use channel scatter/gather mode.

- **Use PortCls** - controls PortCls engine usage. Enable to use a standard and reliable but non-optimal stream data interchange engine.

- **Enable spk pin** - set "Speakers" type for playback end filter pin. Changing this value causes driver restart. Please note that all cables with this option enabled will have **the same playback endpoint name**.

- **Stream buffer** - controls stream internal data buffer length. "**Auto**" means a minimal possible value chosen by a driver.

- **Stream buffer watermark control** - groups stream buffer watermark control parameters. See here for details.

  Enabled - controls watermark control mode. If enabled, buffer watermark values are used. Otherwise, buffer processing is performed as usual.

  Low - low watermark value in milliseconds. When the amount of buffered stream data (measured in duration of recording/playback) becomes less than a specified value, **VAC** delays stream data processing until the client provides enough amount of data/storage.

  High - high watermark value in milliseconds. When stream buffer processing is temporarily delayed because of reaching low watermark limit, the client should provide at least the specified amount of data/storage to resume data processing.

  Max wait - maximum high watermark waiting time in milliseconds. This is a guard time to avoid infinite delays in stream data processing. If stream data processing is delayed but the client does not provide data/storage within the specified time, buffer processing is resumed anyway.

- **Set** - a button to apply specified parameter values to all selected

cables. Most cable parameters cannot be changed if the cable is in use (has clients).

- **Reset counters** - a button to reset overflow/underflow counters for selected cables.

Cable list section

This section shows all available cables and their most important parameters/states.

- **Cable** - cable number, starting from 1. These numbers are the same as recording/playback device numbers like "Virtual Cable N" or "Line N (Virtual Audio Cable)".
- **Ms per int** - number of milliseconds between timer events (interrupts).
- **SR/BPS/NC ranges** - format range.
- **Stream fmt limit** - stream format limiting mode.
- **Volume control** - state of the cable volume control mode.
- **Chan mix** - channel mixing mode.
- **PortCls** - PortCls engine usage mode.
- **Wmk control** - stream buffer watermark control mode.
- **Current format** - current cable format, represented by a single sting: **Type/SR/BPS/NC(CM)**, where:

  Type - format type: PCM - old PCM (WAVEFORMATEX), ExtPCM - modern extensible PCM (WAVEFORMATEXTENSIBLE).

  SR - current sampling rate.

  BPS - current number of bits per sample. If sample container size is greater than required to store actual number of bits per sample (for

example, 4 bytes for 22 bits), container size in bytes is shown in parentheses after number of bits.

NC - current number of channels.

CM - channel configuration mask (hexadecimal).

For example:

- 48000/16/2(3) - 48000 samples per second, 16 bits per sample (in two-byte container), two channels (standard stereo);
- 96000/24(4)/6(3f) - 96000 samples per second, 24 bits per sample (in four-byte container), six channels (old speaker configuration with two back speakers);
- 96000/24/6(60f) - 96000 samples per second, 24 bits per sample (in three-byte container), six channels (new speaker configuration with two side speakers).

- **Rc stms, Pb stms** - number of recording (capture) and playback (render) streams.
- **Signal** - peak signal level (amplitude) indicators. Are shown in a standard channel sequence (FL-FR-FC-LF-BL-BR-FLC-FRC-BC-SL-SR). To have a larger view, use Windows **Magnifier** Tool (Accessories - Ease of Access).
- **Oflows, Uflows** - number of data overflows/underflows.

Cable list is updated twice per second, showing actual cable states.

Please note that even if cables are visible in **VAC Control Panel**, they may be inaccessible for audio applications in remote sessions.

If current cable format parameters (sampling rate, bits per sample and number of channel) are shown for a particular cable, it means the cable is in use (has clients). Also, number of playback and recording streams (pin instances) are shown.

The number of playback/recording streams means how many input/output pin instances are created. Usually it is equal to a number of input/output clients but first rendering pin instance is internally used by System Audio Engine (see audio layering issues section). Additionally, an single application can create multiple pin instances.

Since Windows audio subsystem reserves a first pin instance for System Audio Engine, you may see some **Virtual Cable** is still in use within several seconds after playback is stopped. It is because audio subsystem keeps a mixing thread running several seconds after all other pin instances are closed.

Current cable format can help if you don't know which audio format is used by a particular application or are not sure is cable port really open or not.

The **OFlows** ans **UFlows** columns show cable data overflow/underflow counters. An overflow is registered if a recording client does not provide free data storage (a buffer) within an appropriate time, and some audio data to be recorded are lost (there is a drop-out). An underflow is registered if a playback client does not provide filled data storage (a buffer) within an appropriate time, and some audio data to be played back are late (there is a gap).

Overflows/underflows are registered for a cable client but counted for entire cable. If a cable has several clients while some of them behave well and provide their buffers fast enough but some others fail to provide buffers fast enough, recording client overflows affect only their own data but playback client underflows affect entire cable and data of all recording clients. If a playback client experience underflows, the cable will not receive a solid data stream from this client and resulting mixed cable stream will be incomplete. On the contrary, recording clients receive a mixed cable stream so if a client experience overflows, only its own data will be incomplete.

If overflows/underflows occur only at stream start/stop, you can ignore them. It happens, for example, if an application first starts a rendering/playback stream then begins to provide data buffers and/or first stops to provide data buffers then stops a stream. Such behavior is not

desirable but it is not an error.

If the cable is accessed in shared mode, there may be some issues with overflow/underflow counters. System Audio Engine does not pause its shared stream when all client streams are paused. Instead, it simply stops providing memory buffers to the driver (**VAC**) if there are no running client streams.

For example, if you run an audio player, start playback to a high-level audio port (endpoint) provided by System Audio Engine from **VAC**'s low-level port (pin) while there are no other playback streams in the system, System Audio Engine becomes **VAC** driver client, creating a playback stream, and begins providing audio data buffers. If you pause the player, System Audio Engine does not pause its pin instance provided by **VAC**, it just stop providing data buffers. Since **VAC** does not know why its client does not provide data buffers, it registers buffer overflows and increments the counter continuously. If such situation occurs with audio recording, underflow counter is continuously incremented.

Please don't confuse cable's overflows/underflows with Audio Repeater's overflows/underflows.

Changing driver/cable parameters

To change parameter values for a single cable, select particular cable row in the list, change values in selection/edit fields, and click **Set** button. Make sure that the selected cable is not used (the format parameter in the list is blank).

To change parameter values for multiple cables, hold **Ctrl** key while selecting individual rows or hold **Shift** key to select several adjacent rows. In multi-selection mode, a field filled with a value or a cleared/checked box mean that the parameter value is equal for all selected cables. An empty field or a greyed box mean that this parameter values are different for some cables. If you leave (or make) some fields empty and/or some boxes greyed before clicking **Set**, these parameters will not be affected.

To change number of cables, select appropriate value in the **Number of**

cables field and click **Set** button next to this field. Before changing number of cables, it is **strongly recommended** to close all audio applications. Also, many audio applications query a list of available devices only on startup and they would not work properly until restarted.

Please note that changing number of cables require driver restart.

System limitations and overhead

Windows 2000 limits number of MME devices to 8 while Windows XP/2003 increase this limit to 32. So MME applications that use waveIn/waveOut functions won't see more than 32 cables in Windows XP/2003 and more than 8 in Windows 2000. To use more cables you need either to use DirectSound or WDM/KS interfaces or switch to VAC 3 because it is a legacy MME driver.

Windows 6.x don't limit number of MME/DirectSound endpoints.

Don't create more cables than really needed. Having many audio devices in your system, especially in Vista/Server 2008/Win 7, you may experience excessive system overhead due to very slow endpoint processing. For example, creation of 16-20 cables may keep Windows **100% loaded during a minute** after each driver startup (system boot process, installation or restart from **Control Panel**) before all audio devices become accessible. Creating 100 cables in Vista/Server 2008/Win 7 may cause up to **15-20 minutes of full load** before Windows finishes endpoint processing and all audio devices become accessible.

Cable list update in the **Control Panel** is slow when there are many cables. Audio application startup also may take a long time due to device initialization.

Moreover, creating too many (100 and more) cable devices can cause other audio devices to **disappear**. Use this feature with care.

Also don't set **Milliseconds per interrupt** to a small (1..3 ms) value unless really needed. Very small values significantly increase system time interrupt frequency and overhead. On a slow machine, you may experience performance degradation instead of raising it.

Don't enable cable volume control features unless really needed. If you don't need a volume control, keep it disabled to improve sound quality and minimize CPU load.

Restarting the driver

After changing number of cables and/or maximum instance count, **VAC** driver must be restarted to apply changes. The **Control Panel** tries to automatically restart the driver. You also can manually restart the driver using the **Restart** button (for example, to prolong a silent period in trial version).

To restart the driver, **VAC Control Panel** needs to be started with administrative privileges.

Additionally, driver restart is possible only if there are no active streams (no cables are currently used by client applications). So you need at least to stop all recording/playback using **Virtual Cable** devices. If number of cables is changed, it is **strongly recommended** to close all audio applications before restarting the driver.

If you have configured a **Virtual Cable** device as a default playback device and system event sounds are enabled, it may prevent **VAC** driver from being restarted dynamically. To be able to restart the driver dynamically, either disable system sounds or change default playback device to a hardware audio adapter.

If **Control Panel** cannot automatically restart the **VAC** driver, number of cables and/or maximum instances count will be changed after Windows restart.

Restarting System Audio Engine

In some rare cases, when System Audio Engine seems to work incorrectly (stops accessing some endpoint, reports invalid parameters etc.), you can restart it without rebooting Windows. To do that, stop all audio streams and close all audio applications. Then click **Restart Audio Engine** button.

Audio Repeater Application

Audio Repeater is a simple application that transfers a real-time audio stream from any **Wave In** MME recording port to any **Wave Out** playback port. **Audio Repeater** can be used with **VAC** to transport audio data from a real sound card to a **Virtual Cable**, monitor a signal going through a **Virtual Cable** with a real sound card, transport audio data from between two **Virtual Cables** etc.

Audio Repeater uses MME interface to access audio devices. It means that playback to **Virtual Cables** goes via System Audio Engine layer and **Audio Repeater** cannot fully control actual audio format used by System Audio Engine pin instance creation (see audio layering issues). To access via WDM/KS interface, use Kernel Streaming version.

To supply a **Virtual Cable** with a real audio signal, choose a sound adapter (card) port in the **Wave In** field of the **Audio Repeater**, and choose **Virtual Cable N** in the **Wave Out** field. To monitor an audio signal transferred through the **Virtual Cable**, choose **Virtual Cable N** in the **Wave In** field and the sound adapter (card) port in the **Wave Out** field.

In some simple cases, you can use the Listen feature (available since Win7) instead of **Audio Repeater**.

You can leave "None" in the **Wave Out** field. In such case, **Audio Repeater** will only record from the device specified in **Wave In** field, without playing the data back. It can be useful for visual signal level monitoring if you don't need to pass the signal to another device.

In MME, device name length is limited to 31 characters so you will not always see full endpoint names.

Before starting a transfer with the **Audio Repeater**, select format parameters possibly close to formats used by other applications connected to this **Virtual Cable**. To eliminate redundant format conversions, pay attention to the audio layering issues.

The **Channel config** parameter specifies a typical channel configuration

(mono, stereo, surround etc.) or a custom mode that allows you to select any channel combination.

Transfer is started as you click the **Start** button. **Audio Repeater** opens both audio ports and maintains a circular buffer movement between them. When a buffer filled with data is received from the **In** recording port, it is immediately passed to the **Out** playback port, and vice versa.

The **Queue** gauges show how much data buffers are held in device queues. If a gauge goes to the right, it means that data amount if the queue is increased, and vice versa. In the most stable state, both queues are half-filled.

Both queues start near to a medium state to minimize a clock difference effect. But the longer is transfer time, the larger is the difference, and the higher is stream interruption probability. Side activity, such as disk read/write, application launch, high CPU load can affect stream stability.

Gauge indicators under channel configuration checkboxes show peak signal levels in appropriate channels.

Overflow and underflow counters

There are two counters, **Overflows** and **Underflows**. **Overflows** counter is incremented when **Audio Repeater** has no free buffers for recorded data (input queue gauge comes to the left). **Underflows** counter is incremented when **Audio Repeater** has no buffers to play back (output queue gauge comes to the left).

Overflows and underflows can occur as a result of a clock difference effect, heavy system load (when Audio Repeater does not receive enough CPU time) etc.

Using **Virtual Cable** devices in **Audio Repeater**, please note that these application's overflows and underflows are opposite to cable's overflows and underflows. For example, when **Audio Repeater** has no free buffer to pass to cable's recording end, it registers an underflow (there are no data). On the contrary, the cable has data but cannot give them because there are no buffers so the cable registers an overflow (there is no room

for data).

Queue exhaust and overflows/underflows may occur due to different timing event periods. Each device has its own time period to notify the driver and/or client application about streaming progress. For example, if the device issues a notification event each 15 milliseconds and you have specified total buffering time of 40 milliseconds in 8 parts (5 ms each), every notification will indicate that 3-4 parts are recorded/played. It is a significant portion of the entire buffer so the streaming will be rough and unstable. You will need to increase buffering time and/or number of buffer parts to smoother the stream. Unfortunately, only **Virtual Cables** allow to specify a timing event period; other devices don't expose this information and you should determine it by experiment.

Clock rate indicators

Clock fields near overflow/underflow counters show clock rates for each port in relation to another port. If clocks of both ports have the same rate, "100%" is shown. If port clock is faster than other port clock, the value is over 100%, otherwise it is below 100%.

Clock rate values may help to specify the clock difference value for a **Virtual Cable**.

Please note that clock rates are calculated roughly because data amount is counted in portions (buffers).

Choosing proper buffering parameters

To optimize audio transfer, you can vary the buffering time (**total buffer**) and **number of buffers** (a number of individual parts, or data blocks). The **total buffering time** (in milliseconds) specifies how much audio data reserve a device driver will have. The larger is buffering time, the more robust is the transfer, the less is chance of sound interruption due to task switching, disk operations etc. But the larger is the buffering time, the larger is latency value, and the larger is a delay between incoming and outgoing audio data. Typical buffering times are from 200-300 to 1000-2000 ms.

The **number of buffers** defines how many parts of the total buffering memory will be used. The more buffers are used, the smoother is the transfer even the total buffering time is small. But large number of buffers raises system overhead and use of more than 15..20 buffers doesl not make data transfer smoother.

To avoid sound interruptions, each audio buffer part should contain at least 3..7 ms of audio. Therefore, having 100 ms total buffering time, you can divide it into 8-12 parts but for 20 ms, it is not recommended to use more than 4 parts.

The **priority** field controls priority of the **Audio Repeater** process. The higher is priority value, the smoother a process is executing, the less time is rest for another processes. Most processes have **Normal** priority value, **Audio Repeater** has **Normal** value by default.

By default, **Realtime** priority is available only for users with administrative privileges. Be careful setting this priority value because it can cause Windows to freeze. If you want to use this priority under a limited user account, enable the "Increase scheduling priority" policy for this account (or for a group), using local security policy editor from an administrative account. The editor can be accessed from the "Administrative tools" folder in Windows Control Panel or by typing "secpol.msc" in the Start-Run dialog.

Please note that clock difference effect **can** be lowered but **cannot** be eliminated at all. Therefore don't rely on **Audio Repeater** as a robust audio transfer tool. A **Virtual Cable** can do a robust, smooth and long-time transfer between two applications but there is no way to establish a robust transfer between two Windows audio devices. You can try to minimize a clock difference effect by using **VAC** clock correction feature.

Format descriptor selection

When opening a wave device, **Audio Repeater** uses the **WAVEFORMATEX** format descriptor if sample size if 16 bits or less and number or channels is 1 or 2. If sample size is greater than 16 bits and/or number of channels is greater than two, **Audio Repeater** opens the device with a modern **WAVEFORMATEXTENSIBLE** descriptor.

Command line options

Audio Repeater accepts several command-line options, allowing to pre-configure it:

/Input:<str>	Input (capture, recording) device name
/Output:<str>	Output (playback, rendering) device name
/SamplingRate:<num>	Sampling rate (samples per second)
/BitsPerSample:<num>	Bits per sample
/Channels:<num>	Number of channels
/ChanCfg:<str>	Channel configuration
/BufferMs:<num>	Total buffering length in milliseconds
/Buffers:<num>	Number of buffers (parts of buffering space)
/Priority:<str>	Process priority
/WindowName:<str>	Name of application instance window
/AutoStart	Start audio transfer automatically
/CloseInstance:<str>	Close a specified Audio Repeater instance by its window name

Option names are case-insensitive. **<Num>** means a number, **<str>** means a string. Specify option values **exactly** as you see them in **Audio Repeater** window, even they are cut. Device names must be specified **exactly** as they are shown in Audio Repeater **Input** and **Output** menus. Don't specify names shown in Device Manager or in Sound/Multimedia applet.

If a string contains spaces or other special characters, enclose it in double quotemarks ("). To avoid name matching problems, leading/trailing spaces are stripped, space groups between words are replaced with a single space. To insert a quotemark, specify two consecutive quotemarks in the string.

If you want to enter a command line from a console and a string contains national characters that don't exist in a console code page, create a

command (batch) file, placing the command line into it, and insert **CHCP** command before the command line. See **Windows Help and Support** for **CHCP** command description. Save this file in UTF-8 encoding to allow the command interpreter (cmd.exe) to recognize it.

With **/Input** and **/Output** options, you can specify device numbers as **#n**, where **n** is a device number in **Audio Repeater** device menu, starting from zero.

The **custom** channel configuration option has the following format:

/ChanCfg:custom=<num>

where <num> is a hexadecimal channel mask as defined for the **KSAUDIO_CHANNEL_CONFIG**.

For example, to configure **Audio Repeater** with "Virtual Cable 1" as input device, "USB Speakers" as output device, sampling rate 48000, 16 bits per sample, two channels (stereo), command line parameters should be the following (a single, continuous line):

/Input:"Virtual Cable 1" /Output:"USB Speakers" /SamplingRate:48000 /BitsPerSample:16 /Channels:2

/CloseInstance option has a special meaning. Instead of specifying a parameter for a newly started **Audio Repeater** instance, it instructs to close an existing instance that has a specified window name (specified earlier by **/WindowName** option). If used, it must be the only option in command line. If a named window exists (even if hidden), the window close message is sent to it. The option can be used to close any application window, not only **Audio Repeater**'s.

Automation

You can create a shortcut for **Audio Repeater** or invoke it from a batch (command) file to achieve better automation. See **Windows Help and Support** center for details (type "create shortcut" in the Search field to see how to create and edit shortcuts, type "batch files" to see how to use batch files).

To create a shortcut, locate either **Audio Repeater** executable file in **VAC** installation directory or default **Audio Repeater** shortcut in **VAC** folder in the **Start Menu**. Command-line parameters should be appended (space separated) to the executable file path in the "Target" field.

In a batch file, use **Start** command (type "**start /?**" in the Windows Command Prompt window for explanation) to start several instances simultaneously.

To run **Audio Repeater** instances automatically after a logon, place shortcuts to the **Startup** folder of the **Start Menu**. See **Windows Help and Support** for details.

Minimizing to tray icon

Audio Repeater supports tray minimizing. To restore, left-click the icon. To open a context menu, right-click the icon.

To start **Audio Repeater** minimized to the tray, use either the **start** command with **/min** option (see "**start /?**" for details) or a shortcut with the "**Minimized**" running mode.

Kernel Streaming version

This **Audio Repeater** version is intended to access audio devices via through high-performance and low-latency Kernel Streaming (WDM/KS). It looks and works like standard MME version but has some important differences.

Using KS version, please note it accesses device pins directly, not using System Audio Engine or other intermediate layers. It means no format conversion is performed by System Audio Engine, only device and/or its driver format conversion features will work, if any.

Additionally, KS version allocates a pin instance directly, so if device driver does not allow multiple pin instances, this pin cannot be used simultaneously by several applications and/or Windows itself. Since you have opened such device in KS version, you will be unable to open another instance, even via other Windows audio interfaces.

See audio layering issues for details.

Therefore, don't use KS version if you don't clearly understand how it works and how it differs from the basic MME version.

Queue indicators in Kernel Streaming version behave slightly different than in MME version due to intermediate buffering. The input queue is fully filled at the start but the output queue is only half-filled to minimize clock difference effect. In a stable state, input queue indicator should be near its right side and output queue indicator should be near its center.

Please note that KS device names are usually different from MME names. If you use KS version in a command file, copy device name from KS version, not from MME one.

Additionally to the **/ChanCfg** option, KS version supports separate channel configurations for both input and output (**/ChanCfgIn** and **/ChanCfgOut** with the same syntax). Mostly, it is intended to be used with cable channel scatter/gather feature but can be used for other channel mapping purposes. See the example of scatter/gather feature usage.

Audio drivers support only a limited set of formats (format ranges) at WDM/KS level. If a given format is not supported by device pins, **Audio Repeater** displays an error message ("No appropriate pin found"). In such case, try to use another format (for example, 48000 samples per second instead of 44100 or vice versa). If you don't know what formats are supported by the driver, use ASIO2KS that can show supported format ranges.

Please note that Kernel Streaming version interacts directly with an audio device driver and uses a different communication technique than standard Windows audio subsystem. Most audio drivers, even WHQL certified, are tested only using standard Windows audio subsystem requests. On non-standard requests, such drivers may generate errors and even BSODs. For example, some Realtek WHQL-certified drivers generate BSODs if a sampling rate less than 8000 is selected for a capture operation.

Such driver behavior is definitely incorrect and should not be considered

as **Audio Repeater**'s problem because a well-built driver never causes BSODs. If a hardware driver causes a BSOD, please contact driver's vendor instead of complaining to **Audio Repeater**.

Examples

Recording an audio signal produced by an application

- Launch a **playback** (producing a sound) application.
- Find audio **output** settings in playback application.
- Select a particular **Virtual Cable N** device as the **output** (playback/render) device.
- Choose a particular audio format for a playback.
- Launch a **recording** (consuming a sound) application.
- Find audio **input** settings in recording application.
- Select the **Virtual Cable N** device as the **input** (recording/capture) device, making sure the **N** (a cable number) is the same for both applications.
- If possible, choose the same audio format as chosen for playback.
- Start recording. You will see a silence is recorded (with full **VAC** version) or there will be a periodical voice reminder (with trial version).
- Start playback. You will see an audio position is moved and level meters are alive but there will be no audible sound because a signal is passed via **Virtual Cable** to a recording application directly.
- Stop both recording and playback.
- If your recording application can play recorded data (for example, any audio editor like Audacity, Wavosaur, Sound Forge, or Cool

Edit/Audition), you can immediately hear a recorded signal. Just make sure that its wave output is assigned to a real audio adapter (a card) playback port and headphones/speakers are connected.

Monitoring a signal during recording

- Configure playback and recording applications as described above.
- Launch Audio Repeater.
- Select a proper **Virtual Cable N** as an input device.
- Select a real audio adapter playback port as an output device.
- Choose the appropriate audio format parameters.
- Click **Start** button. You will see data queues moving.
- Start recording and playback as described above.

A signal produced by the playing application will be distributed among your recording application and **Audio Repeater** which passes it to a real device and you will hear it. You should hear clear sound produced by the playback application (with full **VAC** version) or the sound with periodical voice reminder (with trial version).

You can also leave "None" in the **Wave Out** field of the **Audio Repeater** to only watch signal levels on the screen, not playing a signal back to the speakers. To only watch signal level, you can use Control Panel application instead.

Starting from Win7, there is a built-in Listen feature to monitor a cable. You don't need **Audio Repeater** in a simple case. But in more complex cases, **Audio Repeater** could be useful (for example, to connect a **Virtual Cable** to several devices at the same time).

Determining an actual audio format that application uses

- Launch VAC Control Panel.
- Launch an application and select a **Virtual Cable N** as a playback or recording device.
- Start playback or recording. In VAC Control Panel, you will see the current cable format parameters. If the cable format is not limited, it probably reflects a format used by this application.

But due to the layering issues, if the application uses shared-mode device access, **Virtual Cable** device can receive a request from System Audio Engine, not from the application directly. Please also note that applications selecting a format automatically can change it from time to time.

Chaining several applications one by other

- Create an appropriate number of **Virtual Cables** using VAC Control Panel.
- Launch your applications.
- In the first application, which will be audio signal creator/producer, select a **Virtual Cable N** as a playback/output device.
- In each intermediate application, which will be both audio signal consumer and producer, select a **Virtual Cable X** as a recording/input device and a **Virtual Cable Y** as a playback/output device. The **X** for each recording device must be the same as the **Y** used for a playback device in previous application in the chain. In other words, each intermediate application must be connected to the next by a separate cable. You don't need to number them in ascending order, they simply must be separated.

- If last application is not a final recording destination, select a real audio adapter as its playback device.

- To temporarily exclude an intermediate application from the chain without changing other applications' settings, simply insert Audio Repeater instead.

Recording **Skype** conversations

To record Skype conversations, you need two **Virtual Cables** and at least two Audio Repeater instances:

- In **Skype**, you must assign recording/input to **Virtual Cable 1** and assign playback/output to **Virtual Cable 2**.

- First **Audio Repeater** instance must transfer from your audio adapter's recording port to **Virtual Cable 1**, supplying first **Virtual Cable** with a microphone signal.

- Second **Audio Repeater** instance must transfer from **Virtual Cable 2** to your audio adapter's playback port, passing **Skype** output to the speakers.

- **Audio Repeater** uses one-second buffer by default that introduces a significant (about 1..2 seconds) delay. You need to experiment with total buffering time, lowering it until audible interrupts occur. The more buffering time, the stable the audio transfer, the longer delay, and vice versa.

- Now you can separately record your voice from the **Virtual Cable 1** and other party's voice from the **Virtual Cable 2**. Almost any multitrack recorder can be used - for example, N-Track Studio or Adobe Audition (formerly Cool Edit Pro).

- If you want to record both parties from a single source (mixed), you need to start third **Audio Repeater** instance and transfer from the

Virtual Cable 1 to the **Virtual Cable 2**. Then you could record a mixed signal from **Virtual Cable 2**.

- To avoid an echo, make sure your microphone source line monitoring channel is muted in audio card's mixer.

Distributing audio signal among several audio devices

- Configure your playback application to use **Virtual Cable N** device as an output.
- Launch Audio Repeater instance, select **Virtual Cable N** device for input and first audio device for output. Change format parameters if needed and start audio transfer.
- Launch another Audio Repeater instance, select the same **Virtual Cable N** device for input and another audio device for output. Change format parameters if needed and start audio transfer.
- Start playback in your application. You will hear a sound in speakers/headphones connected to both audio devices.
- Please note that due to clock difference issue you will not get a stable sound within a long period of time.

Please note that due to clock difference issue you will not get a stable sound within a long period of time.

Adding an audio signal to an audio broadcast

If you have an audio broadcasting application that broadcasts a signal from your microphone and want to add a sound from another application (audio player, synthesizer, signal processor etc.), use the following configuration:

- In the broadcasting application, choose **Virtual Cable N** as the input/recording device instead of the microphone.

- Launch Audio Repeater, choose a microphone device as input and **Virtual Cable N** as output. If your microphone is mono, adjust the channel config. Click **Start** - Audio Repeater will transfer a signal from the microphone to **Virtual Cable N**. Say some words in the microphone - signal level indicators must react to them.

- Start the broadcasting and check microphone signal latency. It will be greater than broadcasting directly from the microphone. If needed, try to adjust buffering settings to lower the latency.

- Launch an application that should produce the additional sound and choose **Virtual Cable N** as output device. Activate audio playback - a signal will be mixed together with a microphone signal.

The same method can be used to mix an audio signal into a Skype conversation.

Extracting selected channels from a stream or inserting them to a stream

To **extract (gather)** some channels from a stream:

- Choose a **Virtual Cable** device and configure its format range to accept the source multichannel stream. Make sure that the **Enable channel mixing** checkbox is cleared (channel mixing is disabled).

- Route a multi-channel audio stream to the chosen **Virtual Cable** device.

- Run Kernel Streaming version of Audio Repeater, choose the appropriate **Virtual Cable** as input device and leave output device field to be "None". Choose input channel configuration matching the source audio stream, click **Start** and check if all appropriate channel level indicators are active. If no, check your setup for common multi-

channel issues. Finally, click **Stop**.

- Choose **Custom** for input channel configuration and select channel positions that you want to extract.

- Choose an output device (possibly another **Virtual Cable**) and appropriate channel configuration. For example, if you want to extract two channels, choose **Stereo**. If you want to extract to a non-standard channel set (for example, three channels), choose **Custom** and select appropriate number of adjacent channel positions, starting from **FL**. Then adjust the common number of channels if needed.

- Configure another **Virtual Cable** (if used) and recording/processing application that will consume extracted channels.

- Start recording/processing then start audio transfer in **Audio Repeater**. Signal level indicators corresponding to the **source** channels selected to extract should move.

To **insert (scatter)** some channels to a stream:

- Choose a **Virtual Cable** device and configure its format range to accept the destination multichannel stream. Make sure that the **Enable channel mixing** checkbox is cleared (channel mixing is disabled).

- Run Kernel Streaming version of Audio Repeater, choose a source device (possibly another **Virtual Cable**) as input. Choose input channel configuration (standard or custom) matching the source audio stream. For example, if you want to insert two channels, choose **Stereo**. If the source stream has a non-standard channel set (for example, three channels), choose **Custom** and select appropriate number of adjacent channel positions, starting from **FL**.

- Select the appropriate **Virtual Cable** as output device, choose **Custom** for output channel configuration and select channel

positions where you want to insert (mix) the source channels. Then adjust the common number of channels if needed.

- Run a recording/processing application and configure it to record a multi-channel audio stream from the chosen **Virtual Cable** device.

- If needed, run another application that plays a multi-channel stream to the chosen **Virtual Cable** as a base sound. In such case, selected channels will be mixed into this sound stream.

- Start recording/processing then start signal transfer in **Audio Repeater** then start the source stream and another application (if used). Signal level indicators corresponding to **source** stream channels should move. Recording/processing application will receive a multi-channel stream with source channels inserted at the specified positions.

- If there are problems with multi-channel playback/recording, check your setup for common multi-channel issues.

Uninstallation

Before starting **VAC** uninstallation, make sure **VAC** driver is **not used** by any application. Cable device usage is shown by its Control Panel application. If you attempt to uninstall the driver while it is in use, the uninstaller will fail to properly remove the driver from the system and to delete its files and folders. So it is recommended to **close all audio applications (including those hidden in the tray)** before uninstalling the **VAC**.

Since uninstallation requires privileged actions, you must explicitly log to the Administrator account if UAC is not enabled in your system.

Before you start uninstallation, open the Audio Properties applet. If a **Virtual Cable** device is selected as a default device for the recording and/or playback, select any other devices instead. Then close the properties page.

To start removing **VAC** software from the system, open the **Virtual Audio Cable** folder in the **Start Menu** and select the **Uninstall** item. If UAC is enabled in your system, you will be prompted to elevate your privileges to run the uninstaller.

If you have accidentally deleted your **VAC** Start Menu folder, you can start the uninstaller manually. Click **Start** - **Run** to open the **Run Application** dialog, click **Browse**, locate a directory where **VAC** is installed (by default, it installs to "C:\Program Files\Virtual Audio Cable"), and double-click **setup** item; a path to the uninstaller application will be inserted in the command line. Then append a space character and "**-u**" option to the command line (outside of the quotes, if any). Resulting command line should look like this:

"C:\Program Files\Virtual Audio Cable\setup.exe" -u

Then click **OK**, the uninstaller should start.

After successful uninstallation, you will receive a confirmation message.

If some driver files, services and other objects are currently in use by the

system or by , they cannot be deleted immediately. In such case, the uninstaller creates a "delayed delete requests" for them and prompts you to restart Windows. If you don't want to restart immediately, you **must not try** to reinstall **VAC** in this Windows session otherwise severe errors may occur.

Uninstallation troubleshooting

If the uninstaller fails to properly remove software components from the system, try advanced setup mode that allows you to over-install the software. In some cases, after successful over-installation you will be able to successfully uninstall the software. As a last chance, you can try to remove the driver manually:

1. Log in with the Administrator account.
2. Open Device Manager.
3. Expand the "Sound..." device category by clicking on the "+" sign.
4. Right-click the **Virtual Audio Cable** device and choose **Uninstall**.
5. In Windows 6.x, check the **Delete the driver software** item.

This sequence deletes only **VAC** driver and its devices but driver service may not be automatically deleted. To manually delete the kernel-mode service, unzip **VAC** distribution package into any empty folder. Using **My Computer** or **Windows Explorer** navigation, locate this folder and select the **RemoveService** file. Right-click this file, and select **Install** in the context menu. Service will be deleted immediately if it is stopped, or marked for deletion and deleted at next system startup.

After removing system references, you can delete driver file **vrtaucbl.sys** located in the **system32\drivers** subdirectory of the Windows installation directory.

VAC directory and its **Start Menu** folder can be deleted using any Windows file management utility (i.e., Explorer).

Please report all problems not covered by this manual, to the author.

Distribution policy

VAC is shareware.

Trial version of the product is free of charge and can be distributed freely if its package and file contents remain unchanged and there are no commercial purposes. You cannot sell it, solely or within other products.

Full version is commercial and is granted to registered users after payment. This version can be used only in conditions specified in the license agreement. See the **license.txt** file in the distribution package.

Audio Repeater application is free of charge and can be used separately, without installing the entire product.

Support and feedback

VAC is developed by **Eugene Muzychenko**, an independent software developer living in Novosibirsk, Russia.

Bug fixes and updates are available at the VAC homepage.

You can also visit Other Eugene Muzychenko products page.

If you encountered an unexpected behavior and/or a technical problem, firstly please make sure that you have read the documentation and understand basic principles and usage rules of the product. Looking for troubleshooting information, please see troubleshooting section and FAQ/How to... lists at first.

If the problem is not covered by this manual, please email to the author.

Writing a message, add "Virtual Audio Cable" or "VAC" to the subject field. Messages with empty subject or only with a common word/phrase like "Hi", "Help request" or "Some questions" may be considered as SPAM and automatically killed by the filters.

Only English and Russian languages are supported in communication.

Please include following items in your email:

- Application name (**VAC**), its version and type (trial or full).
- Platform type (single/multiple CPU) and CPU type.
- Operating system (type, language, build and patch level or service pack number).
- **Exact** situation description:
  - What applications did you use (names and versions)
  - How their audio settings are configured
  - What **Virtual Cables** did you use (cable numbers, recording or payback operations).
  - How these cables are configured (what VAC Control Panel shows for these cables in "Cable parameters" box).
  - How **VAC** driver is configured (what VAC Control Panel shows in "Driver parameters" box).

- VAC Control Panel window image may be useful.
- What default shared recording/playback formats are set in the system, if any.
- What actions did you perform
- What did you expect
- What was really happen
- Exact error message texts (or pictures)
- Other details, if any

For example, if you have a problem with audio connections, you should describe how you connect applications to audio devices (real and virtual), audio formats used, recording/playback modes (if any), and specify any other information that could help to understand your problem. The more meaningful information you provide, the faster and helpful response you will get.

If the problem is related to installation/uninstallation, please attach a log file created by **Setup** application.

Identifying yourself

Please identify yourself properly. If you are a registered (paid) user, please specify your order number or email address used for the order placement. If you use a different email address and don't specify your full name or your name is very common, it could be difficult or even impossible to find your order.

Explaining a problem

Please write problem explanation to be rich in content. Avoid meaningless messages like "I cannot install it" or "It did not work". Your message must be informative and allow to understand your problem. Please write as exact as possible: what did you want to achieve, what did you do, and what was happened. If you have received an error message and code, please include them into your report. For long messages, attach some screenshots instead.

Attaching files

If your problem can be described better by some data files, you can attach them to your message. For example, if you experience a strange noise in audio data, it could be useful to present a data fragment saved to a WAV/MP3 file. But don't send large (a minute or more) audio fragments. Send meaningful fragments only. In most cases, 5-10 seconds of audio is quite enough.

When possible, please avoid large (greater than 1-2 Mb) email attachments except explicitly asked.

Attaching logs

During installation and uninstallation, a log file is created, describing a process details. After successful installation, the log file is kept in the **VAC** installation directory (**install.log** file). If installation/uninstallation process fails, a log file is kept in the temporary directory; in such case, the installer informs you how this file is named and offers you to view its contents.

If your problem is related to installation/uninstallation process, please include log file contents into your report. You may either copy/paste file contents to the email text or attach a log file. It may help to investigate your problem.

You may be asked for Windows SetupAPI log (device installation process log). In Windows 2k/XP/2k3, log file is named **setupapi.log** and located in Windows root folder, usually named **Windows**. In Vista/Server 2008/Win7, log file is named **setupapi.dev.log** and located in the "**inf**" subfolder.

By default, only most important messages are included in SetupAPI log. To provide maximum information, it is better to enable all possible messages to be written to the log. To enable them, create a **.reg** file with the following text:

```
REGEDIT4
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVe
"LogLevel"=dword:4800ffff
"LogMask"=dword:fffffff
```

Save the file then double-click it and allow the changes to be added to the registry. **Changes will take effect after Windows restart**.

To restore default log settings, use **.reg** file with the following text:

```
REGEDIT4
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVe
"LogLevel"=-
"LogMask"=-
```

Attaching screenshots

To illustrate your problem, you can attach some screenshots to your report. If there is only a single informative window (a dialog box, an error message), take a "window screenshot" (**Alt**+**PrintScreen** keys), not full screenshot (single **PrintScreen** key). If there are several informative windows at the screen, arrange them to be all visible and take a full screenshot (single **PrintScreen** key).

Attaching screenshots, please pay attention to an image size. Don't attach uncompressed BMP images because they are very big and redundant, and don't wrap images with Word or PDF documents. Convert BMP images to a compressed image format (JPG, GIF and so on) using standard Windows **Paint** utility or a free image viewer/editor (IrfanView, XnView etc.). You can also reduce color depth, converting 24-bit color to 8-bit but don't resize an image because text information may become unreadable.

Attaching videos

If your problem is hard to be explained in text and static images, you may capture a screen video with a screen recording software like free CamStudio. Plase note that video clips may be large; don't attach clip files larger to 5-10 MB to a message. Upload them to a free file hosting service and provide a link in the message.

Attaching minidumps

If your system crashes, configure it to make a crash minidump. A

minidump file contains some essential information about a crash, it may help to analyze your situation.

To configure system to create a minidump, open Windows Control Panel, then open **System** item. Select **Advanced** tab and click **Settings** button in **Startup and Recovery** group. In the **Write Debugging Information** box, choose "**Small memory dump**". Leave a default path (%SystemRoot%\Minidump) in the **Small Dump Directory** field.

Additionally, you may clear **Automatically Restart** checkbox to prevent automatic reboot after a crash. In such case, Windows will stay at the blue screen message.

If you had a system crash, then tried to enable minidump creation and saw it is already configured, it means you already have a minidump for this crash and can send it, not needing to wait for another crash.

After each system crash, a new minidump file is created in the **Windows\Minidump** folder. File names contain a date and a sequence number within a date. Locate 1..3 of last minidump files and attach them to your report. Make sure you are attaching only minidump (64 kb) files, not a big dump (several megabytes in size) files.

Upgrading to new versions

Within a major version (2.xx, 3.xx, 4.xx etc.) you can freely upgrade to any newer minor version. For example, if you bought version x.08, you can freely upgrade to x.10 or x.12.

To upgrade to a newer minor version, please see your purchase order confirmation email message. It contains download and installation instructions for all available minor versions.

If you bought a previous major version (for example, 3.xx), you can upgrade to a newer major version (for example, 4.xx) with a discount. To obtain discount conditions, please visit VAC homepage mentioned here.

How to...

I want to use **VAC** with my applications XXX and YYY. How to configure Virtual Cable for my case?

There are no predefined recipes to use **VAC** together with given applications. **VAC** is a common tool and its usage is simple if you understand its basic concepts. You definitely must understand the "interface", "port", "client" and "format" terms, VAC principles and basic usage rules. This manual describes common rules, and you must apply these rules to your configuration. See the typical usage examples.

Please ask for a support only if you are sure your task is significantly different from most others and you have read the manual carefully but have not found an answer.

I bought **VAC** several months (years) ago. How can I upgrade to a newer version?

Upgrade policy is described here.

How can I record my **Skype** conversations?

To record Skype conversations, you need to create relatively complex routing scheme. It requires clear understanding of Windows audio operation basics, **VAC** principles and learn other examples. In most cases, it is better to look for a special **Skype** recording application.

Here is an example of **Skype** recording.

I have several audio adapters with their own speakers attached to each adapter. How to play a sound using all adapters at the same time?

Just route your sound stream to **Virtual Cable N** and use several Audio Repeater instances to route from the same **Virtual Cable N** to each of your adapters. The stream will be copied to all output devices simultaneously.

The same method can be used to play a stream simultaneously to different destination lines of a single adapter under Windows 6.x (for example, Speakers and SPDIF) if simultaneous playback is supported by the adapter.

But please note you will not hear all these streams as a single, solid sound due to different digital audio paths inside the system. Additionally, sounds coming from different speakers will "slide apart" from each other due to a clock difference effect.

How to transfer a multichannel (5.1, 7.1 or similar) audio stream through a **Virtual Cable**?

Possibly you need to adjust the Speaker Configuration for this **Virtual Cable** first to match the channel distribution of your audio format.

In Windows 6.x, you also need to choose appropriate default audio formats for both playback and recording cable devices.

How to save and restore **Audio Repeater** settings?

Audio Repeater does not support saving/restoring its settings. But it supports command-line options to specify all needed settings on startup. You can create a Windows shortcut and/or a command/batch file or VBScript to invoke **Audio Repeater** with the specified settings. If you need to start **Audio Repeater** automatically after logon, place the shortcut to the "**Programs** - **Startup**" folder of the **Start Menu**.

How to extract some channels from a stream or insert them to a stream?

Use the channel scatter/gather feature. Please see the example.

How to record from several cables at the same time using multi-track recording software?

Recording from several cables is identical to a recording from several hardware devices. You can even mix hardware devices and **Virtual Cables** in your multitrack session. For example, you could use Adobe Audition (formerly Cool Edit Pro) to record **Skype** conversation placing

your voice and your party's voice to separate tracks. Most multitrack recorders have a common, fixed format in each session; you can specify a proper format at new session creation.

I have enabled test-signing mode in my system for unsigned VAC version. Now I got signed version, how to disable test signing mode?

Please open a command-line window (**Start**-**Programs**-**Accessories**-**Command Prompt**, TotalCommander, FAR Manager) and enter the following command:

bcdedit /deletevalue {current} testsigning

To enter the command correctly, don't type it manually, use Windows copy/paste feature. In **TotalCommander** or **FAR Manager**, you can simply paste the command by **Ctrl-C** or **Shift-Ins**. To paste command text into **Command Prompt** window, click on icon in upper-left corner to open a context menu, open **Edit** submenu and select **Paste** to insert the command. Then press **Enter** to execute command. Type **exit** to close command window.

Frequently Asked Questions

Does **VAC** ensure an absolutely reliable and stable streaming?

Definitely, no. There is no way to ensure an absolutely reliable and stable streaming in Windows because Windows is not a real-time operating system. In most cases, a hardware-assisted streaming, when you play and/or record a stream to/from a hardware audio adapter, is quite stable because all audio data processing is performed by a separate hardware device, not by the main CPU, and redundant buffering is used. Since **VAC** represents virtual devices that have no dedicated hardware support, it has to compete with other user threads and system components in main CPU resources usage.

In Windows, there is no guarantee that an application (and even a kernel-mode driver) code will certainly executed within a given period of time. For relatively long periods, like one second, the probability is almost 100%, and even for 100 ms, the probability is very high. So, in general, multi-threading works well. But for short periods (10 or 1 ms), the probability significantly drops. Only enough buffering can prevent arbitrary data loss.

Therefore, some special measures must be taken to achieve more reliable and stable streaming. Please learn about audio layering issues, advanced usage rules, HowTo and Troubleshooting sections.

Does **VAC** use registration keys, Internet activation procedure or other technical licensing measures?

No, it doesn't. Once purchased a license, you will get full version package and need just to install it instead of trial version.

My audio adapter (soundcard) has no "What You Hear" or "Stereo Mix" recording function. Does **VAC** implement it?

Definitely, no. **VAC** does not interact with other audio devices. It cannot automatically intercept any audio signal, it can route a signal only if a **Virtual Cable** device is **explicitly** used in audio path.

If you are using Vista/Win7, there can be a loopback source line in your audio adapter but disabled by default. To check it, right-click on speaker icon in the system tray, select "Recording devices", right-click on any device in the list and select "Show disabled devices". Some additional devices may appear. If there is a device like "Stereo Mix", right-click on it and select "Enable".

You can record sounds from applications and/or system by directing their audio output to some **Virtual Cable** device. It can be achieved by selecting **Virtual Cable** device in application's audio settings or using **Virtual Cable** as a system default device.

Since audio output will go to **Virtual Cable** instead of your audio card, you need to monitor the cable to hear the sound.

My audio adapter (soundcard) has analog and digital outputs but I cannot use them in the same time. Can **VAC** help?

Definitely, no. **VAC** cannot extend limited hardware capabilities. If an audio adapter allows to use its inputs/outputs in the same time and its driver doesn't restrict it, you should be able to use them independently. But some adapters have simplified drivers that allow to use only a single output line. You could look for another driver, maybe it exists. For example, there can be a driver for Windows 5.x suitable for Windows 6.x too.

Can **VAC** route sound between different PC's?

No, **VAC** operates only locally. But you can use **VAC** together with an network audio software like Skype to route audio signals over a network. Please note that audio quality will depend on network software behavior.

I am evaluating trial version and hear the repeating "trial" voice. Does it mean that **VAC** works correctly?

Definitely, no. It only means **VAC** driver is loaded and a signal is routed from a **Virtual Cable** device to the speakers/headphones. To make sure **VAC** works for you, please build a proper application chain that routes a useful signal (with the "trial" voice mixed).

In other words, if trial version works exactly as you need, successfully routing your audio data, and you only want to get rid of the "trial" voice, then you can consider that **VAC** correctly works for you.

I have tried a trial version but was not satisfied. Will full version work better?

Full version features differ from a trial ones only by a maximal number of cables. Full version also does not add a female voice reminder to the signal. See the trial version limitations for details. Sound quality is the same in trial and full version, no quality degradation measures are taken in trial version.

Therefore, trial version has all the features of full version and can do anything that full version can do except number of cables. If three cables are enough for your task but you cannot get the trial working, it means you are doing something wrong or **VAC** is not intended to your task at all.

If you have successfully applied trial version to your configuration but your task simply requires more than three cables, full version will work for you. But please note that using many cables may introduce some problems.

How many cables I need to implement my task?

You need as many cables as many **independent** audio signals you want to pass between applications:

- To pass a single audio signal from one application to another, you need a single cable.
- To mix three signals from three applications and pass a single combined signal to fourth application, you still need a single cable.
- To pass two independent signals, you need two cables.
- To mix two signals into a single combined signal and mix three other signals into a separate combined signal, you still need two cables, and so on.

Please note that each cable is single-directional. Therefore, if you have two applications capable to produce and receive audio signals at the

same time, you need two cables to create two-way communication between these applications.

Why additional steps are needed to increase number of cables limit from 16 to 256?

Technically, it would be simpler to enable all cables by default and avoid additional steps required to do it. But each cable, regardless of its actual presence, needs to be installed as a system subdevice and needs some records in system registry. In some cases, Windows needs to query not only subdevices actually present but all installed but currently unavailable subdevices. If all 256 cables are installed, Windows 6.x may require up to one minute to open audio device applet when you right-click the speaker icon in the taskbar and select "Playback devices" or "Recording devices".

Therefore, usual number of cables is limited to 16 to avoid unneeded delays. In rare cases when more cables are needed, it is easy to replace the .inf file and reinstall **VAC**.

How much is sound latency with **VAC**?

VAC itself adds almost no latency but actual value is much more dependent from an audio interface used by the application and application's audio buffer configuration than from a **Virtual Cable** timing event frequency.

Very good performance and latency can be achieved using exclusive-mode device access (hardware-accelerated DirectSound in Windows 5.x, WASAPI in Windows 6.x). For even better performance and lower latency, you can use WDM/KS, directly or via an ASIO wrapper like ASIO4ALL or ASIO2KS. Shared-mode streaming methods have lowest performance and highest latency.

Typically, latency time is comparable with an audio buffer granularity. For example, if application uses a total buffering time of 500 ms divided into four buffer parts, average application latency time will be 500 / 4 = 125 ms. If a cable is configured to 10 ms per interrupt, cable latency will be no more than 10 ms.

To reduce latency time, first try to decrease application's audio buffer granularity. Some MME applications allow to specify number of audio buffers and a total buffering time; set these values to 50..100 ms per buffer part and 500..1000 ms of total time. DirectSound applications also may allow to specify a minimal buffer fragment size for processing.

You can also try to decrease the number of milliseconds per interrupt for a cable but it may help to decrease latency time by several milliseconds only.

With a well-tuned configuration (hardware, OS and applications), it is possible to reduce latency time down to several milliseconds. Latency less than 1-2 ms is not possible in Windows because a lowest timing interval is 1 ms.

Can **VAC** map default audio devices to different cables for different applications?

VAC does not map audio devices. If you assign some **Virtual Cable** device as a system default recording and/or playback device, mapping is performed by Windows. But in Windows 5.x, you can use different user accounts to map default devices differently for different applications.

Can **VAC** be used on a virtual machine (virtual server, VPS/VDS)?

VAC can be used on any Windows machine, regardless of its type. But there are some important issues since **VAC** is a kernel-mode driver, not a user-mode application. Please read about virtualized environment compatibility issues.

Working remotely, please read about remote access compatibility issues.

What are the difference between old and new 5.1 and 7.1 speaker/channel configurations?

Old 5.1 speaker/channel configuration named "**5.1**" or "**5.1 back**" contains back speakers/channels (back left and back right). New configuration contains side speakers/channels instead and is named "**5.1 surround**".

Old 7.1 speaker/channel configuration named "**7.1**" or "**7.1 wide**" contains "front left of center" and "front right of center" speakers/channels. New configuration contains left and right side speakers/channels instead and is named "**7.1 surround**".

How much are CPU resources consumed by **VAC**?

It depends primarily on number of audio streams processed at the same time, then on CPU type, clock frequency, number of cores, L1/L2 cache sizes, memory type, clock frequency, Windows configuration, audio interface used, application/device interaction algorithm, audio buffers size, number of buffers used, cable event rate, volume control, format conversion options etc.

For example, the following configuration:

- T8300 (2 cores, 2.4 GHz), Q9550 (four cores, 2.8 GHz) or 3610QM (four cores, 2.3 GHz) CPU
- Single Virtual Cable device configured for 5 ms per event, no volume control
- 32 Audio Repeater KS instances (**Virtual Cable 1** to **Virtual Cable 1**, 44100/16/2, 20 ms per buffer, 4 buffers, no format conversion)

that creates 64 independent audio streams processed by **VAC** driver, produces about 1% overall CPU load. The load depends mostly on number of streams. For example, 16 cables and 32 Audio Repeater KS instances (**Virtual Cable N** to **Virtual Cable N+1**) produce the same 1% load.

32 Audio Repeater MME instances (500 ms per buffer, 12 buffers) creating 64 streams processed by System Audio Engine and a single stream processed by **VAC** driver, produce about 5-10% overall CPU load under Windows 7.

With format conversion and/or volume control, each stream processing may consume up to dozen percent of CPU resources. The higher are sampling rate and number of channels, the more CPU resources are taken to process the stream.

Please note that system tracing/debugging features (for example, Driver Verifier) may significantly increase CPU load and decrease stream processing performance and stability.

I see very high CPU load when **Virtual Cables** are used. Is it normal?

It is normal if there are format differences between a current cable format and recording/playback clients' formats. In such case, format conversion will be performed. It takes a high amount of CPU resources.

Check the number of milliseconds per interrupt parameters for all active cables. If this parameter is very low (1..3) for some cables, try to increase it at least to 5..7.

CPU resources are also used to perform a volume control. If you don't need volume control for a particular cable, it is better to disable it.

In Vista/Win7, I see high CPU load produced by **AudioDG** process. Is it normal?

In Windows 6.x, **AudioDG** process hosts almost all audio processing in the system (basic audio interface support, System Audio Engine, local and global audio effects (LFX/GFX) etc.). If you run several audio application that create several audio streams, especially using audio effects, **AudioDG** may produce significant CPU load, regardless of are **Virtual Cable** devices used or not.

Can **VAC** render my MIDI sequence to a Wave file?

Definitely, **no**. **VAC** simulates a simple electric cable, it cannot synthesize a sound from a MIDI sequence. Although, **VAC** can help you to record output of some software synthesizers to a Wave file. You also could use a built-in Windows GS software synthesizer that plays synthesized waveform to a system default playback device.

Troubleshooting

VAC has been installed successfully but you don't know how to run it. There are no new shortcuts on the desktop.

VAC is a "passive" tool. It is not an application that you can run, it is a **virtual device driver**. You can use it with almost **any** audio application by selecting "**Virtual Cable N**" devices in their audio settings. Please read about VAC principles and consult other parts of this manual.

You hear an unwanted female voice in the recorded audio. How can I get rid of it?

You are using a **trial** version, please note the trial version limitations. This voice reminder is normal for the trial. To get rid of this voice, purchase the full version.

You have set output of your audio application to **Virtual Cable N** but you hear no sound.

If you connect audio source (an application) to the playback/render cable side and leave other side unconnected, you will not hear any sound. You need to connect other (input/capture) side of the cable to another application which will receive (record/capture) a sound produced by the source application. See VAC principles, simple usage rules and examples for details.

You have set output of your audio application to the **Virtual Cable N**, then set input of recording application to the **Virtual Cable N**. Sound recording is fine but you hear no sound from the headphones/speakers during recording. What to do to hear a live sound?

You need to monitor cable transfer as mentioned in the simple usage rules. The simplest way to monitor is using Audio Repeater application included in **VAC** package. See the example.

Starting from Win7, you also can monitor a cable using the built-in Listen feature.

You have configured playback and recording applications to be connected through **Virtual Cable** but audio signal is not passed as expected.

- Make sure that playback application really plays audio signal:

  If there is a song position pointer, it should move.

  If there is a playback/rest time, it should go forward/back.

  If there is a signal level indicator, it should move.

- Make sure that playback application plays audio signal to the chosen **Virtual Cable**:

  Application's settings specify this **Virtual Cable** device or this **Virtual Cable** device is assigned to be system default.

  Control Panel application shows audio format and signal level indicator for this cable as you start playback and does not show them as you stop playback.

- Make sure that the proper audio signal is played by playback application:

  Use Audio Repeater application or Windows Listen feature to monitor a signal passed through the cable.

- Make sure that recording application really records from the cable (time counter is running, level indicators are moved, Control Panel application shows npn-empty recording stream count.

After successful **VAC** installation, system sounds disappear.

Maybe you did not explicitly set your system default device before. In such case, Windows may decide to use some **Virtual Cable** device as a default one. Please read here for details.

When a USB audio device is plugged in, some sounds disappeared.

Many USB audio devices are configured to become a system default audio device when plugged in. So, if you have a device that is set as a system default and plug a USB device, the new device becomes to be a new system default, replacing a previous default device. All sounds that were routed to a previously chosen device are now routed to a newly connected device. A device that was previously set as a default was not lost, it remains accessible by a name, as all other devices.

To avoid this problem, configure audio applications to use explicitly chosen audio devices, not system default ones. Use system default devices only if the application does not allow to select a device from a list.

During **VAC** installation, Windows cannot find some files.

First, try to determine do these files belong to **VAC** file set or to Windows file set. For example, browse **VAC** distribution package contents. If required file names are present in the package, check if they are present in a folder where you have unpacked them before installation. Maybe you have unpacked the ZIP archive without pathnames (nested folders) or missed some files.

If the archive had beed unpacked properly but Windows installer cannot access some files, it might be due to restricted access rights. Check file properties for the access rights. If a folder where files are unpacked has special characters in the full path (punctuation characters, national alphabet characters), try to unpack installation files to a folder that does contain only Latin characters in the path.

If Windows requires some files that are not present in **VAC** package, they might be system files that Windows needs to install and/or check their versions. Usually, all the original files needed are located at distribution packages (Windows installation disk, Service Pack disks). To minimize external package requirement, Windows keeps some frequently used files in its installation folder (**Driver Cache**, **system32\dllcache** subfolders) and use them instead of asking for a disk. But if you delete these packages (**driver.cab**, **sp2.cab** or others), Windows cannot find an original file on the hard disk and asks for a CD/DVD.

If Windows asks for a system or a service pack CD/DVD but you have no these disks, try to locate required files in Windows installation folder. For example, **ksuser.dll** file could be located in **system32** subfolder and **portcls.sys** could be located in **system32\drivers**.

During installation, you get a driver selection error.

In some cases, **VAC** installer aborts with the following message: "Cannot install device (15/515 - There is no driver selected for the device information set or element.)". Most probably, this error occurs if your system is misconfigured.

Please locate the "setupapi.dev.log" file in your system and search it for "access is denied" or "error 5" messages. If you find such errors not only for **VAC** installation attempts, it means that your system is misconfigured (administrator account does not have enough rights and/or system folders have invalid Access Control Lists). You need system repair and even re-installation to restore proper file access rights.

If access deny errors are listed only for **VAC** installations, please send a report containing setupapi.dev.log file to analyze it.

VAC was installed but there are no **Virtual Cable** devices and/or VAC Control Panel reports that driver is not loaded.

Please open Device Manager, expand the "**Multimedia**" or "**Audio**" device subtree and look for "**Virtual Audio Cable**" device. If there is no such device, it means that **VAC** is not properly installed. Make sure that you have carefully followed all installation instructions.

If "**Virtual Audio Cable**" device is present, open its properties and read device status in the "General" tab. There may be some common problems:

- Error code 22 (This device is disabled) - try to enable device from a context menu.
- Error code 39 (Windows cannot load the device driver for this

hardware. The driver may be corrupted or missing) - **VAC** driver file was corrupted/deleted for an unknown reason. Try to uninstall **VAC** and install it back.

- Error code 48 (The software for this device has been blocked from starting because it is known to have problems with Windows) - please read about virtualized environment compatibility issues.

If the device is working properly (no error codes in Device Manager) but VAC Control Panel reports that driver is not loaded, make sure that you run same version of **Control Panel** application executable (from the same distribution package).

If the problem is not described here, please contact support describing the problem in details.

VAC installation produces error 1072 (The specified service has been marked for deletion).

Most likely, you previously tried to uninstall **VAC** when some cables were in use and uninstaller was unable to remove the service. Please reboot and retry installation. If there will be some "already exists" errors, use Advanced Setup Mode to repair.

Trying to start recording from **Virtual Cable** device, you get a message saying it is busy.

At first, please make sure that error message really applies to **Virtual Cable** device, not to another audio device.

This error occurs if there are no available pin instances to allocate for recording operation. By default, each cable pin allows several instances but if all of them are exhausted, there will be an error message on any Windows platform. For details, please read about audio stack layering.

Another cause may be the known bug in Windows 6.x systems.

Trying to start transfer in **KS Audio Repeater**, you get "No appropriate pin found" error.

This error means that Audio Repeater was unable to find a pin that supports requested audio format for the specified device.

Kernel Streaming version of Audio Repeater connects directly to audio device driver, not to System Audio Engine that has format conversion features. In most cases, device driver accepts only formats supported by the device and performs no format conversion. For example, if the driver accepts only 24-bit formats, 16-bit formats can be used only via high-level interfaces - MME/DS/WASAPI, and only 24-bit formats are allowed to be used via WDM/KS interface.

Trying to change **VAC** parameters or perform a restart in **VAC Control Panel**, you get "Access denied" error.

To perform some privileged operations (for example, driver restart via device property change, or restart a service), VAC Control Panel needs to have administrative rights. So you need either to run **VAC Control Panel** "as administrator", right-clicking its shortcut, or log in using the **native** Administrator account (not a user with administrative privileges available).

Trying to restart the driver in **VAC Control Panel**, you get "Unable to restart driver" error.

Some applications and/or services may hold references to **VAC** devices without stream creation. Total count of these references is shown by **VAC** Control Panel in the "Clients" fiels. Normally, there are 4-5 references from system processes, they don't prevent driver from restarting. But if other services/applications hold additional references, driver cannot be restarted until these references are released.

If you have closed all audio applications but client count is greater than 4-6 and driver cannot be restarted, you could try to search processes holding references to **VAC** devices. Download Process Explorer (GUI) or Handle (command line) utilities from Sysinternals and search for paths containing "\topo" and "\wave" substrings. **VAC** device interface paths

end with "\topoN" and "\waveN" where N stands for cable number. Please note that some references are held by system services (svchost.exe processes) so there is no way to determine a service holding the reference.

If you found some non-Windows processes and/or services holding **VAC** device references, try to determine software products that they belong to (by name, by Internet search etc.). Maybe such product has a tray icon that allows to stop the service. As a last chance, try to kill the process in **Task Manager**.

Never kill svchost.exe processes except you are familiar with Windows services.

Using **Audio Repeater MME**, you hear gaps in the audio stream if smooth window minimizing/restoring occurs.

MME version of Audio Repeater uses a traditional window messaging (WM_xxxx) technique to receive real-time event notifications. When Windows performs window minimize/maximize animation, a delay occurs in window message queue processing and Audio Repeater does not receive its buffer recording/playback events in the proper time.

To disable window minimize/restore animation:

- Windows 7/Vista: Control Panel - All Control Panel Items - Performance Information and Tools - Adjust Visual Effects - Animate windows when minimizing and maximixing.
- Windows XP: Control Panel - System - Advanced - Animate windows when minimizing and maximixing.

In Win 6.x, you also can try to enable **Aero** interface that performs some graphical operations by hardware so there are more CPU resources available for audio transfer.

KS version is free of such problem so you can use it instead of MME version if not don't want to disable window animation.

Video-intensive applications cause sound distortions and/or

interruptions.

Applications that draw images on the screen intensively (animations, games, presentations) often consume many CPU and bus resources. Many video drivers violate Microsoft load balancing requirements to draw faster and smoother. Since **VAC** has no own hardware and uses main CPU and bus resources too, there is a competition and video drivers might hold their critical sections for a long time, preventing **VAC** worker threads to maintain reliable data transfer.

To solve these problems, try the following:

- If the problem is related to Windows interface visual effects like window animation, try to disable visual effects at all or enable **Aero** interface.
- If the problem is related to a particular application, try to modify application's settings (for example, lower the frame rate in a game, simulation speed in a simulator etc.).
- Try to increase worker thread priority.
- Try to update video drivers to the latest.

The **Configure** button in playback audio device list of Audio Device Control applet is not accessible.

It is known Windows 6.x behavior. These Windows versions show **Configure** button only for speaker-type pins but don't allow to specify a name for the endpoint, always using "**Speakers**" name. Using speaker-type pins in **VAC** would not allow to distinguish between different **Virtual Cables**, they all would be equally named "**Speakers**".

So **VAC** uses LineOut-type output pins instead of speaker-type ones by default. These pins only allow to select a default format for shared-mode access.

You can enable speaker type using the "**Enable spk node**" parameter in cable parameters section of VAC Control Panel. In such case, please **carefully read the precautions**.

You want to record a multichannel audio stream but there are only two

channels.

At first, start VAC Control Panel and make sure that cable format range includes desired number of channels. For example, if you need 5.1 stream (6-channel), maximum number of channels should be set to 6 or more. In general, number of channels is set to 1..N but in some cases you could need to use a more strict rule (N..N) to disable undesired formats.

If you use Audio Repeater, make sure that an appropriate multichannel format is selected.

In Windows 6.x, shared-mode recording format is set to stereo by default. To use multichannel formats on recording, you need to configure default device format for the cable.

Trying to record from a free device in Windows 6.x, you are informed that the device is in use.

Windows 6.x use endpoints to access different source (input) lines. Most modern audio devices have multiplexed source lines, not mixed ones, so only a single source line can be used at the same time. If a recording endpoint in a multiplexed source line group is currently in use, all other endpoints in the group become unavailable.

An endpoint can be currently used by an application or by the system (as a default device, including default communication device). Audio device usage can be non-obvious, for example:

- The Listen feature is turned on for the device.
- **Recording Device** tab is open in the Audio Device Properties applet (it opens audio devices to show current signal levels).

To make all recording endpoints available for explicit selection by recording applications, make sure that no endpoint from a multiplexed group is currently in use.

VAC Control Panel does not start, displaying "0/1 - Incorrect function" error message.

First make sure that **VAC Control Panel** executable file is started from the **VAC** installation folder and **VAC** was correctly installed before. If you start it from a folder with unpacked **VAC** distribution files, there may be incompatible driver and application versions.

To make sure that driver/application versions are the same, you can compare product version for both driver file (**vrtaucbl.sys**) located in **System32\Drivers** subfolder of **Windows** root folder, and application file (**vcctlpan.exe**) located in **VAC** installation folder (usually **C:\Program Files\Virtual Audio Cable**). To see a product version, right-click on the file, select **Properties**, open **Version** tab and select **Product Version**. Product version for driver and application files must be the same except of a build number (last of four numbers delimited with the period).

Also please make sure that **VAC Control Panel** is started from the folder you specified in last **VAC** installation. If installation and/or uninstallation procedures were performed inaccurately, there may be alternate folders containing different versions of **VAC Control Panel** application, and your Desktop and/or **Start Menu** may contain shortcuts to them.

If driver/application versions are the same, check if **Creative Live! Cam Virtual** driver is installed. It can be checked in **Device Manager** (by name) or in **System32\drivers** subfolder of **Windows** root folder (by the **livecamv.sys** file name). This driver was developed from **VAC** by a third-party company and incorrectly uses some identifiers used by **VAC**, making a collision.

Creative Live! Cam Virtual driver is a part of some **Creative Live! Cam** web camera products. To avoid a collision, disable or even uninstall it.

Please visit Creative website to obtain updated camera software.

At each boot, you see "the data is invalid" message window with "TBIA" header.

This message is generated by **M-Audio Fast Track** product. Most likely, this application queries audio drivers for their properties and handles returned values incorrectly. Please upgrade your **Fast Track** version and contact **M-Audio** for support if the message still appears.

Audio signal passed through a **Virtual Cable** becomes distorted.

Signal distortion (garbling, crackling, popping, static-like clicking etc.) usually occur due to buffering problems in audio applications, System Audio Engine or **VAC** driver itself.

If **Virtual Cable** is the only path between two applications (one is playing back and another is recording), try the following:

- If your application uses DirectSound under Windows 5.x, check and adjust the DirectSound hardware acceleration level.
- Check the CPU load and try to lower it if high.
- Try to stop unnecessary application activity.
- Try to increase length of stream data buffer.
- Try to increase the buffering time in audio application.
- Try to decrease the number of milliseconds per interrupt parameter for the cable.
- Try to increase worker thread priority.
- Try to use stream buffer watermarks.

If there are other (hardware or software) devices in the path (for example, an application records from a device and plays back to **Virtual Cable**, or vice versa), please also check if this issue is not caused by clock difference.

If you are using trial version that adds a female voice reminder, please pay attention to how this voice reminder sounds. Because it is added by **VAC** driver, playback side problems cannot affect it. Therefore, if you play back a signal to a **Virtual Cable** device and hear voice reminder clear but the signal becomes distorted, it means that recording side is OK but playback side has buffering problems. If both voice reminder and useful signal are distorted, it means that buffering problems occur on recording side and/or inside **VAC** driver.

Additionally, look to overflow/underflow counters for the particular **Virtual Cable** in **VAC** Control Panel. If overflow counter is not increasing, or is increasing rarely, but underflow counter is increasing rapidly, it means that buffering problems occur on the playback side, and vice versa. If both underflow and overflow counters are increasing rapidly, it means a

total buffering problem due to CPU overload, high disk/network load, low cable timing event period, system timer problems etc.

As you start audio transfer over a **Virtual Cable**, there are distortions but as you restart transfer, all works fine.

Many system and driver code and data parts are designed as pageable. It means that they are loaded to physical memory on demand as really needed so they may not be loaded automatically on system startup. Additionally, these parts may be unloaded from memory if not used frequently. To load such code/data parts, the system must perform some paging operations, reading from (and probably writing to) some disk files. Disk operations, especially on slow laptop HDDs, are much slower than memory operations and can affect real-time performance.

Therefore, when you start audio transfer, some code/data parts needed to process audio signals may not be loaded from their disk files or be unloaded to a paging file so the system must bring them back to physical memory before they can be used for signal processing. Additional time may be required for first-time initialization or re-initialization. It may cause delays and short-time distortions. Because these delays are random, streaming synchronization may be affected, especially if there are several real-time streams. As all code/data parts used for real-time processing are stabilized, streaming should become stable and smooth.

In Audio Repeater or similar applications used for device-to-device transfer, such delays may cause buffering failures (input or output queues become overflowed or underflowed immediately). In most cases, stopping and immediate restarting restores a smooth streaming.

Overflow and/or underflow counters in **VAC** Control Panel are growing continuously.

Massive buffer overflows/underflows usually occur in two situations:

- **VAC** driver client (an application or System Audio Engine) fails to provide memory/data buffers in time (fast enough). In such case, you need to check your system's performance (CPU and memory speeds, CPU consumption, background disk/network activity etc.).

- **Virtual Cable** pin is used by System Audio Engine in shared mode and all System Audio Engine's clien streams are paused. It is a normal sutuation caused by System Audio Engine's behavior because **VAC** cannot distinguish between buffer/data absence due to a temporary client failure or such pause processing technique.

You use **Audio Repeater** to monitor **VAC** cables with a soundcard and there are periodic interrupts of the sound.

Like a hardware audio adapter, **VAC** has internal software clock to time audio data transfer. There always is a slight difference between each clock. Within several minutes, a total difference may become enough to buffer overrun or underrun. Try to adjust cable clock to match device's clock.

Trying to re-install **VAC**, you get an error message saying that a file, folder or device is already present.

Maybe you forgot to uninstall the previous (trial or older) version of **VAC** or uninstallation completed with an error, failing to remove some components. Uninstall it and then carefully follow installation instructions. If you encounter an error, read the uninstallation troubleshooting section.

When Windows returns from a suspended state, audio streams are not resumed or there are error messages.

Going to a suspended (Sleep/Standby/Hibernation) state, Windows closes some or all active application-to-device connections and sometimes disconnects (unmounts) some audio devices. Returning back to working state, Windows tries to connect (mount) the devices back and/or restore application connections. But some devices, especially external ones, need some time and/or some additional conditions to be reconnected. So special non-trivial measures are required for an audio application to resume all streams automatically.

To avoid such problems, don't put your system into a suspended state if audio streaming is active. Don't close audio applications, just stop audio streaming.

Using **Media Player** with **Virtual Cables**, you see **Virtual Cable** is in use after playback ends.

It is known **Media Player** behavior in Windows 6.x. **Media Player** opens playback device on first playback but does not close it until player window is closed. If you need to free **Virtual Cable** device (for example, to change cable format), close **Media Player** window.

Under Win 6.x, DirectSound interface cannot use more than two channels.

This problem occurs because DirectSound in Win 6.x gets channel configuration only from speaker-type pins. Try to enable speaker pin type for the cable.

VAC Control Panel shows that some cables are active but there are no audio applications open.

Unfortunately, Windows keeps no track of applications or processes that use audio devices. There is no way to show a list of applications or processes that use a particular device, you can find them only sequentially or by intuition.

- Check all applications showing their icons in System Tray. Except system icons, each additional tray icon represents at least one hidden application. Do not forget that Win 6.x allows to hide some notification icons (up arrow is displayed in the tray).
- Check if Audio Properties Applets are open. They can keep audio devices open to show signal levels.
- Check if system default playback device is assigned to Virtual Cable device. If yes, system notification sounds (menu/button click, window maximizing/minimizing, device insertion/removal etc.) create audio streams at **Virtual Cable**.
- Check if Listen feature is enabled for any recording device, using **Virtual Cable** as input or output.
- Check if speech recognition is active. If yes, input device may be assigned to **Virtual Cable**.

When more than one **Skype** instances are used with several **Virtual**

Cables, **Skype** freezes.

If you don't use PortCls data processing engine, most likely it happens due to known PortCls bug. Please try affinity restriction technique.

An application accesses **Virtual Cables** via Kernel Streaming interface but sound is broken.

Maybe you have the same problem as with multiple Skype instances. If two applications (and even two threads in the same application) access PortCls simultaneously, it may cause buggy PortCls behavior. Please try affinity restriction technique.

When **VAC** is used with iTunes and/or **QuickTime**, sound is choppy.

iTunes uses **QuickTime** to play audio. To configure audio settings for **iTunes**, you must configure them for **QuickTime** instead.

To configure **QuickTime** audio settings, open its preferences by right-clicking **QuickTime** icon in system tray or by opening its applet in Windows Control Panel. Select **SoundOut** to display audio output settings page. In the playback device field, make sure that DirectSound interface type is selected, not the WaveOut (MME). If **WaveOut** is selected, change it to **DirectSound** and try to play again.

If DirectSound interface is already used but a sound is choppy, click **Options** to open advanced options page. Increase a buffering time in the **FIFO size in millisecs** field to 200..500 ms.

An audio signal passed via the cable becomes very quiet.

If stream format limiting is not used or upper number of channels is greater than in your stream format, System Audio Engine creates its main stream with maximal allowed number of channels and converts your format, lowering channel volume.

To solve this problem, make sure that stream format limiting mode is not set to "None" and correct upper channel limit for a particular cable.

You have selected **Virtual Cable N** for the application but it still uses a different device.

Some audio software developers test their software in a typical system that has only a single audio device. In such case, this device always has ID 0 and is a system default. Even if such application allows to select audio device explicitly, it could have a bug due to that it always uses ID 0 (first available device) instead of particular device ID (1, 2 or more). If such application is tested only in a single-device environment, there are no other IDs than 0 and the bug may not be caught. Since most users, having single-device systems too, will
not report it, the bug may persist for a long time.

A similar bug may occur if an application always used ID -1 (a system default device) instead of real device ID.

If you select a particular device for playback/recording but the application remains to use a first available or system default device, try to contact application's developers and/or support sevice.

As you decrease event/interrupt period duration for some cables, CPU load increases significantly.

In most cases, even 1 ms per event/interrupt does not produce a significant overhead. But some poorly designed applications/drivers that need a small delay, use the "skip current time slice" operation instead of specifying a specific delay time. By default, Windows time slice duration is 15 ms so such applications/drivers don't cause context switches more than 66 times per second. But with, for example, 3 ms per interrupt/event, system timer resolution is set to 1.5 ms and they can switch their contexts up to 666 times per second, increasing their overhead more than 10 times.

You can use **Windows Task Manager** or Process Explorer utility from SysInternals to see which processes (and their threads) cause most context switches in the system and try to exit them for a while.

The same cable/application configuration under server OS works worse than under desktop OS.

Windows Server operating systems use different scheduling policies than Workstation (desktop) ones. In particular, default thread execution quantum (time slice) in server OSes is several times longer (usually six times). It allows server threads to make more work within a single execution period but decreases responsiveness to real-time events and audio/video streaming stability.

You could try to decrease the quantum using the **System** applet in Windows Control Panel. Open **Advanced System Settings** dialog, select **Advanced** tab and click **Settings...** button in the **Performance** frame. In the next dialog, select **Advanced** tab. If the **Background Services** option is selected in the **Processor Scheduling** frame, change selection to **Programs**. If **Programs** is already selected, your system is already configured to use a short quantum.

You use Remote Desktop connection (RDP) but there are no **Virtual Cable** devices.

It is a common problem that occurs if remote sessions are used. Please read about remote access compatibility issues.

When trying to use Kernel Streaming version of **Audio Repeater**, Windows crashes.

At first, please configure your system to create minidump files. They help to isolate the error.

Additionally, you can download either Sysinternals' sync utility or FlushVol utility to flush file buffers to disks immediately before clicking "Start" on Audio Repeater. If there will be BSOD, flushing the buffers will minimize a chance of information loss.

At second, please determine an error source. Audio Repeater is a user-mode application that cannot cause a system crash (the blue screen, BSOD). It can be caused only by a kernel-mode code like a driver. Since Audio Repeater works with audio devices via their audio drivers, each driver is possible to cause a crash. So please try KS version of Audio Repeater to transfer from/to your hardware audio devices only, not selecting **Virtual Cable** devices at all. If there will be a crash, it means a

bug in hardware card drivers, not in **VAC** driver. If so, please contact audio card manufacturer for driver update. You may also attach some minidumps to help the manufacturer to fix the bug.

If KS version of Audio Repeater works normally with your hardware audio devices, try KS version with **Virtual Cable** devices only (for example, from VC1 to VC2). If there will be a crash, it will mean that bug is located in **VAC** driver. In such case, please send a couple of latest minidump files for investigation.

Using **VAC** to route audio from the microphone, you hear an unwanted echo.

An echo can appear if your audio connection scheme has some loopbacks and/or multiple repeaters. For example, each audio card can repeat (monitor) an input signal to its own output (the speakers). If monitoring is enabled (input line is not muted), you will hear your voice in the speakers even there is no audio application running.

If an application records an input signal and also reproduces (repeats) it to the speakers, a signal copy is created, sounding as an echo. This copy appears slightly later due to the buffering. Input audio signal also could be copied to the speakers due to improper **Virtual Cables** assignment in applications.

To avoid an echo, you should disable either input line monitoring in the audio card or recorded data playback in the application. You can control audio card monitoring status (volume and mute) in its volume control panel (the mixer). To disable repeating in the application, see application's settings. Also check your **Virtual Cable** connection scheme to avoid possible loopbacks and parallel signal paths.

You have started wave record, then run **Cakewalk** but there are no **Virtual Cables** in **Cakewalk** port list.

Cakewalk may behave slightly strange, querying audio devices upon startup only. During them, **Cakewalk** queries wave devices to know whether they support various audio formats or not. If wave device does not support a particular format, **Cakewalk** excludes it from available

wave ports.

If the stream format limiting is enabled for a **Virtual Cable**, all wrong formats will be rejected. Thus, stop all activities for all cables whose you want to use in **Cakewalk** and/or disable stream format limiting. Then start **Cakewalk**, wait until it will be ready, and then restore previous settings/activities again.

You want to record a signal from (or play to) your favorite application but it does not allow to set an audio format and starts using an unknown format automatically.

You can use VAC Control Panel which will show you formats that are currently in use with each cable. See the example.

Glossary

Audio Format

A digital audio stream is represented by a sequence of momentary amplitude values, or **samples**. All samples in the stream have the same bitness, or **size**: 8, 16, 20, 24 bits and so on. Sample size defines sample value range and a dynamic range of the digital sound in decibels, which is near to six times larger than sample size in bits.

Audio stream contains various number of parallel **channels**. Usually, audio streams have a single channel (mono) or two channels (stereo). Modern audio hardware and software support five, eight or even more channels (for example, Dolby 5.1, 7.1 etc.). In each sample block, called a **frame**, channel sample values are placed together, from left to right. In a single frame, all channel sample values are sampled simultaneously, in the same moment of time.

A **sampling rate** means how frequently sample values are measured on recording or converted to signal amplitudes on playback. Sampling rate means how many frames (sample blocks, not individual channel sample values) are transferred within a second. This parameter is often called "samples per second" but it is correct only for a mono stream, when a frame contains a single sample value. In the general case, it should be read as "frames per second".

Sampling rate also defines an maximal signal frequency available for coding; it is a half of a given sampling rate, in Hertz. To represent audio signal containing frequencies up to 16 kHz, at least 32000 samples (frames) per second is required.

A combination of the sampling rate, sample size and number of channels is called a **digital audio format**.

An audio format can be shortly specified as these three values: 44100/16/2 or (44100, 16, 2) means 44100 samples per second, 16 bits per sample and two channels (stereo).

Multichannel (more than 2 channels) formats also contain a **channel configuration (distribution)** parameter.

Channel/speaker configuration (distribution)

For mono and stereo audio data formats, only known, dedicated speaker placements are available: for mono, there is only a single speaker; for stereo, there are only two speakers or headphone parts. So it is not necessary to specify their placement additionally.

For multi-channel formats, there can be several speaker placement schemes for the same number of channels. For example, old 5.1 speaker configuration used back channels (5.1 back) while modern configuration uses side channels (5.1 surround) instead. Therefore, it is not enough to specify only a number of channels; you need to specify channel-to-speaker relations as well.

VAC supports the following audio channels:

Abbreviation	Location	Hex mask
FL	Front Left	1
FR	Front Right	2
FC	Front Center	4
LF	Low Frequency (Subwoofer)	8
BL	Back Left	10
BR	Back Right	20
FLC	Front Left of Center	40
FRC	Front Right of Center	80
BC	Back Center	100
SL	Side left	200
SR	Side right	400

Hex mask represent a bit mask corresponding to a single channel, in hexadecimal form. To get a mask for several channels, add their masks together, using Windows calculator in HEX mode. For example, a mask for FC+BR+SL channels will be 4+20+200 = 224 hex (0x224 in popular C/C++ notation).

Channel data (sample values) in the frame are always arranged within the frame in the same order as listed in the table above. So left channel samples always precede right channel samples, subwoofer channel samples always precede back channels samples, and so on.

For details, see a description on the Microsoft site (enter "KSAUDIO_CHANNEL_CONFIG" in the search field if no results are displayed). Additional information can be found here (enter the "channel mask" if no results are displayed).

See here how to set speaker configuration for an audio device.

Audio data encoding

Sample values can be represented using various encoding methods. A simplest and most widely used method is the Pulse Code Modulation (PCM) when a numeric sample values immediately represents a linear absolute signal amplitude.

There are compressed encoding methods (ADPCM, a-law, u-law, MPEG, WMA etc.). They allow to compress audio data but require more resources to process them and frequently cause quality loss.

VAC supports only fixed-point PCM encoding.

Format conversion

Audio format conversion (resampling) is a particular case of data conversion in audio streams. To convert an audio data stream with a given format to a stream with another format, the following actions should be performed:

- Sampling rate conversion if sampling rates are different. Usually, it is

a most time-consuming operation.
- Sample size (bit depth) conversion. It is the lowest time-consuming operation.
- Channel set conversion. Depending on the source and destination channel sets, it can consume less or more processing time.

See here for some known format conversion issues.

Audio interface

To communicate with an audio devices, each application must use some **interface** presented by Windows. Each interface consists of a set of functions and is restricted by a set of conditions and rules.

MME (MultiMedia Extensions) is an oldest audio interface introduced in Windows 3.0. it is intended for a streaming audio and has relatively high latency. Under Windows 9x/ME, **MME** interface uses old-style 16-bit code, as some other operating system parts. But **MME** is simple, exists in all Windows versions since 1991 and its behavior had not been changed. **MME** supports only shared access mode. Audio data are passed via buffer chain passed by a client. Most audio applications can use **MME** interface. Windows limits port name length to 31 characters.

DirectSound interface has been introduced in Windows 95 as a part of the DirectX acceleration set. It combines a low-level, hardware-close, low-latency audio operations with a high-level, device-independent programming. Audio data are passed via circular buffer. Originally intended for games, **DirectSound** interface quickly became very popular in sound synthesis and recording/playback applications. In Windows 5.x, **DirectSound** supports both shared and exclusive access modes. In Windows 6.x, **DirectSound** acceleration features (the exclusive access mode) are not supported anymore so interface efficiency became the same as **MME**.

Kernel Streaming (or **WDM/KS Audio**) interface has been introduced in Windows 2000/98. **WDM** stands for the Windows Driver Model, an universal driver structure and behavior making available to use a common binary driver file in Windows 98/ME/2000/XP and higher systems. **KS** stands for the **Kernel Streaming**, the audio/video streaming

technology of the Windows kernel. **WDM/KS Audio** is a sophisticated lowest-level interface including many new features, it allows to achieve highest audio precision and lowest latency. In Windows 2000/XP and later, all higher-level interfaces are implemented on top of the **WDM/KS Audio** interface. **WDM/KS** supports only exclusive access mode. In Windows 5.x, audio data are passed via buffer chain (**standard streaming** or **legacy mode**); in Windows 6.x, circular buffer mode is also supported (**looped streaming** or **RT Audio** mode). Streaming mode support is dependent on a particular driver.

WASAPI interface has been introduced in Vista. The abbreviation stands for Windows Audio Session API. It supports both shared and exclusive access modes. In the exclusive mode, it is highly efficient like hardware-accelerated **DirectSound** but in shared mode, its efficiency is comparable with shared-mode **MME** and **DirectSound**. Main **WASAPI** advantage is that it has a modern style, object-oriented, flexible, allows to use local and global effect processors (LFX and GFX) and provides a way to insert user-defined processing objects (APOs) in a signal path. Audio data are passed via circular buffer.

Only **KS** interface allows direct driver connection with no intermediate layers. An application having properly-implemented **KS** interaction can achieve highest quality and efficiency. Only driver and/or hardware bugs may affect signal quality and latency.

Other interfaces work via System Audio Engine. **WASAPI** in exclusive mode has almost the same efficiency as **KS** but may be affected by system layer implementation (for example, there are some problems in Win7 fixed in Win8).

Four interfaces described above are Windows standard ones. Additionally, there are some third-party interfaces intended for professional audio processing. Most popular of them is Steinberg's **ASIO** (Audio Streaming Input/Output). **ASIO** is designed for extreme performance and precision. **VAC** doesn't directly support **ASIO** but there are some **ASIO** wrappers over **WDM/KS** like ASIO4ALL or ASIO2KS.

Audio Port

Each audio device in Windows is represented by its **port**. The port is a "logical" or "software" device connection point in the system like network ports. Don't confuse software ports with hardware input/output ports.

The "port" term is almost not used in official documentation; the "device" term is used instead. It often causes misunderstanding because a single hardware device is usually represented by multiple software connection points in the system. So **VAC** documentation uses the "port" term to represent a "software connection point".

Audio device ports are called **Wave ports**. By example, a typical audio card has two wave ports: **Wave Input** and **Wave Output**. The recording port is used to record (capture) audio data **to** applications, and the playback port is used to play (render) audio data **from** applications.

Each port is served by an appropriate **provider**. Low-level ports (pins) are provided by low-level device drivers and high-level ports (endpoints) are provided by System Audio Engine.

Audio ports are accessed via audio interfaces. Different interfaces can expose different port sets. For example, for a single WDM/KS port that represents a pin, higher-level interfaces may create several logical endpoints that represent separate source lines.

For recording/input ports, the "capture" synonym can be used. For playback/output ports, the "render" synonym can be used.

Pin

In the WDM/KS driver technology, each device driver exposes a set of **pins**. Each pin represents a device connection point: recording/capture, playback/render, volume control/mixer, clock and so on. A pin is a low-level synonym of a port. The "pin" term is similar to the "lead" in electronics.

To use a pin for audio streaming, Windows creates an **instance** of the pin by "opening" it. Each pin instance forms a separate audio stream. If an application accesses the pin directly, using WDM/KS interface, the instance is created for the application itself, nobody else can use this

instance. If an application accesses the pin in the shared mode through System Audio Engine, the engine creates a single instance for itself and then shares it among all connected applications. It allows single-client drivers to be used by multiple applications at the same time.

If the driver supports multiple pin instances, there can be multiple clients of a corresponding port. Most drivers support only a single pin instance so there can be only a single client (System Audio Engine or a WDM/KS application).

Source line

A typical audio adapter has a single digital recording channel that application can use to capture audio signal but several source (input) lines like Microphone, Line, Phone, CD and others. To capture audio signal coming from a particular source line, this line should be selected first. In Windows 5.x, you need to use Windows Mixer application to select a source line. Windows 6.x creates audio endpoints for this purpose.

As a virtual device, **VAC** has no real source lines. But some applications want to connect to a given line **type** (Microphone or S/PDIF) so **VAC** provides several **fake** source lines. All of them are identical but Windows can use different default settings for each line. For example, Windows 6.x uses less sampling rate recording from the Microphone line than from the Line In.

Source line set is a cable configuration parameter.

Audio endpoint

Windows versions prior 6.x create recording (capture) waveform audio port for each capture waveform device or pin exposed by a driver. This recording audio port represents entire device that may have several source lines. Selecting such port in a recording application was not enough to record audio signal from a particular source line. A device mixer should be configured to select (connect) an appropriate source line before recording starts.

To eliminate additional steps, Windows 6.x introduced a new thing called **endpoint**, or an "**endpoint device**". Instead of creating recording audio port from an entire device, Windows creates a separate endpoint from each device's source line, naming them like "**Mic Volume (AC97 Audio)**" or "**Line In (AC97 Audio)**", and exposes them as a recording port set. When an application selects a particular endpoint as a recording port, Windows automatically selects the appropriate source line in the device's mixer.

Since source lines in most audio devices are multiplexed, not mixed, only a single endpoint can be used for recording at a time. If a recording endpoint is used for recording, all other device endpoints multiplexed in the same group become unavailable.

VAC devices and their source lines are software-emulated so they are represented as "mixed" and several applications can record from different cable endpoints simultaneously.

Some endpoints can be connected (plugged in) while other are unconnected (not plugged in). With a real hardware device, an endpoint can become connected when a connector is plugged into a jack. Only connected endpoints are available for playback/recording and visible in application's audio device list.

You can view all audio endponts and their states in Windows Audio Properties applet, right-clicking on any item of **Recording** or **Playback** lists and enabling "**Show disabled endpoints**" and "**Show disconnected endpoints**".

Please note that endpoint creation in Windows 6.x is a **very** slow process. For example, creation of 30-40 endpoints (when a driver is initially loaded or restarted) may require up to a minute of 100% CPU load on 2 GHz machine.

Also please note that all **Virtual Cables** with enabled speaker pin type will have the same playback endpoint name, like "**Speakers (Virtual Audio Cable)**". It is by system design, not due to a bug.

Stream (pin instance)

In general, a term "**stream**" is used to designate any kind of data stream (a continuous data flow).

For the audio application, the "stream" means an audio connection with a data flow to (or from) an audio device.

For a WDM/KS driver, this term is used to designate a data flow connected to a particular pin instance by request from a client. Each new pin instance forms a new stream. Each stream has its own audio format. An application can be connected to a stream either directly or via the System Audio Engine proxy layer.

Client

The application that uses an audio port/endpoint and opens it, connecting to its provider, is called a **client**. Most low-level audio drivers allow only single client (instance) for each pin but System Audio Engine supports unlimited number of clients. **VAC** driver allows any number of clients to open each pin directly (in the exclusive mode).

If an application requests a port connection several times without closing it, it means that port provider has several clients. In other words, each port opening request is treated as new client appearance. Client disconnects from the provider when it closes a port instance.

When an application accesses low-level audio port (pin), provided immediately by a low-level driver, by WDM/KS interface, it becomes a driver's client. When an application accesses a high-level audio port (endpoint), it becomes System Audio Engine's client. In turn, System Audio Engine accesses a low-level port (pin) and becomes low-level driver's client.

Audio buffers

All audio interfaces use audio **buffer** conception to interchange audio data between applications and devices. A buffer is a memory block containing an audio data fragment or intended to be filled with such data fragment. Data buffering is used to make data transfer more smooth and

reliable.

A smallest data unit that a computer can transfer between a device and main memory is a word. In audio, each computer word contains only 1-2 frames of 16-bit stereo sound. It is extremely small amount for real-time streaming. So audio frames are packed in blocks containing thousands, hundreds or (as an exception) dozens of frames each.

In MME and legacy WDM/KS, a **buffer chain** algorithm is used to interchange audio data between an application and the audio subsystem; application sends several buffers to driver, driver plays them or fills them with data, and returns them back to application. An application must use more than one buffer because some time passes between a moment when a device driver inform about buffer completion and the moment when it receives a next buffer from the application.

In DirectSound, WASAPI and modern (RTAudio) WDM/KS, **ring buffers** are used; application allocates a single memory block logically divided into several parts, tells the driver which part is ready for playback/recording, and the driver notifies the application when part processing is done.

To maintain a smooth and continuous audio stream, an application must provide enough buffering time. Most applications use 500.1000 ms of buffering time, divided into 8..12 chain buffers in **MME** or to 4..12 circular buffer parts in **DirectSound**, **WDM/KS** or **WASAPI** . But the more buffering time, the more latency is introduced. So you need to balance these parameters for each configuration used.

Cable format

When a **Virtual Cable** gets its first input or output client (stream), a particular audio format is chosen as the **cable format**. Cable format parameters are determined from a first client (stream) format and the cable format range.

Cable format is fixed while a cable is active (has at least one client/stream).

Internal mixing is performed in the cable format so all render/output stream data are converted to the cable format, and mixed results are converted from the cable format to capture/input stream formats. If cable channel mixing is enabled, **VAC** also converts channel sets in multichannel streams.

Cable format range

A range of audio formats allowed to be selected as a cable format. The wider this range is, the more formats are available to be chosen for a cable, the more often format conversion occurs. Format range is a cable configuration parameter.

Cable channel mixing

By default, **VAC** converts channel sets in multichannel streams if number of channels in stream and cable formats are different. For example, when converting from stereo stream to mono, **VAC** mixes (sums) both left and right channels, producing a single channel stream. When converting from mono stream to stereo, **VAC** spreads a single channel into left and right. When converting 4-, 6- and 8-channel streams, more complex rules are applied.

If channel mixing is disabled, **VAC** performs channel scattering (placing sequentially packed channel data to specified channel configuration positions) or gathering (extracting specified channel configuration positions and placing them to a sequentially packed set) instead of mixing them. See here for details.

In some situations, disabled channel mixing may produce undesirable effects.

Channel mixing mode is a cable configuration parameter.

Stream format limiting mode

A rule to limit a new stream format. Mostly used to control automatic format selection features of System Audio Engine. See here for details.

Limiting mode is a cable configuration parameter.

System audio engine (formerly KMixer)

KMixer (kernel-mode audio mixer) is a system kernel-mode audio component (a special kind of an audio driver), a part of the Windows 98/ME and 2k/XP/2k3 audio subsystem. In Vista/Win7, **KMixer** has been replaced by user-mode **AudioDG** service. For convenience, it is named **System Audio Engine** here.

System Audio Engine acts as a "proxy" to each WDM/KS audio driver accessed via DirectSound, MME and WASAPI interfaces in the shared mode. When an application uses shared connection mode, a separate pin instance is implicitly created to the **System Audio Engine**. See Audio layering issues for details.

In some cases, restarting **System Audio Engine** may help to eliminate some audio problems without rebooting the entire system.

Shared and exclusive pin access

Most audio device drivers support only a single instance of each capture or render pin (they are single-client drivers). To allow to access these pins from several applications at the same time, an intermediate (proxy) layer is required. In Windows, this layer is provided by the System Audio Engine: MME (all) and DirectSound/WASAPI (by default) connections are established in the **shared** mode when the engine creates a single pin instance for itself and all clients are connected to the engine, not immediately to the pin. System Audio Engine chooses an appropriate format for the pin instance, and then converts audio data between pin format and client stream formats. This mode is convenient but often not efficient enough.

DirectSound (in Windows 5.x), WASAPI (in Windows 6.x) and WDM/KS (in all systems) support **exclusive** pin access modes while the pin instance is created for a requestor application only. No other clients (applications and even system sounds) are allowed to share this instance. The pin is instantiated with the format requested and no format

conversion is performed between client application and the driver. This mode is efficient but not convenient enough because there can be only a single application that can use the pin at a time. If the driver supports multiple pin instances like **VAC**, there is no such restriction.

Implementing multi-client pin access, **VAC** behaves like **System Audio Engine** in the **shared mode**, mixing playback streams together, distributing cable data among recording streams and performing format conversions. So most efficient **VAC** usage method is to use **exclusive** access modes when connecting to **Virtual Cables** is possible.

PortCls

PortCls stands for "Port Class Driver". It is Windows kernel-mode library (portcls.sys) that should be used by each audio miniport driver to connect to the system. **PortCls** receives all system requests, translates it and passes to a miniport driver.

In Windows XP and later systems, on a multi-CPU/core hardware, **PortCls** has some bugs. To avold these bugs, **VAC** implements a workaround, processing most streaming requests without calling PortCls. Processing can be switched back to PortCls engine for particular cables using cable configuration parameters.

Latency

When an application and a driver deal with digital audio, they don't interchange single samples/frames; instead, they use memory buffers to store blocks of audio data. To respond to a real-time event, an application or a driver needs some time, from microseconds to dozens of milliseconds. Due to buffering and processing delays, there are some time between an audio signal arrives at a device input and its digitized value arrives in application memory. This time period is called **latency**.

Timing event (formerly an Interrupt)

Due to a discrete nature of the digital audio, continuous digital audio stream transfer is performed in series of blocks. To transfer a stream from

each cable's output port to its input port, **VAC** has internal timing clock that generates system events allowing **VAC** to be called to transfer a next data block of the stream. Earlier **VAC** versions used timer interrupts for that. Current versions use timer events but the "interrupt" term is kept for a succession.

Interrupt/event period duration is a cable configuration parameter.

The more is event frequency, the shorter is event period, the less is block size, the smoother is stream transfer. But decreasing interrupt/event period causes increase of system timer resolution and timer interrupt frequency so system overhead is increased too. **VAC** driver sets system timer resolution to a half of the "MS per int" parameter specified.

Mixer (volume control tool)

In general, a **mixer** stands for a device that mixes several audio signals together and allows to control their volume, balance, timbre and other parameters. Almost each audio adapter has its own mixer that allows to select recording sources, change recording/playback levels, and so on. These features are called a "adapter mixer" or a "driver mixer".

Windows prior to Vista/Win7 have a standard **mixer control application** that is often called "Windows mixer". It can be invoked by double-clicking at the speaker icon in the system tray. A window with a set of sliders and check boxes is displayed. You can view the mixer in two modes: playback (the default) and recording. In playback mode, the mixer controls output audio lines available for playback and repeated (monitored) input lines routed to the output (speakers). In recording mode, the mixer controls input lines available for recording.

To configure mixer panel, open **Options** menu and select **Properties**. In the dialog, select a device you want to control, select a mode and check input/output lines you want to see in the panel.

In Windows 6.x, system mixer became much more simpler. It can be opened by right-clicking the speaker icon in the system tray and selecting "Open Volume Mixer". You can change playback/recording levels for devices selected under an icon or for system/application sounds. Clicking

on speaker icons at the bottom, you can mute/unmute audio sources.

Don't confuse Windows mixer control application with KMixer (System Audio Engine) Windows component.

System tray

System tray (notification area) is a rightmost area of the horizontal taskbar or a lowest area of the vertical taskbar, where system clock and application icons are located. In Win 5.x, this area always shows all existing icons. Starting from Win 6.x, some icons can be hidden. If there are hidden icons, double up arrow is displayed in the notification area.

Windows Control Panel

Windows Control Panel is a set of Windows built-in control and management applets. You can open it by clicking **Start** - **Settings** - **Control Panel**.

Don't confuse **Windows Control Panel** with VAC Control Panel.

Audio Properties Applet

Audio Properties Applet is a built-in Windows Control Panel applet that manages and controls multimedia and audio devices. The applet can be opened by two ways:

- If there is a speaker icon in system tray, right-click it and select **Playback/Recoding devices** (Windows 6.x) or **Adjust Audio Properties** (Windows 5.x).
- Open Windows Control Panel and open **Sound** (Windows 6.x), **Sound and Audio Devices** (Windows 5.x).

On the **Audio** tab, you can view and change default playback and recording devices, adjust DirectSound acceleration level, set speaker configuration and so on.

Windows Device Manager

Device Manager is a built-in Windows Management Console applet, displaying a device list and allowing to configure/reinstall them. Device Manager can be opened by several ways:

- Right-click on **My Computer** and select **Manage**.
- Right-click on **My Computer**, select **Properties**, open **Hardware** tab and click **Device Manager**.
- Open Windows Control Panel, open **Administrative Tools** then open **Computer Management**.

Clock

Each streaming device, both hardware and software, needs a clock source to transfer streaming data smoothly and uniformly. After each time period measured by the clock, a portion of streaming data is sent or received.

An asynchronous device (a computer, audio card, radio etc.) has its own clock source (a clock generator). A synchronous device (also called "slave") receives a clock signal from a master device having its own clock generator. Some audio devices (for example, professional audio cards) having their own clocks and normally working asynchronously, allow to use an external clock source to work synchronously with en external device.

When the CPU interacts with an audio adapter (on-board chip, a card, an external USB adapter) to send/receive a command or data block, CPU clock is used as a source, and the audio adapter acts as a slave (synchronous) device. But in audio streaming mode, adapter's clock is used (except of professional adapters that support external clock source) and CPU acts as a slave device.

In other words, the CPU determines when to send/receive a command or data block but an adapter determines when playback/recording of data portion completes. As the streaming has been started, CPU cannot instruct the adapter to move streaming data slower or faster; it only can

change the sampling rate if the adapter supports sampling rate changing on the fly.

Clock difference

Two asynchronous devices, each of them has its own clock source, cannot have exactly the same clock frequency (rate, or speed). Since their clocks are different, their rates always are slightly different, and actual sampling rates are slightly different too. As a pair of a real watches, two asynchronous audio devices cannot have their data streams to flow in a strict match, and one stream slowly forestalls while other stream is late. As a result, there will be some gaps or losses in the overall audio data stream.

VAC supports clock correction feature to minimize this effect.

System default (preferred) audio device

Windows supports up to 256 different MME/DirectSound/WASAPI audio devices (ports) for audio input and output. Any of them can be assigned as a **system default**, or a **preferred** device that is used by default. Such assignment can be performed using Audio Properties applet. In application's device selection menus, a default device appears as the "Microsoft Sound Mapper", "Wave Mapper", "System Default" or something like.

If an application issues a request to a default device, Windows routes it to an actual device that was previously set as a default.

In Windows 5.x, default audio devices are set at a per-user basis. For each user account, there are its own default recording and playback default device settings. In Windows 6.x, default devices are set at per-system basis.

See the system default device issues for details.

Default audio format for a device

To support shared access mode, System Audio Engine creates a single, common pin instance for all client applications, using a common audio format. But the engine does not know which formats will be used in the future so it cannot choose a best format automatically.

In Windows 5.x, common format selection is based on the first connection request. For a recording request, the common format is the same as requestor's format. For a playback request, the common format is most "wide" format supported by the pin. If the pin supports high sampling rates, bit depths and number of channels, using most usual formats involves unnecessary format conversions, overhead and audio degradation.

To control these issues, Windows 6.x maintain a **default audio format** for each playback and recording device. When a device is opened in shared mode, System Audio Engine always uses its default format. So you can freely choose between default formats to achieve either best audio quality or best performance or best compatibility.

Default formats can be chosen in the "Sound" applet from Windows Control Panel. Open "Playback" or "Recording" tab, double-click a device and select "Advanced" tab.

Listen feature

Starting from Win7, Windows implements the **Listen** feature intended to monitor (hear a signal from) a recording audio device. When this feature is turned on for a particular recording device, System Audio Engine starts to continuously record from the device and immediately play back recorded signal to the given playback device. Thereby this feature is similar to Audio Repeater application.

To use Listen feature for a device, open the Audio Properties Applet, select **Recording** tab, double-click an endpoint, open the **Listen** tab, check the "**Listen to this device**" checkbox and select desired output device (usually speakers).

With this feature, you can either listen for a signal coming from a **Virtual Cable** (enabling Listen feature for a source endpoint for the cable and

choosing headphones/speakers as a target device) or supply a **Virtual Cable** with a signal (enabling Listen feature for a source device and choosing **Virtual Cable** as a target device).

Once you turned Listen feature on, it remains active until explicitly turned off. Since there is no visible activity indicator, it is easy to forgot about it. If you have wrong/unwanted audio signals played back and/or recorded, make sure that there is no Listen feature activated by mistake for some devices used.

Speaker pin type

WDM/KS drivers of most audio adapters provide KSNODETYPE_SPEAKER type for their playback pins. Such pin type allows to configure channel distribution with the "Configure" button of Windows Audio Properties Applet but System Audio Engine starting from Win 6.x always assigns the "Speakers" name for endpoints linked to these pins. If an audio adapter (real or virtual) has more than a single output line, it is impossible to distinguish them by name.

As a workaround, **VAC** uses KSNODETYPE_LINE_CONNECTOR for playback pins by default. It allows to use an unique name like "Line N" for each output line but does not allow to use system channel configuration features. Additionally, some channel processing problems may occur in applications using DirectSound.

To control playback pin type, use the "**Enable spk pin**" parameter in cable configuration section of VAC Control Panel.

Please note that enabling speaker pin type for some cables under Win 6.x causes playback endpoints of these cables to have the same "Speakers" name. Most audio applications distinguish between audio devices only by their names, not unique internal identifiers. Therefore, if you enable speaker pin types for two **Virtual Cables** and both these cables have the same "**Speakers (Virtual Audio Cable)**" name, audio application might confuse between these cables. Most probably an application will lose proper cable selection between running sessions.

Administrator account

A privileged (superuser) account in the system. Applications that run from this account can perform various administrative actions. Since **VAC** is a

device driver, a special privilege is required to install it into the system or to restart it, applying a configuration change.

In Windows 5.x, any member of the "Administrators" group has such privilege and can perform these actions.

In Windows 6.x, there is UAC feature (enabled by default). Only built-in administrator account (named "Administrator") can perform all privileged actions without additional measures. Other accounts marked as "administrator" are "virtually privileged" and can perform privileged operations only by privilege elevation. The difference is only that a plain user account requires administrator password for privilege elevation while account marked as administrator does not.

If UAC is enabled, Windows tries to automatically elevate privilege level for applications marked as privileged (such applications cannot work from a non-privileged account). **VAC** installer and uninstaller are marked as privileged applications so Windows will ask for privilege elevation if you don't have enough privileges.

But VAC Control Panel is not marked as privileged because it can perform both privileged and non-privileged actions. So if you run **VAC** Control Panel from an account other than "Administrator" (even marked as administrator), only non-privileged actions will be available. You will not be able to perform actions that require driver or System Audio Engine restart.

To start an application behalf on the built-in Administrator account, either log in using "Administrator" as a user name or right-click an application icon and select "Run as administrator" item.

Remote session

Initially, Windows supported only local work session when a user sits immediately at the console (display, keyboard, and mice) of a computer running Windows. Starting from Windows 2000, remote sessions were introduced, when a user sits at a "terminal" (or a "client") computer but all user's actions are sent to a remote "terminal server" computer running Windows and the results (screen contents, sounds etc.) are sent back to

a client computer. Terminal connections are established via Remote Desktop Protocol. Using such connection, a user can work with a remote computer as if he/she worked with it locally. But there are some limitations.

There are two types of Windows remote connections: **separate** (isolated) and **direct** ("console"). For a separate connection, Windows establishes a new login session, isolating user environment from other users. For a direct connection, remote user is connected to a main console session environment.

With desktop-sharing products like TightVNC, TeamViewer, LogMeIn, Remote Administrator, connections are established directly by translating screen updates from the server to your computer and translating your mouse/keyboard actions from your computer to the server. No separate logon session is created and your computer acts as a "remote console" for the server computer. Having such connection, you can use all server resources, feeling yourself like sitting at the server immediately.

There are some compatibility issues related to remote connections.

Virtualized environment

An environment created by virtualization tools is called "virtualized environment".

Different virtual machine software offer different layers of platform virtualization. Some platform virtualization products offer full hardware virtualization (VMware, Virtual PC, Virtual Box, Parallels Workstation, Virtual Iron, Bochs) while most other products like Parallels Virtuozzo (mostly used in Windows VPS/VDS technology) offer only a partial virtualization (for example, OS level virtualization).

There are some compatibility issues related to virtualized environment.

Digital signature

Digital signature is a special digital sequence to check data authenticity. If

data are illegally modified (tampered with), their signature becomes invalid and it can be easily checked.

Windows uses two types of digital signatures for executable (PE) files:

- **Publisher** signature is applied directly by a software publisher (vendor) using a publisher certificate and certify the file as authentic one. A publisher certificate is obtained once and then used to sign all developed software. Publisher certificates can be obtained from many companies but only six of them can issue certificates to sign Windows kernel-mode executables. These six companies issue publisher certificates only for companies (juridical persons), not for individuals (natural persons). Therefore, individual software developers cannot obtain a publisher certificate.
- **Windows Logo** signature is applied by Microsoft Windows Hardware Quality Lab (WHQL) and certify the file as passed Windows-compliance tests. A software package is submitted to WHQL to test and if tests are successful, WHQL signs executable files as Windows-compliant. Windows Logo signature requires a publisher signature. WHQL must sign each software release, even a minor bugfix.

Buffer overflows/underflows

A buffer **overflow** occurs if the buffer has no room for data should be placed into it (some data will be lost from a flow). An **underflow** occurs if the buffer has not enough data should be retrieved from it (data flow will contain a gap). If a buffer is filled at one side and drained at another side, overflows will occur if the filling flow is more "thick" than the draining one, and vice versa.

Compared with car fuel tank usage, an overflow occurs if your car tank has no more room to accommodate fuel provided via the filling hose. If the hose has no special protective cutoff, fuel excess will flow out and you lose it. On the contrary, if you don't fill a tank at the proper time, it runs out and your car will stop suddenly, requiring a maintenance.

In a multi-node or multi-layer chain, stream data overflow/underflow not necessary means a crack, pop or glitch. On the contrary, no data

overflows/underflows do not mean that there are no cracks, pops or glitches. Data processing node/layer (a driver, an application, a service, a plugin etc.) that registers an overflow/underflow knows nothing about other nodes/layers in the chain; if they use extensive buffering, it may prevent audio stream from breaking. Therefore, if a node/layer had successfully passed data portion to another node/layer in the chain, there may be buffering problems in other nodes/layers, causing stream breaks.

Affinity restriction

PortCls system driver has a bug causing a significant drop of audio data processing efficiency on multi-CPU or multi-core hardware. **VAC** implements multiple subdevices where most audio drivers implement only a single subdevice so this bug does not occur with most hardware audio drivers.

The problem is described in details in **microsoft.public.development.device.drivers** newsgroup. You can find the post, entering "skype portcls getmapping" in Google search string. Sorry, there is no stable link.

The best way to avoid this problem is to use **VAC** internal data processing engine instead **PortCls** one (internal engine is used by default).

If **PortCls** engine should be used for some reason, only possible workaround is to disable concurrent request processing in PortCls. For this, **VAC** offers some driver parameters and cable parameters to restrict CPU affinity of worker threads, belonging to System Audio Engine or to any client. Create the appropriate driver parameter value in the registry, setting it to a number of cables causing the problem, or create affinity restriction parameters for selected cables. Then restart **VAC** driver to propagate parameter changes.

Worker thread

A **worker thread** is a driver's thread that performs all stream processing work (data transfer, volume control, format conversion etc.). Different

threads can be executed on different CPUs/cores, achieving optimal performance and load distribution. By default, **VAC** driver starts one worker thread per physical CPU/core so all available CPU power can be used to process stream data.

By default, worker thread scheduling priority is relatively high (between a normal thread priority and highest possible priority value). It guarantees that stream data processing will take advantage over most regular threads but not to consume all possible CPU time.

You can limit number of these threads and/or change their priority using Control Panel application or modifying driver parameters in the Registry.

Stream buffer watermarks

The watermark technique is used to optimize buffer usage if client applications use non-optimal buffering settings and don't allow to specify them explicitly. In terms of the car fuel tank usage, this technique is like a manager that helps an absent-minded car owner to fill the tank in good time. Usually, a car can be driven until its fuel tank is completely exhausted; a manager detects if fuel level is too low and suggests to refill before fuel is exhausted completely.

In stream buffer watermark technique, stream data buffer provided by a client is like a tank, and the marks are measured in milliseconds of audio data duration, marking minimal and maximal audio data/space reserve limits. If watermark control is enabled, driver does not process stream data until client supplies enough data/room buffers up to high watermark specified (in duration).

Watermark technique does not eliminate buffering problems because **VAC** driver cannot force applications to provide buffers as fast and smooth as possible. This technique only can reduce amount of buffer overflows/underflows, replacing continuous popping/crackling with isolated and less frequent pops/cracks.

You can control this feature using Control Panel application or modifying cable parameters in the Registry.

Stream data buffer

VAC driver uses a small internal data buffer for each stream, used for technical purposes. In most cases, a very small (several milliseconds) buffer length is enough to perform all tasks needed by the driver. But this buffer can significantly improve stream stability if a client application or System Audio Engine does not provide data buffers in time. Please read here for details.

You can control this feature using Control Panel application or modifying cable parameters in the Registry.

Windows versions

Modern Windows NT releases are the following:

- 5.0 - Windows 2000 (Win2k)
- 5.1 - Windows XP (WinXP)
- 5.2 - Windows 2003 Server (Win2k3)
- 6.0 - Windows Vista, Server 2008
- 6.1 - Windows 7 (Win7), Server 2008 R2
- 6.2 - Windows 8 (Win8)

So the "Win 5.x" means Windows 2000/XP/2003 and "Win 6.x" means Vista, Server 2008, Windows 7 and Windows 8.

Windows 9x/ME have 4.x versions and don't belong to Windows NT family.

History

Version 4.14 (30.05.14)

- Fixed a bug caused playback at higher speed.
- Added a workaround for PortCls bug (GetMaxMapRegisters function) causing position freezing.
- Fixed a bug caused signal level to be painted over the header in Control Panel application.
- Added stream buffer control to minimize overflows/underflows.
- Added button to restart System Audio Engine.
- Now filter volume/mute nodes always report 8-channel support.

Version 4.13 (23.07.13)

- Fixed a bug preventing stream buffer watermark parameters from being loaded on restart.
- Fixed bugs causing integer divide overflows in Audio Repeater (both MME and KS versions).
- Added a workaround for PortCls bug causing BSOD (bugchecks 0x50 or 0xD6) in GetMaxMapRegisters function if Driver Verifier with the special pool is used for VAC driver verification.
- Added peak signal level indicators to Control Panel.
- Changed signal level indication in Audio Repeater from average to peak.
- Fixed stream position timestamp error if PortCls is not used (occurred in Open Broadcaster Software).
- Added a feature to control playback pin type (speaker or line out).
- Fixed sleep/hibernate problems.
- Added master volume control.

Version 4.12 (01.04.12)

- Fixed RtAudio buffer processing bug in Audio Repeater KS.
- Added clock rate display to Audio Repeater.
- Changed number of worker threads limit to a number of logical CPUs.

Version 4.11 (11.03.12)

- Added cable multi-selection feature to Control Panel.
- Clock correction precision increased to 0.0000001% (1E-8%).
- Added volume boost feature.
- Fixed sampling rate change bug caused incorrect playback speed.
- Rewritten data processing code from DPC to system thread set to minimize interlocked waiting and improve performance on multi-CPU/core systems.
- Added stream buffer watermark control feature to improve stream stability.
- Added "Reset counters" buttons (Control Panel) to reset cable/driver counters.
- Minimum possible sampling rate increased from 200 to 1000.
- Fixed a bug in KS Audio Repeater (device name string length was limited to 31 chars as in MME version).
- Optimized KS Audio Repeater algorithms for more reliable transfer.
- Added a feature to Audio Repeater: now command-line options can specify device number instead of its name to avoid name collisions if several adapters of the same type coexists.
- Added several channel mixing/distribution schemes.
- Added channel scatter/gather mode and cable channel mixing control.
- Increased clock correction precision (registry value format is changed).
- Added a privilege elevation manifest to Setup application to elevate privileges automatically.
- Maximum sampling rate changed from 1000000 to 384000 in accordance with MS requirements.
- Add "already installed" warning to Setup application.
- Fixed some node property errors.
- Fixed some minor bugs.

Version 4.10 (15.02.10)

- Added IPrefetchOffset interface support (more stable audio in XP/Vista/Server 2008/Win7).
- Added a workaround to avoid Skype freezing and to stabilize

streaming in heavy load cases.
- Changed voice reminder policy in trial version.
- Added tray icon support to Audio Repeater.
- Added Kernel Streaming version of Audio Repeater.
- Added Wikipedia references in the user manual.
- Fixed some bugs in volume control code.
- Fixed a bug caused timer resolution restoration when a last cable stream was closed.
- Added timer resolution correction on return from standby.
- Added load splitting among CPUs/cores.
- Added Authenticode digital signature from NTONYX.
- Added instance closing feature to Audio Repeater.

Version 4.09 (11.06.08)

- Fixed a bug caused system crash in case of incorrect volume settings.
- Now multichannel nodes are exposed correctly (individual sliders instead of the Pan slider).
- Fixed a bug caused break instruction crash (0x80000003) on a floating point format query.
- Fixed a bug caused system crash in low memory situation.
- Added a test signature to 64-bit driver module.
- Increased average time interval between voice reminders in trial version.
- Increased clock correction precision in Control Panel to 1/1000th of a percent.

Version 4.08 (14.09.07)

- Clock correction amount is now represented and specified in percent values.
- Changed clock correction algorithm to increase sampling rate precision.
- Fixed a bug in the INF file that prevented Virtual Cables from being used via KS Proxy (for example, as a DirectShow WDM filter).
- Fixed a bug in format checking code that limited number of channels to 2 under Vista.

Version 4.07 (28.06.07)

- Removed another debug break instruction caused a bugcheck in rare cases.

Version 4.06 (15.05.07)

- Fixed installer bug caused the driver to stay disabled after installation.

- Removed a debug break instruction caused a bugcheck in rare cases.

Version 4.05 (12.05.07)

- Added a clock correction feature.
- Added configuration options for source line set (affects endpoint creation in Vista).
- Changed service name in INF file to eliminate conflicts with VAC 3.
- Added more correct multi-channel support under Vista.
- Added a workaround for Vista portcls.sys bug (a miniport stream can be destructed without been stopped).
- Removed the "Mic" source line (capture endpoint in Vista) from the default source line set.
- Changed "Milliseconds per interrupt" default value to 7 to improve DirectSound stability.
- Improved stream stability on short buffering times.

- Changed the Control Panel interface.

Version 4.04 (31.12.06)

- Fixed a bug in topology description (fake recording controls were not accessible).
- Fixed some synchronization bugs (the system hangs while several cables are heavily used).
- Added a pitch (frequency) shifting support.

- Added a limited Vista support.

Version 4.03 (16.07.06)

- Fixed a bug that caused change of signal pitch. This issue was previously considered as DirectSound bug and was described in FAQ section related to speech engines. This section has been removed from the FAQ.

Version 4.02 (22.05.06)

- Added 64-bit binaries.
- Some minor corrections.

Version 4.01 (24.03.06)

- Fixed some bugs.
- Changed default stream format limiting mode to the "Cable range".

Version 4.00 final release (06.03.06)

- Fixed some bugs.
- Added volume control features.

Version 4.00 beta 6 (27.02.06)

- Fixed some bugs.
- Added format conversion and stream format limiting features.

Version 4.00 beta 5 (16.12.05)

- Fixed some bugs.
- Added a manual file.
- Installer/uninstaller improvements (administrator rights check,

uninstallation entry creation in Add/Remove Programs).

Version 4.00 beta 4 (18.08.05)

- Fixed some bugs.

Version 4.00 beta 3 (23.07.05)

- Fixed some bugs.
- Added installer/uninstaller.

Version 4.00 beta 2 (15.07.05)

- Fixed some bugs.
- Added Control Panel application is available.

Version 4.00 beta 1 (07.06.05)

- First beta release (without an installer).