VideoLAN Streaming Howto

Alexis de Lattre

Johan Bilien

Anil Daoud

Clément Stenac

Antoine Cellerier

Jean-Paul Saman

VideoLAN Streaming Howto

by Alexis de Lattre, Johan Bilien, Anil Daoud, Clément Stenac, Antoine Cellerier, and Jean-Paul Saman

This document explains how to stream, transcode and save streams using the VideoLAN solution

Table of Contents

Chapter 1. Streaming, Muxers and Codecs

Introduction

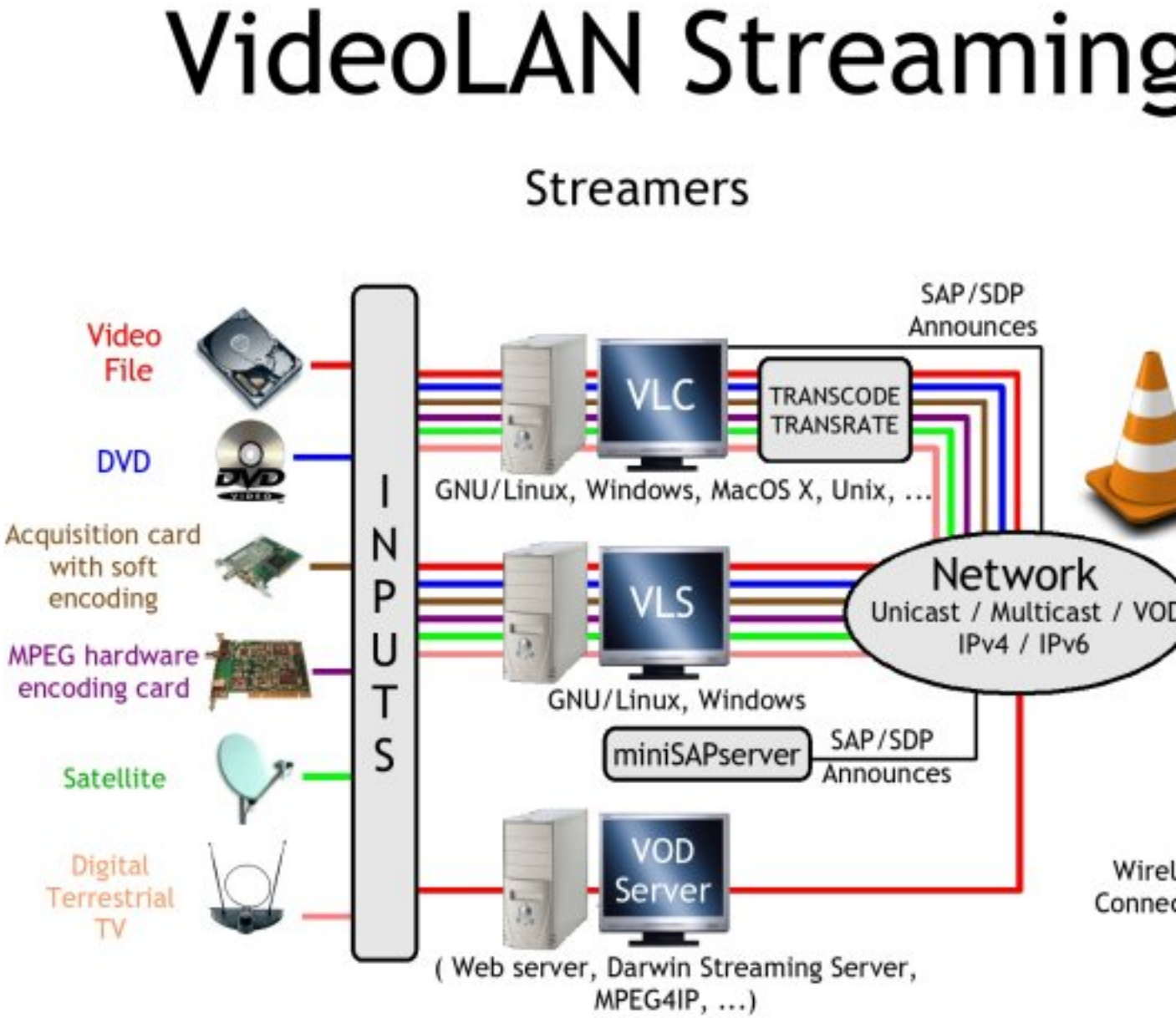
Overview

VideoLAN is a complete software solution for video streaming, developed by students of the Ecole Centrale Paris (http://www.ecp.fr) and developers from all over the world, under the GNU General Public License (http://www.gnu.org/copyleft/gpl.html) (GPL). VideoLAN is designed to stream MPEG videos on high bandwidth networks.

The VideoLAN solution includes:

- VLS (VideoLAN Server), which can stream MPEG-1, MPEG-2 and MPEG-4 files, DVDs, digital satellite channels, digital terrestial television channels and live videos on the network in unicast or multicast
- VLC (initially VideoLAN Client), which can be used as a server to stream MPEG-1, MPEG-2 and MPEG-4 files, DVDs and live videos on the network in unicast or multicast ; or used as a client to receive, decode and display MPEG streams under multiple operating systems

Here is an illustration of the complete VideoLAN solution:



Global VideoLAN solution

More details about the project can be found on the VideoLAN Web site (http://www.videolan.org).

VideoLAN software

VLC Media Player

VLC works on many platforms: Linux, Windows, Mac OS X, BeOS, *BSD, Solaris, Familiar Linux, Yopy/Linupy and QNX. It can read:

- MPEG-1, MPEG-2 and MPEG-4 / DivX files from a hard disk, a CD-ROM drive, ...
- DVDs and VCDs
- from a satellite card (DVB-S)
- from a camcorder (DV)
- MPEG-1, MPEG-2 and MPEG-4 streams from the network sent by VLS or VLC's stream output

VLC can also be used as a server to stream:

- MPEG-1, MPEG-2 and MPEG-4 / DivX files,
- DVDs,
- from an MPEG encoding card,
- from a camcorder DV,

to:

- one machine (i.e. to one IP address): this is called *unicast*,
- a dynamic group of machines that the clients can join or leave (i.e. to a multicast IP address): this is called *multicast*,

in IPv4 or IPv6.

To get the complete list of VLC's possibilities on each plateform supported, see the VLC features page (http://www.videolan.org/vlc/features.html).

Note: VLC doesn't work on Mac OS 9, and will probably never do.

Mini-SAP-server

You can add a channel information service based on the SAP/SDP standard to the VideoLAN solution. The mini-SAP-server sends announces about the multicast programs on the network in IPv4 or IPv6, and VLCs receive these annouces and automatically add the programs announced to their playlist.

The mini-SAP-server works under Linux and Mac OS X.

Muxers and codecs

What is a codec ?

To fully understand the VideoLAN solution, you must understand the difference between a *codec* and a *container format*

A *codec* is a compression algorithm, used to reduce the size of a stream. There are audio codecs and video codecs. MPEG-1, MPEG-2, MPEG-4, Vorbis, DivX, ... are codecs

What is a container format ?

To start off, think of a *container format* as a standard shipping box. You get a box in the mail and you think, "Cool! What's inside." You don't really care about the box itself, you care about what's in that box. The problem? You can't see into the box. So what do you do? You get a knife and cut it open.

A *container format* follows this same basic idea. It contains one or several streams already encoded by codecs. Very often, there is an audio stream and a video one. AVI, Ogg, MOV, ASF, MP4 ... are container formats. The streams contained can be encoded using different codecs. In a perfect world, you could put any codec in any container format. Unfortunately, there are some incompatibilities. You can find a matrix of possible codecs and container formats on the features page (http://www.videolan.org/streaming/features.html)

Encoding a video

This is the first step where you are going to create the shipping box.

First you need to encode your file. That means that a file, wheter it is an audio, video file, is compressed to another format that normally takes up less physical drive space than the previous format. Common video encoding methods are DivX, MPEG-1, MPEG-2, MPEG-4 ... most common audio encoding method is MP3 or ogg-vorbis.

Then you have to mux (or multiplex). This means basically a process where separate parts of the video (or streams) are joined together into one file.

Playing a video

Now that you have your box, you need to open it before to see the content. That's exactly what VLC will do. To decode a stream, VLC first *demuxes* it. This means that it reads the container format and separates audio, video, and subtitles, if any. Demuxing files doesn't weaken the video nor audio quality, it doesn't do anything for these data streams, it justs simply saves them into separate files, each containing one element of the original file. Then, each of these are passed *decoders* that do the mathematical processing to decompress the streams.

There is a particular thing about MPEG:

- MPEG is a codec. There are several versions of it, called MPEG-1, MPEG-2, MPEG-4, ...
- MPEG is also a container format, sometimes refered to as MPEG System. There are several types of MPEG: ES, PS, and TS.

  When you play an MPEG video from a DVD, for instance, the MPEG stream is actually composed of several streams (called Elementary Streams, ES): there is one stream for video, one for audio, another for subtitles, and so on. These different streams are mixed together into a single Program Stream (PS). So, the .VOB files you can find in a DVD are actually MPEG-PS files. But this PS format is not adapted for streaming video through a network or by satellite, for instance. So, another format called Transport Stream (TS) was designed for streaming MPEG videos through such channels.

Chapter 2. Easy streaming

Intro

The easier way to start streaming with VLC is by using one of the graphical user interfaces: wxwindows for Windows and GNU/Linux, the skinnable Windows and GNU/Linux interface or the MacOS X native interface.

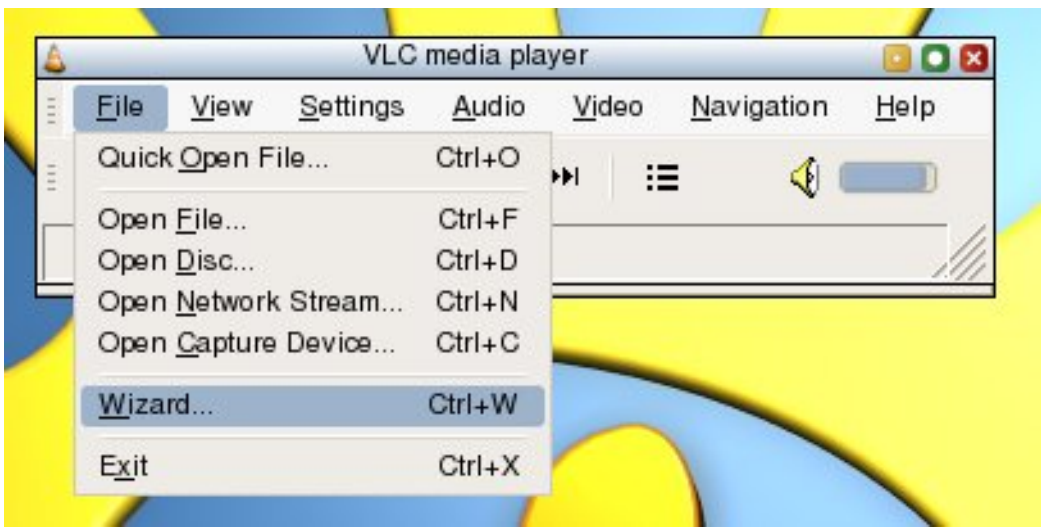
Streaming using the Wizard

The *Streaming/Transcoding Wizard* leads you step by step through the process of streaming your media on a network or saving it to your hard drive. This *Wizard* offers easy to use menus but provides a restricted set of options.

Note: The wizard is only available on the wxWindows interface.

Launching the wizard

To launch the *Streaming/Transcoding Wizard*, open the "File" menu, and select the Wizard menu item.

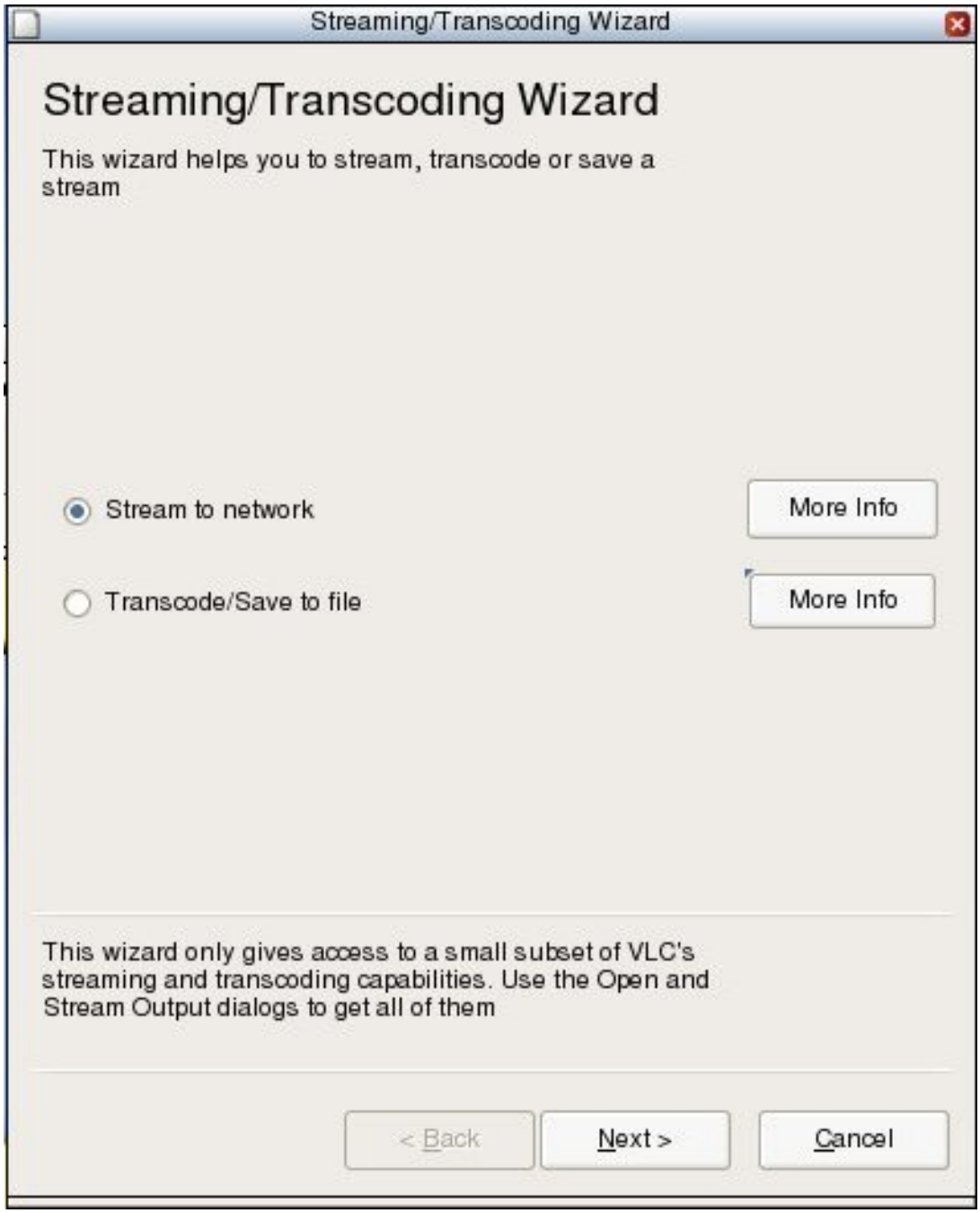


Launching the wizard

Wizard dialog

First select the type of task:

- *Stream to network*: Choose this option if you want to stream media on network.
- *Transcode/Save to file*: Choose this option if you want to change a file's audio codec and/or video codec, its bitrate, and/or encapsulation method.

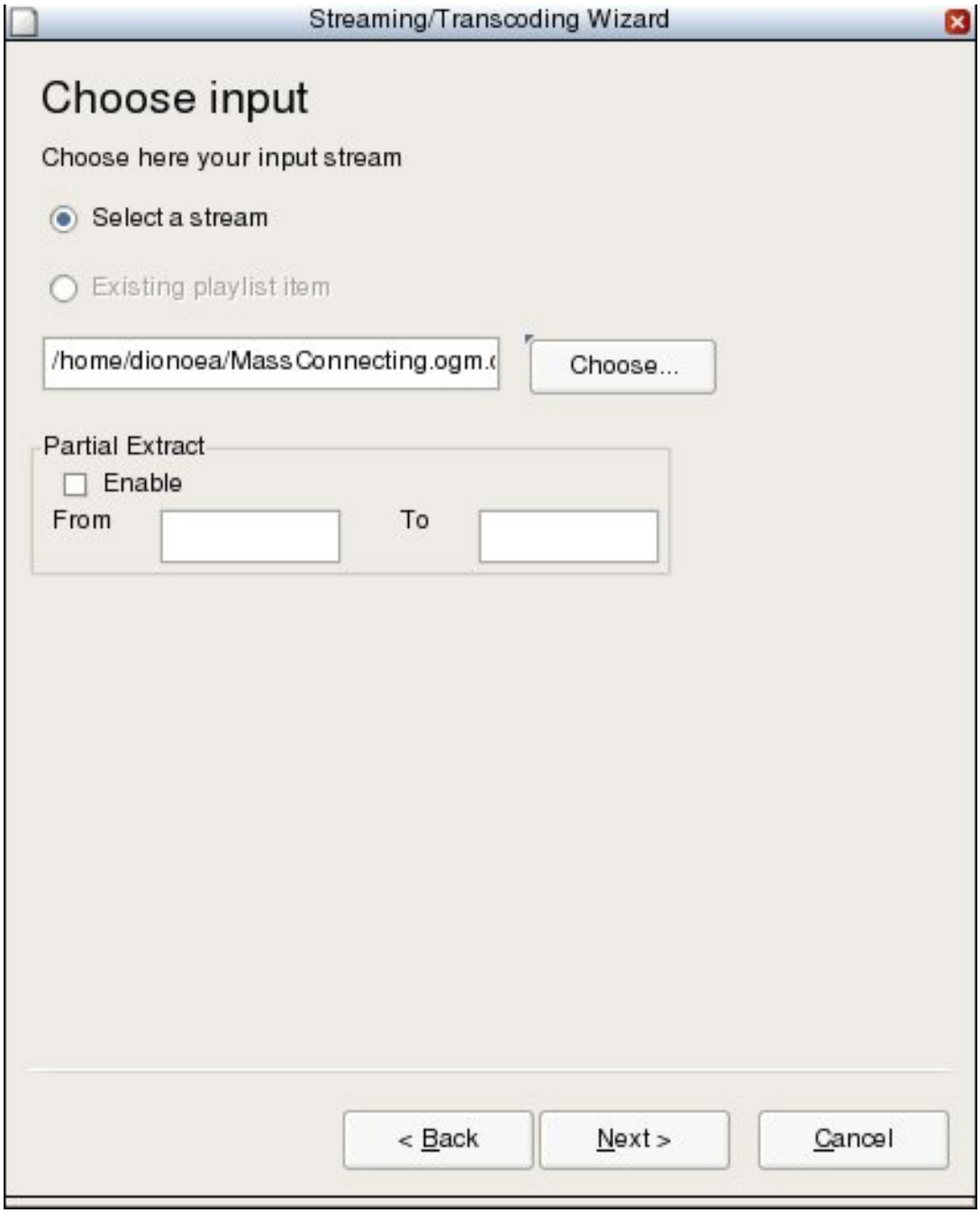


The Wizard Dialog

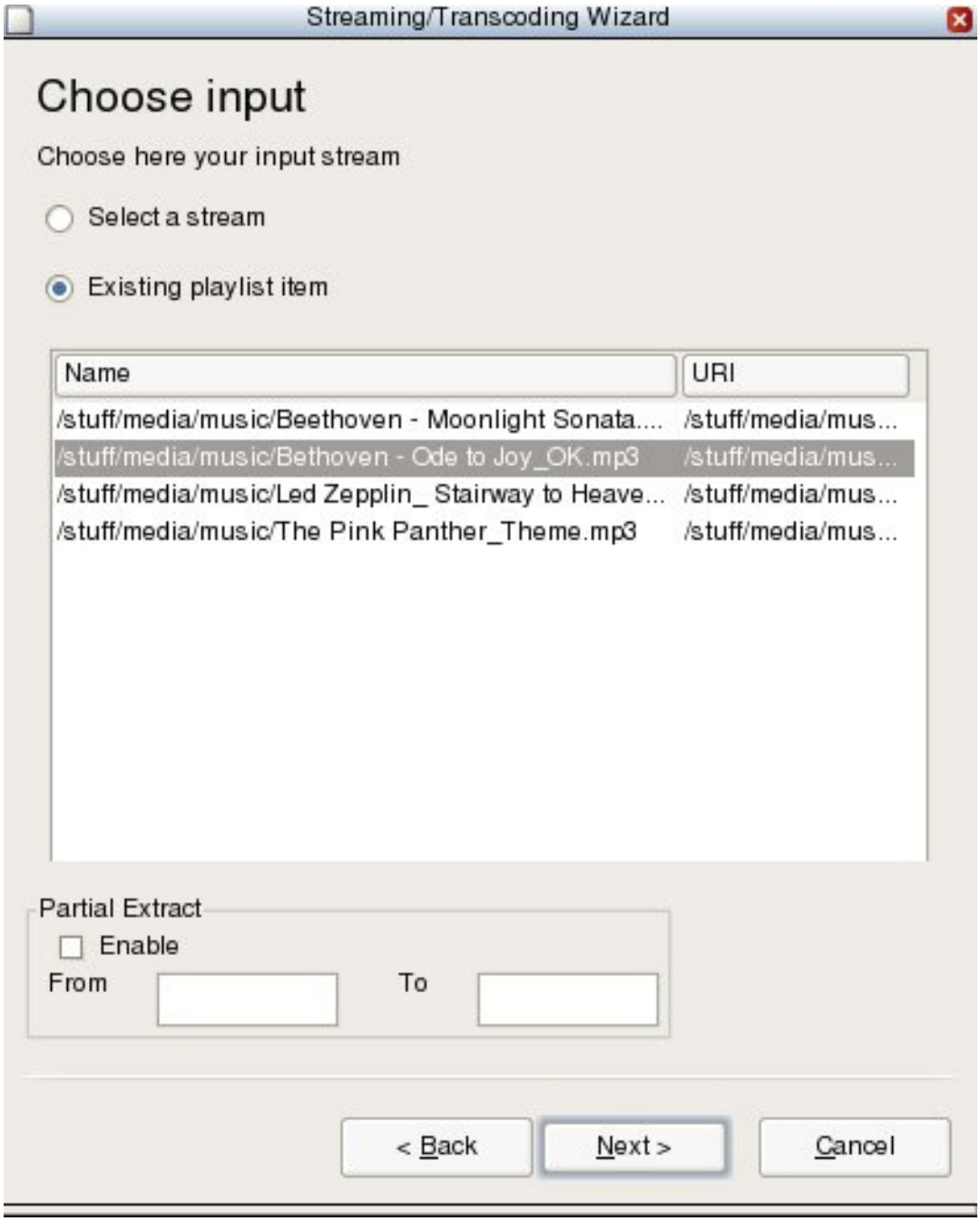
Input selection

Select a stream (such as a file, a network stream, a disk, a capture device ...) by selecting the *Choose...* dialog or an existing item in your playlist, using the *Existing playlist item* option.

Partial Extract: To read only part of the stream, check the "Enable" checkbox and choose a start and end date (in seconds). This option should only be used with streams you can control such as files or discs but not network streams or capture devices.



Wizard input selection



Wizard input selection from playlist

Streaming methods

If you chose *Stream to network* option, you can now specify the streaming method. Available methods are:

- *UDP Unicast*: Stream to a single computer. Enter the client's IP address (in the 0.0.0.0 - 223.255.255.255 range).
- *UDP Multicast*: Stream to multiple computers using multicast. Enter the IP address of the multicast group (in the 224.0.0.0 to 239.255.255.255 range).
- *HTTP*: Stream by using the HTTP protocol. If you leave the *Destination* text box empty, VLC will listen on all the network interfaces of the server on port 8080. Specify an address, port and path on which to listen using the following syntax [ip][:port][/path].

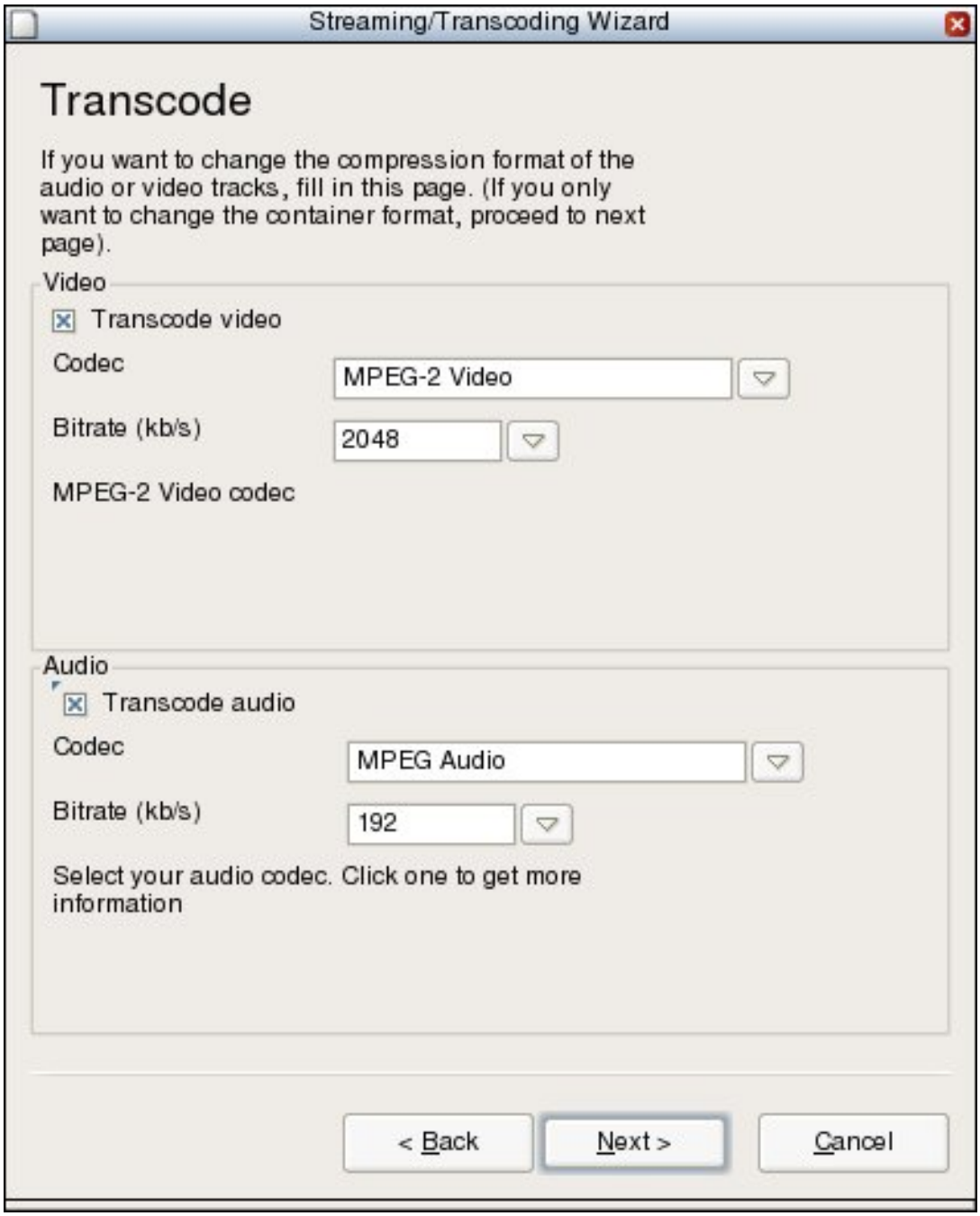
For instance, *192.168.0.1:80/stream* will make VLC listen on the interface carrying the 192.168.0.1 IP address, on the 80 TCP port, in the /stream *virtual file*.

Wizard streaming method

Transcoding options

If you chose the *Transcode/Save to file* option, you can now specify the new audio and video codecs and bitrates you want you input converted to.

(See *Streaming, Muxers and Codecs*)

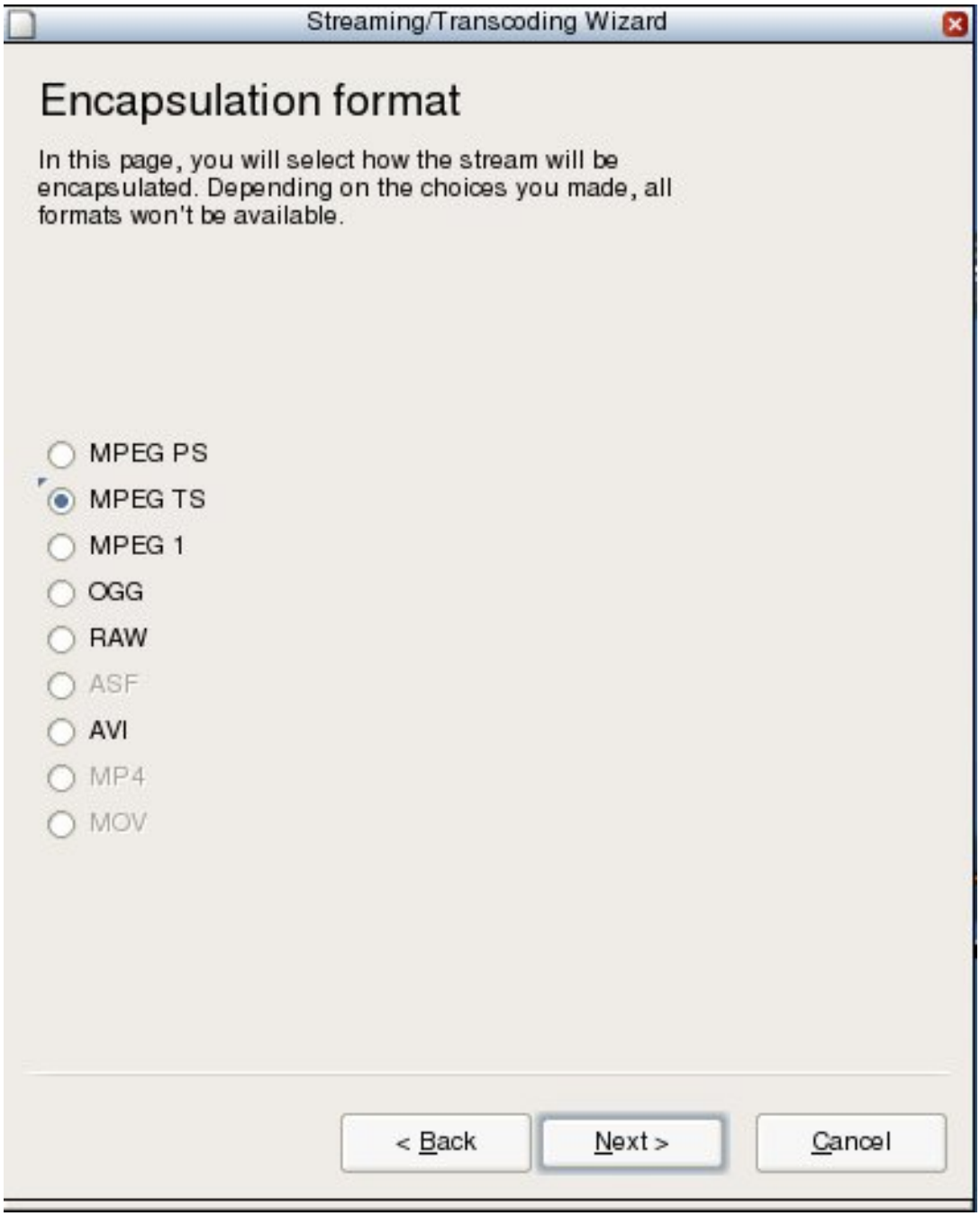


Wizard transcode

Encapsulation method

Choose the method format. The UDP streaming methods require MPEG TS encapsulation. The HTTP streaming method can be used with the MPEG PS, MPEG TS, MPEG 1, OGG, RAW or ASF encapsulation. Saving to a file can be done using any encapsulation format compatible with the chosen codecs.

(See *Streaming, Muxers and Codecs*)



Wizard encapsulation method

Streaming options

If you chose to *Stream to network* you can now specify several options.

- *Time To Live (TTL)* This sets the numbers of routers your stream can go through, for UDP unicast and unicast access methods. If you do not know what this means, you should leave the default value.

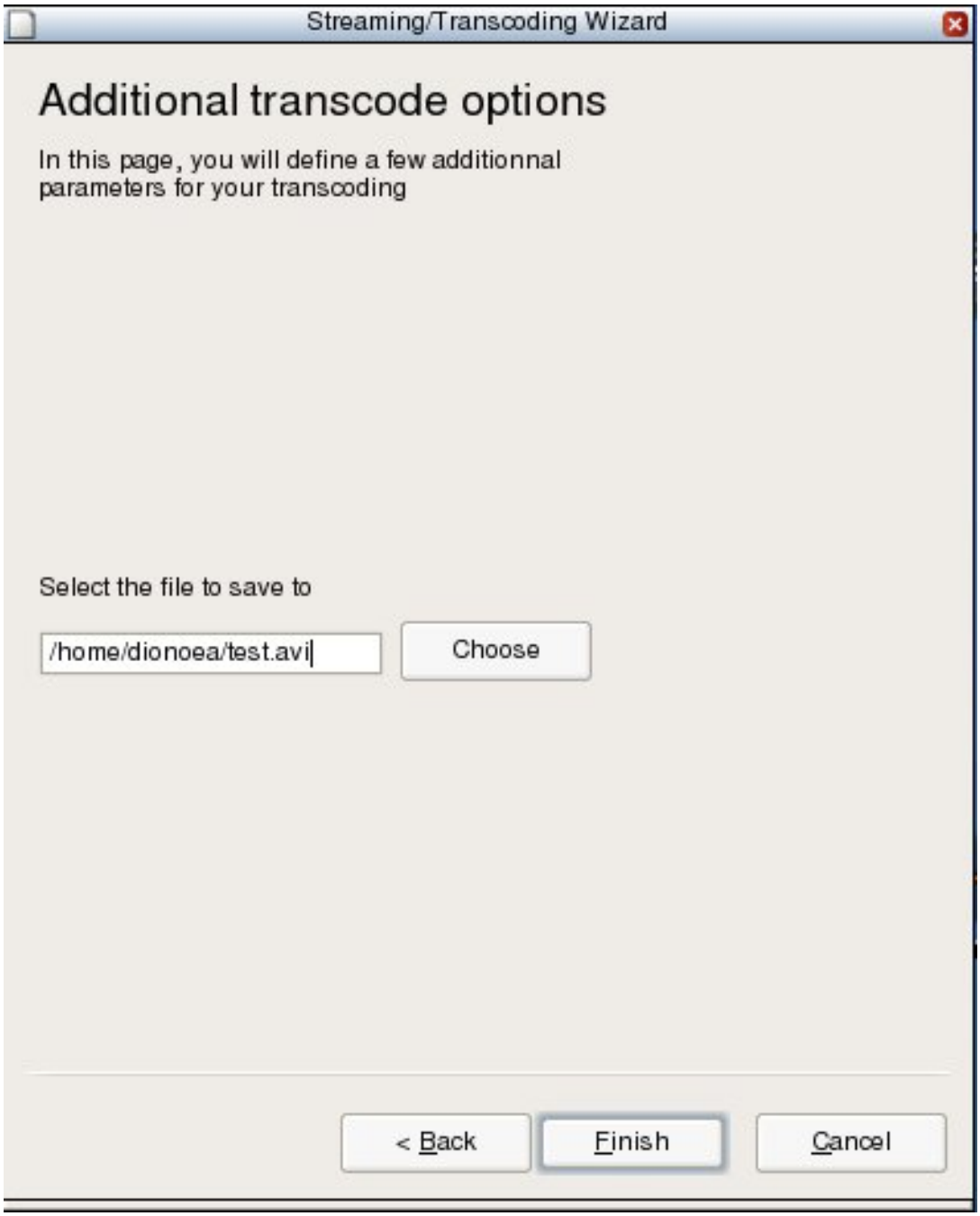
  Note: With UDP multicast, the default TTL is set to 1, meaning that your stream won't get accross any router. You may want to increase it if you want to route your multicast stream.

- *SAP Announce* To advertise your stream over the network when using the UDP streaming method, using the SAP protocol, enter the name of the stream in the text input and check the checkbox. This is NOT available for the HTTP streaming method.

Wizard streaming options

Save to file destination

If you chose *Transcode/Save to file* you can now specify the file you want to save the stream to.



Wizard save file - wxWindows interface

You can now select the *Finish* button to start streaming/converting the source.

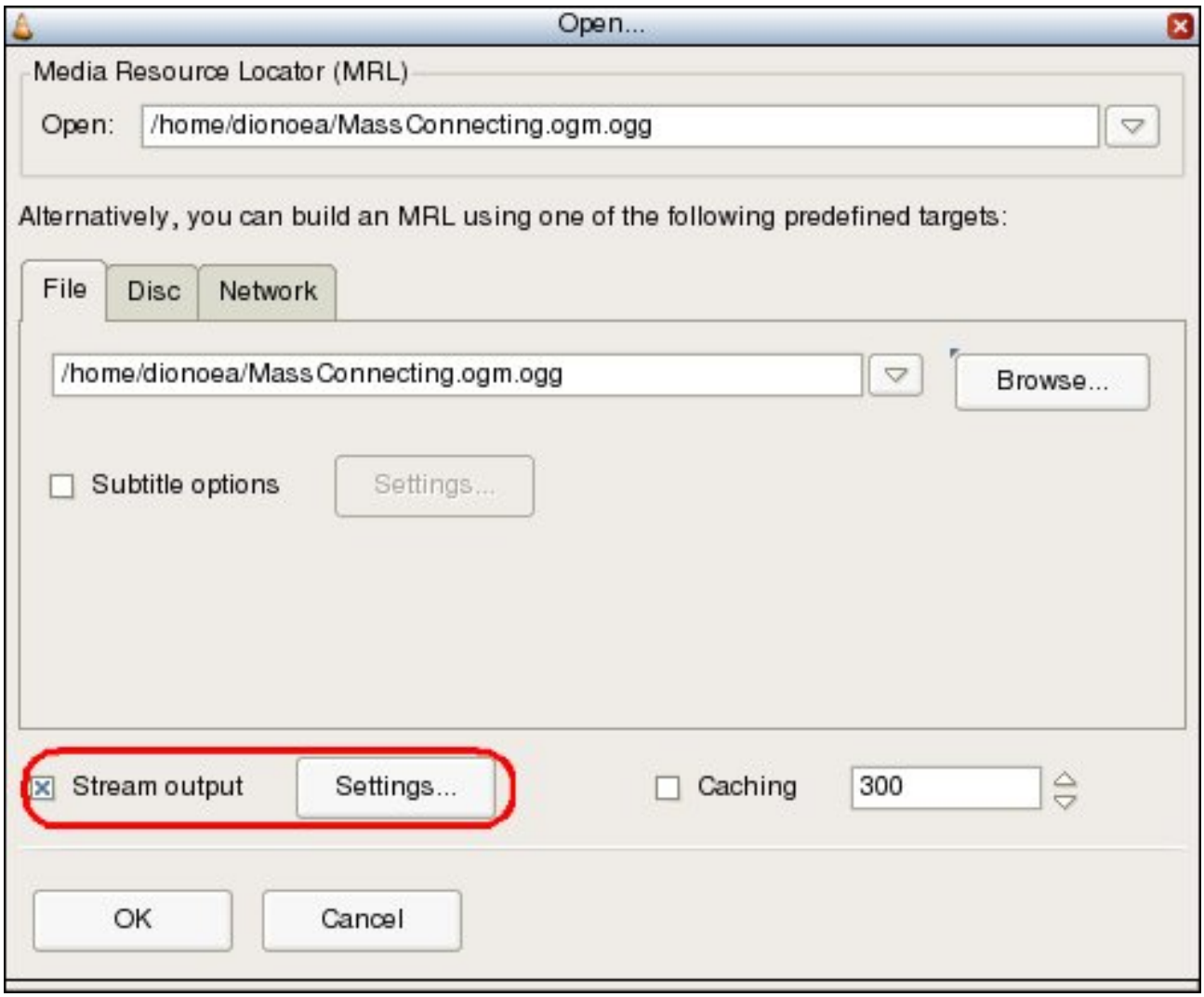
Streaming using the GUI

Introduction

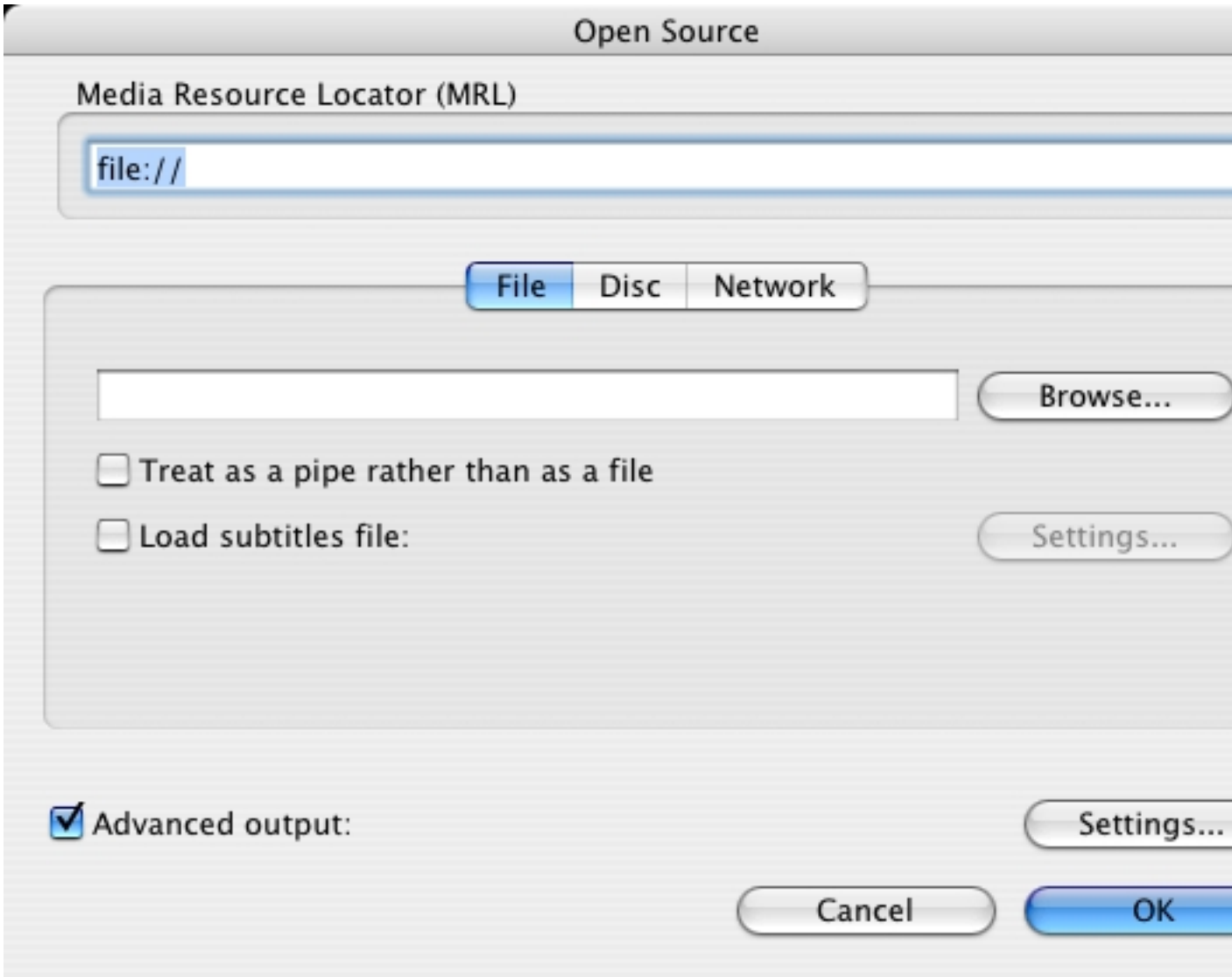
A second way to set up a streaming instance using VLC is using *Stream Output* panel in the *Open...* dialog of the wxWindows (Windows / GNU Linux), skinnable (Windows / GNU Linux) and MacOS X interfaces. Streaming methods

and options used 99% of time should be available in this panel.

To stream the opened media, check the "Stream output" checkbox in the "Open File/Disc/Network Stream/Capture Device" dialog and click on the "Settings" button.

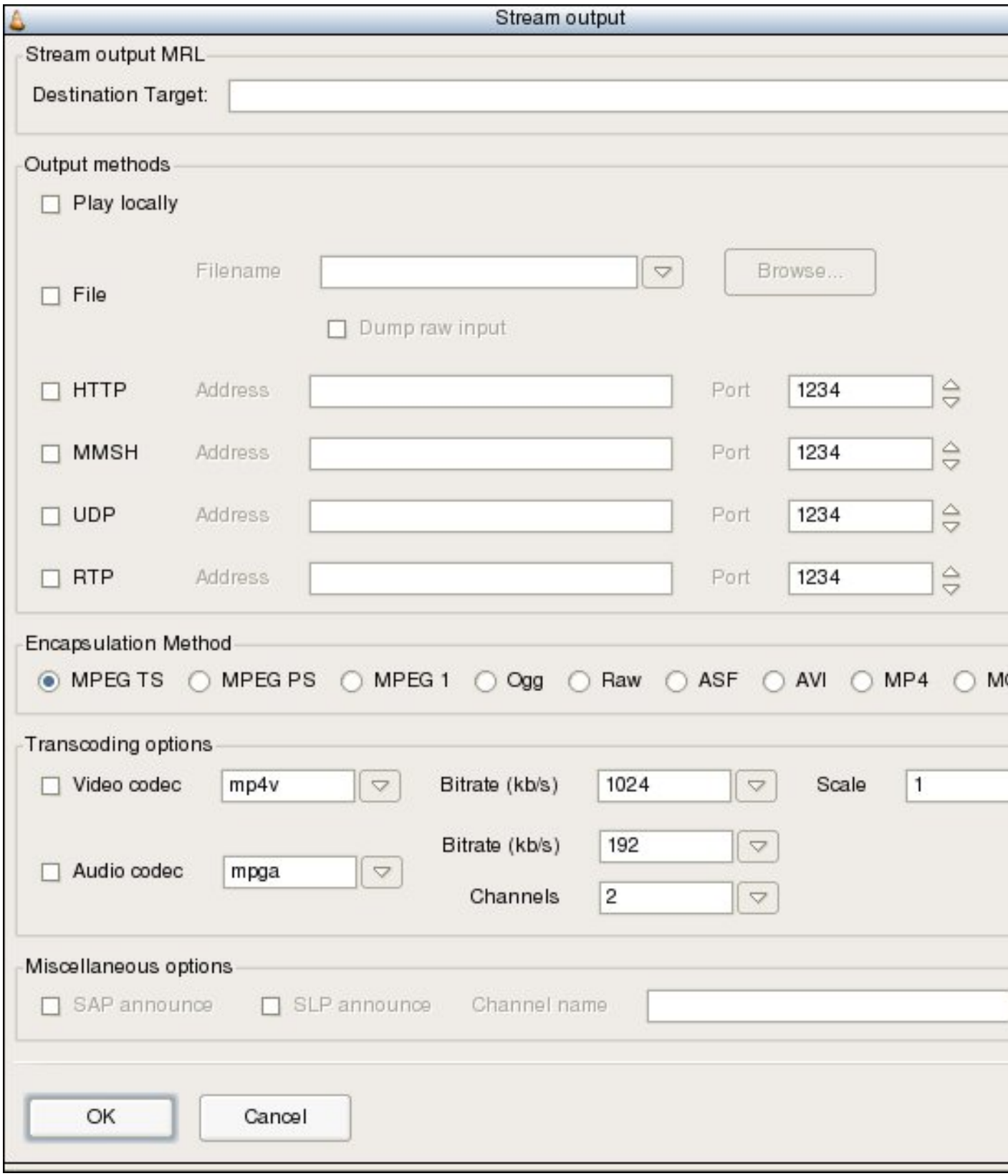


Open file dialog - wxWindows interface

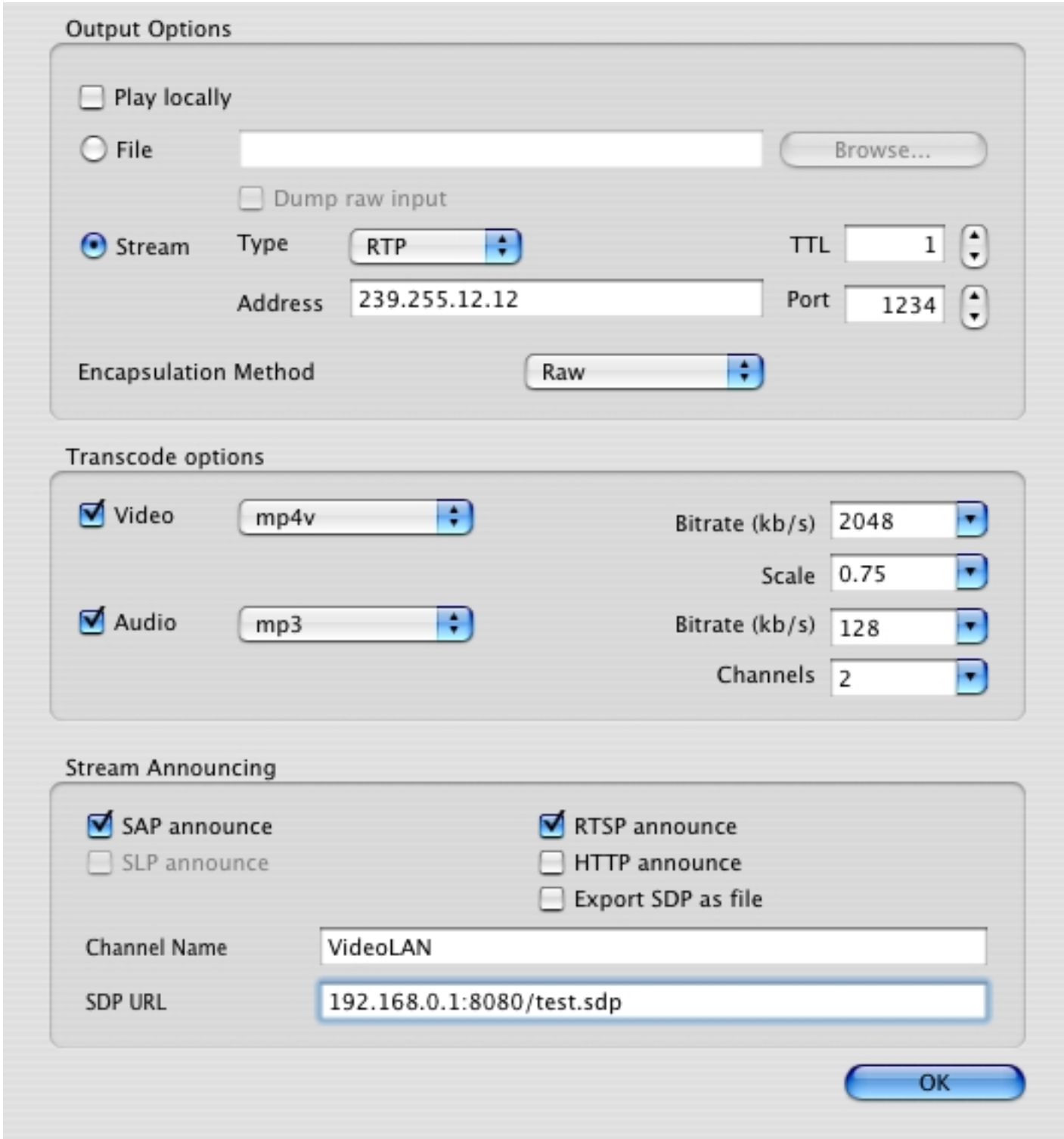


Open file dialog - Mac OS X interface

The Stream Output dialog



Stream output dialog - wxWindows interface



Stream output dialog - wxWindows interface

Stream Output MRL

On the wxWindows interface, a text box displays the *Stream Output MRL* (Media Ressource Locator). This is updated as you change options in the Stream output dialog. For more information on how to edit the *Stream Output MRL* read *Advanced streaming using the command line*.

Output methods

- *Play localy*: display the stream on your screen. This allows to display the stream you are actually streaming. Effects of transcoding, rescaling, etc... can be monitored locally using this function.
- *File*: Save the stream to a file. The *Dump raw input* option allows to save the input stream as it read by VLC, without any processing.
- *HTTP*: Use the HTTP streaming method. Specify the IP address and TCP port number on which to listen.
- *MMSH*: This access method allows to stream to Microsoft Windows Media Player. Specify the IP address and TCP port number on which to listen.

  Note: This will only work with the *ASF* enacpsulation method.

- *UDP*: Stream in unicast by providing an address in the 0.0.0.0 - 223.255.255.255 range or in multicast by providing an address in the 224.0.0.0 - 239.255.255.255 range. It is also possible to stream to IPv6 addresses.

  Note: This will only work with the *TS* encapsulation method.

- *RTP*: Use the Real-Time Transfer Protocol. Like UDP, it can use both unicast and multicast addresses.

  Note: UDP, HTTP, MMSH and RTP methods require to select the *Stream* option on the MacOS X interface

(See *Streaming, Muxers and Codecs*)

Encapsulation method

Select an encapsulation method that fits the codecs and access method of your stream, among MPEG TS, MPEG PS, MPEG 1, OGG, Raw, ASF, AVI, MP4 and MOV. (See *Streaming, Muxers and Codecs*)

Transcoding options

Enable video transcoding by checking the "Video Codec" checkbox. Choose a codec from the list. You can also specify an average bitrate and scale the input. (See *Streaming, Muxers and Codecs*)

Enable audio transcoding by checking the "Audio Codec" checkbox. Choose a codec from the list. You can also specify an average bitrate and the number of audio channels to encode. (See *Streaming, Muxers and Codecs*)

Miscellaneous options

Select methods to announce your stream. You can use SAP (Service Announce Protocol) or SLP (Service Location Protocol). You must also specify a channel name. The Mac OS X interface also allows you to export the description (SDP) file of a RTP session using the internal HTTP or RTSP server of VLC, or as a file. This can be done using the according checkboxes. The *SDP URL* text box allows to give the url or destination where the SDP file will be available.

Chapter 3. Advanced streaming using the command line

Structure of stream output

Stream output is the name of the feature of VLC that allows to output any stream read by VLC to a file or as a network stream instead of displaying it. Different kind of processing can be applied to the stream during this process (transcoding, re-scaling, filters, re-muxing...) Stream output includes different modules, each of them having different capabilities. You can *chain* modules to enhance the possibilities.

Here is the list of the modules currently available:

- *standard* allows to *send* the stream via an *access output* module: for example, UDP, file, HTTP, ... You will probably want to use this module at the end of your chains.
- *transcode* is used to transcode (decode and re-encode the stream using a different codec and/or bitrate) the audio and the video of the input stream. If the input or output access method doesn't allow pace control (network, capture devices), this done "on the fly", in real time. This can require quite a lot of CPU power, depending on the parameters set. Other streams, such as files and disks are transcoded as fast as the system allows it.
- *duplicate* allows you to create a second chain, where the stream will be handled in an independent way.
- *display* allows you to display the input stream, as VLC would normally do. Used with the *duplicate* module, this allows you to monitor the stream while processing it.
- *rtp* streams over RTP (one UDP port for each elementary stream). This module also allows RTSP support.
- *es* allows you to make separate Elementary Streams (ES) out of an input stream. This can be used to save audio and video streams to separate files, for instance.

Each of these modules may take options. Here is the syntax that you must use:

```
% vlc input_stream --sout "#module1{option1=parameter1{parameter-option1},option2=parameter2}:module2{option1
```

> **Note:** Some of the module options (option1 in the example) have to be set, others are optional. Option parameters (parameter-option1 in the example) are always optional. These option parameters are also often very advanced settings. If you don't understand their description, this certainly means that you don't need them.

You may also use the following syntax :

```
% vlc input_stream --sout-module1-option1=... --sout-module1-option2=... --sout-module2-option1=... --sout-mo
```

For example, to transcode a stream and send it, use:

```
% vlc input_stream --sout '#transcode{options}:standard{options}'
```

Description of the modules

standard (alias std)

This module saves the stream to a file or sends it over a network, after having muxed it.

The available options are:

access=

This option allows to set the medium used to save or send the stream. This is a compulsory option. Available options are:

- file: saves the stream to a file.

  Use the *append* option to append the stream to an existing file instead of replacing it.

- udp: streams to a UDP unicast or multicast address.

  Item options are: *caching=<time in ms>* to set the time VLC should buffer data before sending it, *ttl=<ttl>* to set the ttl of the sent udp packets, *group=<amount of packets>* to sent packets by burst instead of one by one, *late=<time in ms>* to drop packets that arrive too late at this stage of the chain, *raw* if you don't want to wait until the MTU is filled before sending the packet.

- http: streams over HTTP.

  Item options are: *user=<user name>* to enable HTTP basic authentication and set the user, *pwd=<password>* to set the basic authentication password, *mime=<mime type>* to set the mime type returned by the server.

- https: streams over HTTP, using a secured SSL connection.

  Item options are the same as for http and: *cert=<path to certificate>*to set the certificate to use, *key=<path to key>* to set the private key file the server should use for the SSL connection, *ca=<path to certificate>* to set the path to the root CA certificates to use for SSL, *crl=<path to certificate>* to set the revocation certificate to use for the SSL connection.

- mmsh: streams using the Microsoft MMS protocol. This protocol is used as transport method by many Microsoft's softwares. Note that only a small part of the MMS protocol is supported (MMS encapsulated in HTTP).

  Item options are the same as for the http module.

- rtp: streams over RTP This can only be used to stream MPEG-TS over plain RTP. Support for this option has been removed in VLC 0.9.0 and latter. You should use the *rtp* stream output module instead. Options are the same as for the *udp* setting.

mux=

This option allows you to set the encapsulation method used for the resulting stream. This option has to be set.

Available options are:

- *ts:* the MPEG2/TS muxer. This the standard muxer used to stream MPEG 2. This muxer can be used with any *access* method. Supported codecs are MPEG 1/2/4, MJPEG, H263, H264, I263, WMV 1/2 and theora for video, MPEG audio, AAC and a52 for the audio stream.

  Item options are: *pid-video=<pid>* to set the PID of the video track, *pid-audio=<pid>* to set the PID of the audio track, *pid-spu=<pid>* to set the PID of the subtitle track, *pid-pmt=<pid>* to set the PID of the PMT (Program Map Table), *tsid=<id>* to set the ID of the resulting TS stream, *shaping=<shaping delay in ms>* to set the minimum interval during which the bitrate of the stream will remain constant, for variable bitrate streams, *use-key-frames* uses I frames as limits for the shaping intervals, *pcr=<PCR interval in ms>* allows to set at which interval Program Clock References will be sent, *dts-delay=<delay in ms>* allows to delay PTS (Presentation Time Stamps) from the DTS (Decoding Time Stamp) from the given time, *crypt-audio* allows to enable encryption of the audio track using the CSA algorithm, *csa-ck=<key as a 16 character word>* allows to set the key used for CSA encryption.

- *ps:* the MPEG2/PS muxer. This the standard muxer for MPEG 2 files(.mpg). It can be used with the file and http output methods. Supported codecs are MPEG 1/2 and MJPEG for video, MPEG audio and a52 for audio streams.

The only available item option is *dst-delay=<delay in ms>*. It allows to delay PTS (Presentation Time Stamps) from the DTS (Decoding Time Stamp) from the given time.

- *mpeg1:* the standard MPEG 1 muxer. This muxer should be used instead of ps with MPEG 1 video streams, when saved to a file or streamed over HTTP. Supported codecs are MPEG 1 and MPEG audio.

  Items options are the same as for the PS muxer.

- *ogg:* the ogg muxer. This is the muxer from the Xiph project. It can be used with the HTTP and file output methods. Supported codecs are MPEG 1/2/4, MJPEG WMV 1/2 and Theora, audio streams can be vorbis, flac, speex, a52 or MPEG audio.

  There is no item option for this muxer.

- *asf:* the Microsoft ASF muxer. This is the standard muxer used for streaming by Microsoft's softwares. Is also used as container for WMA audio files. This muxer can be used with the file and HTTP output methods. Supported codecs are MPEG 4, MJPEG, WMV 1/2 for video, MPEG audio, a52 for audio streams.

  Item options are: *title=<title>*, *autor=<author>*, *copyright=<copyright message>*, *comment=<comments>*, *rating=<rating>* allow you to set what will be displayed in the according field of the stream comments.

- *asfh:* this is a special version of the ASF muxer, that should be used for MMSH streaming. MMSH is the only supported output method. Supported codecs are the same as for ASF.

  Item options are the same as for ASF.

- *avi:* the Microsoft AVI muxer. This is very common encapsulation format for MPEG 4 files. The only supported output method is file. Supported codecs are MPEG 1/2/4, H263, H264 and I263 for video, MPEG audio and a52 for audio streams.

  There is not item option for this muxer.

- *mpjpeg:* the multipart jpeg muxer. This encapsulation format is mostly used on surveillance video cameras with an integrated web-server. Such streams are usually embedded in web-pages and seen with standard Internet browsers, as they are seen as a succession of jpeg images. The only supported output method is HTTP. The only usable codec is MJPEG. No sound track can be muxed in such streams.

  No item option is available for this muxer.

dst=

This option allows to give informations about the location where the stream should actually be saved or sent.

Here is the meaning of the *dst* option depending on the parameter used for the *access* option.

- If the *file* output method is used, dst is the complete path where the file should be saved.
- If the *udp* or *rtp* output method is used, dst is the unicast or multicast destination address and, optionally, UDP port, in the form *address:port*.
- If the *http*, *https* or *mmsh* output method is chosen, dst is the address, port and path of the local network interface on which the server should listen for requests. If no address is given, VLC will listen on all the network interfaces. These information have to be supplied using the *address:port/path* syntax.

sap

Use this option if you want VLC to send SAP (Session Announcement Protocol) announces. SAP is a service discovery protocol, that uses a special multicast address to send a list of available streams on a server.

Note: This option can only be enabled with the *udp* output method.

group=

This option allows to specify the name of an optional *group* of streams. A VLC used as a client will use this field to classify the stream.

> **Note:** This option uses a private extension of the SAP protocol. VLC will be the only client able to read this field.
>
> This option can only be used it the *sap* option has been enabled.

sap-ipv6

Use this option if you want the SAP announces to be sent using the *IPv6* protocol instead of *IPv4*.

> **Note:** This option can only be used it the *sap* option has been enabled.

slp

SLP stands for *Service Location Protocol*. It is an alternative to SAP for session announcement. Use this option if you want to send such announces.

name=

Use this option to specify the name of the stream that will be sent in SAP and SLP announces.

> **Note:** This option can only be used it the *sap* or *slp* option has been enabled.

display

This module can be used to display the stream. This is particularly useful in a *duplicate* chain, in order to monitor a stream while it is being saved or streamed.

available options are:

novideo

You can use this option to disable video in the displayed stream.

noaudio

You can use this option to disable audio in the displayed stream.

delay=

You can use this option to introduce a delay in the display of the stream. Delay has to be given in ms.

rtp

This module can be used to send a stream using the *RTP (Real Time Protocol)* protocol (see RFC 3550).

> **Note:** Although use of *RTSP* is possible using this module, it won't allow you to make Video On demand. Please have a look at the description of the VLM module for that.

The different available options are:

dst=

This option allow to give the destination UDP address. This can be the address of a host or a multicast group. This option has to be given, unless the *sdp=rtsp://*option is given (see below). In the latter case, the stream will be sent to the host doing the *RTSP* request.

port=

This option allows to set the UDP port used to send the first *elementary stream*. This port has to be even. Other streams will be streamed using even ports directly above this one.

port-video=

This option allows to set the UDP port used to send the first video *elementary stream*. This port has to be even.

port-audio=

This option allows to set the UDP port used to send the first audio *elementary stream*. This port has to be even.

sdp=

This option allows to set the way the SDP (Session Description Protocol) file corresponding the the stream should be made available.

Options are:

- *file://<path to the file>*, to export the SDP as a local file
- *http://<local interface IP:port/path>*, to make the file available using the integrated HTTP server of VLC.

> **Note:** The *local interface IP* argument is optional. If not given, VLC will listen on all available interfaces.

- *rtsp://<local interface IP:port/path>*, to make the SDP file available using the *RTSP* protocol (see RFC 2326).

Note: The *local interface IP* argument is optional. If not given, VLC will listen on all available interfaces.

- *sap*, to export the SDP using the SAP (Session Announcement Protocol, see RFC 2974).

ttl=

This option can be used to set the *TTL* (Time to Live) of the sent UDP packets.

mux=

This option allows to set the encapsulation method used to send the stream. See *mux=* options of the *standard* module for a description of the available method. Only *ts* is possible for RTP streams. By default, each elementary stream is sent as a separate RTP media, i.e. no encapsulation is done.

rtcp-mux

This options enables RTP/RTCP multiplexing (see draft-ietf-avt-rtp-and-rtcp-mux), i.e. sends and receives RTCP packets on the same port numbers as RTP packets. By default, RTCP packets are sent and received on the next port.

proto=

This selects the transport protocol to carry RTP packets. Possible values include:

- *dccp*: accept incoming DCCP connections at the specified IP address (dst=),
- *sctp*: accept SCTP connections at the specified IP address (dst=), not implemented yet,
- *tcp*: accept TCP connections at the specified IP address (dst=) and use RFC4571 RTP framing, not implemented yet,
- *udp*: send UDP packets to the specified destination (either unicast or multicast) - this is the default value,
- *udplite*: send UDP-Lite packets to the specified destination (either unicast or multicast).

name=

This option can be used to set the name that will be displayed on the client receiving the stream.

description=

This option can be used to give an additional description of the stream.

url=

This option allows to give the address of a website with additional informations about the stream.

email=

This options allows to give a contact e-mail address

es

The *es* module can be used to separate the different *elementary streams* from a stream, and save each of them in a different file or send it to a separate destination.

The available parameters are:

access-video=

Use this option to set the medium used to save or send the video *elementary streams*. Possible values and item options are the same as for the *access* option of the *standard* module (see above).

access-audio=

Use this option to set the medium used to save or send the audio *elementary streams*. Possible values and item options are the same than for the *access* option of the *standard* module (see above).

access=

This option can be used instead of both *access-video* and *access-audio* options, when they share the same setting.

mux-video=

Use this option to set the encapsulation method used for the video *elementary streams*. Possible values and item options are the same as for the *mux* option of the *standard* module (see above).

mux-audio=

Use this option to set the encapsulation method used for the audio *elementary streams*. Possible values and item options are the same than for the *mux* option of the *standard* module (see above).

mux=

This option can be used instead of both *mux-video* and *mux-audio* options, when they share the same setting.

dst-video=

Use this option to set the location where the video *elementary streams* should be saved, sent, or made available. The exact meaning of this option depends on the value of the *access-video* option and is the same as for the *url* option of the *standard* module (see above).

> **Note:** If you use the *%d* string in the url field, VLC will replace it by the number of the audio or video track considered. The *%c* string will be replaced by the name (FOURCC) of the codec of the track.

dst-audio=

Use this option to set the location where the audio *elementary streams* should be saved, sent, or made available. The exact meaning of this option depends on the value of the *access-audio* option and is the same as for the *url* option of the *standard* module (see above).

> **Note:** If you use the *%d* string in the url field, VLC will replace it by the number of the audio or video track considered. The *%c* string will be replaced by the name (FOURCC) of the codec of the track.

dst=

This option can be used instead of both *dst-video* and *dst-audio* options, when they share the same setting.

transcode

You can use this module to transcode a stream, i.e. to change its codecs or the encoding bitrates. Some additional processing can be done during this process, such as re-scaling, deinterlacing, resampling, etc.

> **Note:** Depending on the bitrate of the original stream and of the options chosen, transcoding can be a very CPU intensive task. As a consequence, streaming of a real time transcoded stream can lead to dropped frames or a jerky image and sound in some cases, when running out of resources.

Available options are:

vcodec=

This options allows to specify the codec the video tracks of the input stream should be transcoded to.

List of available codecs can be found on the streaming features page (/streaming/features.html).

vb=

This option allows to set the bitrate of the transcoded video stream, in kbit/s

venc=

This allows to set the encoder to use to encode the videos stream. Available options are:

- *ffmpeg*: this is the libavcodec encoding module. It handles a large variety of different codecs (the list can be found on the streaming features page (/streaming/features.html).

  Item options are: *keyint=<number of frames>* allows to set the maximal amount of frames between 2 key frames, *hurry-up* allows the encoder to decrease the quality of the stream if the CPU can't keep up with the encoding rate, *interlace* allows to improve the quality of the encoding of interlaced streams, *noise-reduction=<noise reduction factor>* enables a noise reduction algorithm (will decrease required bitrate at the cost of details in the image), *vt=<bitrate tolerance in kbit/s>* allows to set a tolerance for the bitrate of the outputted video stream, *bframes=<amount of frames>* allows to set the amount of B frames between 2 key frames, *qmin=<quantizer>* allows to set the minimum quantizer scale, *qmax=<quantizer>* allows to set the maximum quantizer scale, *qscale=<quantizer scale>* allows to specify a

fixed quantizer scale for VBR encodings, *i-quant-factor=<quantization factor>* allows to set the quantization factor of I frames, compared to P frames, *hq=<quality>* allows to choose the quality level for the encoding of the motion vectors (arguments are simple, rd or bits, default is simple *FIXME*), *strict=<level of compliance>* allows to force a stricter standard compliance (possible values are -1, 0 and 1, default is 0), *strict-rc* enables a strict rate control algorithm, *rc-buffer-size=<size of the buffer in bits>* allows to choose the size of the buffer used for rate control (bigger means more efficient rate control), *rc-buffer-aggressivity=<float representing the aggressiveness>* allows to set the rate control buffer aggressiveness *FIXME*, *pre-me* allows to enable pre motion estimation, *mpeg4-matrix* enable use of the MPEG4 quantization matrix with MPEG2 streams, improving quality while keeping compatibility with MPEG2 decoders, *trellis* enables trelli quantization (better quality, but slower processing).

- *theora*: The Xiph.org theora encoder. The module is used to produce theora streams. Theora is a free patent and royalties free video codec.

  The only available item option is *quality=<quality level>*. This option allows to create a VBR stream, overriding *vb* setting. the quality level must be an integer between 1 and 10. Higher is better.

- *x264*. x264 is a free open-source h264 encoder. h264 (or MPEG4-AVC) is a quite recent high quality video codec.

  Item options are: *keyint=<number of frames>* allows to set the maximal amount of frames between 2 key frames, *idrint=<number of frames>* allows to set the maximal amount of frames between 2 IDR frames, *bframes=<amount of frames>* allows to set the amount of B frames between an I and a P frame, *qp=<quantizer parameter>* allows to specify a fixed quantizer (between 1 and 51), *qp-max=<quantizer parameter>* allows to set the maximum value for the quantizer, *qp-min=<quantizer parameter>* allows to set the minimum value for the quantizer, *cabac* enables the CABAC (Context-Adaptive Binary Arithmetic Coding) algorithm (slower, but enhances quality), *loopfilter* enables deblocking loop filter, *analyse* enables the analyze mode, *frameref=<amount of frames>* allows to set the number of previous frames used as predictors, *scenecut=<sensibility<* allows to control how aggressively the encoder should insert extra I-frame, on scene change.

fps=

This options allows to set the framerate of the transcoded video, in frame per second. reducing the framerate of a video can help decreasing its bitrate.

deinterlace

This option allows to enable deinterlacing of interlaced video streams before encoding.

croptop=

This option allows to crop the upper part of the source video while transcoding. The argument is the number of lines the video should be cropped.

cropbottom=

This option allows to crop the lower part of the source video. The argument is the Y coordinate of the first line to be cropped.

cropleft

This option allows to crop the left part of the source video while transcoding. The argument is the number of columns the video should be cropped.

cropright=

This option allows to crop the right part of the source video. The argument is the X coordinate of the first column to be cropped.

scale=

This option allows the give the ratio from which the video should be rescaled while being transcoded. This option can be particularly useful to help reduce the bitrate of a stream.

width=

This options allows to give the width of the transcoded video in pixels.

height

This options allows to give the height of the transcoded video, in pixels.

acodec=

This options allows to specify the codec the audio tracks of the input stream should be transcoded to.

List of available codecs can be found on the streaming features page (/streaming/features.html).

ab=

This option allows to set the bitrate of the transcoded audio stream, in kbit/s

aenc=

This allows to set the encoder to use to encode the audio stream. Available options are:

- *ffmpeg*: this is the libavcodec encoding module. It handles a large variety of different codecs (the list can be found on the streaming features page (/streaming/features.html).
- *vorbis*. This module uses the vorbis encoder from the Xiph.org project. Vorbis is a free, open, license-free lossy audio codec.

  Item options are: *quality=<quality level>* allows to use VBR (variable bitrate) encoding instead of the default CBR (constant bitrate), and to set the quality level (between 1 and 10, higher is better), *max-bitrate=<bitrate in kbit/s>* allows to set the maximum bitrate, for vbr encoding, *min-bitrate=<bitrate in kbit/s>* allows to set the minimum bitrate, for vbr encoding, *cbr* allows to force cbr encoding.
- *speex*. This module uses the speex encoder from the Xiph.org project. Speex is a lossy audio codec, best fit for very low bitrates (around 10 kbit/s) and particularly video conference.

samplerate=

This option allows to set the samplerate of the transcoded audio stream, in Hz. Reducing the samplerate is be a way to lower the bitrate of the resulting audio stream.

channels=

This option allows to set the number of channels of the resulting audio stream. This is useful for codecs that don't have support for more than 2 channels, of to lower the bitrate of an audio stream.

scodec=

This options allows to specify subtitle format the subtitles tracks of the input stream should be converted to.

List of available codecs can be found on the streaming features page (/streaming/features.html).

senc=

This allows to set the converter to use to encode the subtitle stream.

The only subtitle encoder we have at this time is *dvbsub*.

soverlay

This option allow to render subtitles directly on the video, while transcoding it. Do not confuse this option with senc/scodec that transcode the subtitles and streams them

sfilter=

This option allows to render some images generated by a so called *subpicture filter* (e.g. a logo, a text string, etc) on top of the video.

List of available *subpicture filters* can be found on the streaming features page (http://www.videolan.org/streaming/features.html). The Item options of this modules can be found using the following command line:

```
% vlc -p --advanced <module name>
```

threads=

This options allows to set the amount of threads that should be used to encode the streams. Increasing this number to the amount of processors on the computer, (or twice this number on Intel P4 HT processors) should improve transcoding performance.

audio-sync

When this option is enabled, VLC will drop/duplicate video frames to synchronize the video track on the audio track.

duplicate

This module can be used to duplicate the stream, and so process it through several different chains.

Available options are:

dst=

This options allows to give the chain through which the duplicated stream should be processed.

> **Note:** Several *dst=* options have to be used in the same duplicate block to actually duplicate the stream.
>
> Any of the stream output module described earlier can be used as parameter of this option.

select=

This options can be used to duplicate only a part *elementary streams* of a complete stream.

Several criteria can be given, by separating each of them with a comma.

For criteria that need a parameter, such as *es* and *program*, you can also specify a range, using the syntax *criteria=num_start-num_end*.

Available parameters are:

- *program=*: duplicate only *elementary streams* belonging to the selected program (or SID). This option only works with MPEG2/TS streams.
- *noprogram=*: do not duplicate *elementary streams* belonging to the selected program (or PID). This option only works with MPEG2/TS streams.
- *es=*: duplicate only the *elementary stream* with the selected id.
- *noes=*: do not duplicate the *elementary stream* with the selected id.
- *video*: duplicate only video *elementary streams*.
- *novideo*: do not duplicate video *elementary streams*.
- *audio*: duplicate only audio *elementary streams*.
- *noaudio*: do not duplicate audio *elementary streams*.
- *spu*: duplicate only subtitle *elementary streams*.
- *nospu*: do not duplicate subtitle *elementary streams*.

Example:

```
#duplicate{dst=std{...},select="program=100-200,novideo"}
```

This *duplicate* chain will only output the non video *elementary streams* belonging to the programs which PID are between 100 and 200.

Miscellaneous

Here are a few additional global options:

--sout-all, --no-sout-all

Enable streaming of all ES (default disabled). By default VLC will only stream one audio ES and one video ES (the first ones). If you enable sout-all, all ES (audio, video and SPU) will be streamed.

--sout-keep, --no-sout-keep

Keep sout open (default disabled) : use the same sout instance across the various playlist items, if possible.

--no-sout-audio

This options allows to disable audio in the outputted stream.

--no-sout-video

This options allows to disable video in the outputted stream.

Simplified Syntax

The stream output also offers a simplified syntax, with which you can only you use the *standard* module main options:

```
% vlc input_stream --sout access/mux:url
```

where *access*, *mux* and *url* are as defined in the options of the standard module.

Examples

To understand fully the complex syntax of VLC's stream output, please look at the examples in the next section.

Chapter 4. Examples for advanced use of VLC's stream output (transcoding, multiple streaming, etc...)

Transcoding

Transcode the input stream and send it to a multicast IP address with the associated SAP announce:

```
% vlc -vvv input_stream --sout '#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}:
rtp{mux=ts,dst=239.255.12.42,sap,name="TestStream"}'
```

Display the input stream, transcode it and send it to a multicast IP address with the associated SAP announce:

```
% vlc -vvv input_stream --sout '#duplicate{dst=display,dst=
"transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}:
rtp{mux=ts,dst=239.255.12.42,sap,name="TestStream"}"}'
```

Transcode the input stream, display the transcoded stream and send it to a multicast IP address with the associated SAP announce:

```
% vlc -vvv input_stream --sout '#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}:
duplicate{dst=display,dst=rtp{mux=ts,dst=239.255.12.42,sap,name="TestStream"}}'
```

Multiple streaming

Send a stream to a multicast IP address and a unicast IP address:

```
% vlc -vvv input_stream --sout '#duplicate{dst=
rtp{mux=ts,dst=239.255.12.42,sap,name="TestStream"},
dst=rtp{mux=ts,dst=192.168.1.2}}'
```

Display the stream and send it to two unicast IP addresses:

```
% vlc -vvv input_stream
--sout '#duplicate{dst=display,
dst=rtp{mux=ts,dst=192.168.1.12},
dst=rtp{mux=ts,dst=192.168.1.42}}'
```

Send parts of a multiple program input stream:

```
% vlc -vvv multiple_program_input_stream
--sout '#duplicate{dst=rtp{mux=ts,dst=239.255.12.42},select="program=12345",
dst=rtp{mux=ts,dst=239.255.12.43}, select="video,program=1234-2345"}'
```

This command sends the program of the input stream which id is 12345 to 239.255.12.42 and all video programs with id between 1234 and 2345 to 239.255.12.43.

Transcoding and multiple streaming

Transcode the input stream, display the transcoded stream and send it to a multicast IP address with the associated SAP announce and an unicast IP address:

```
% vlc -vvv input_stream --sout '#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}:
duplicate{dst=display,dst=rtp{mux=ts,dst=239.255.12.42,sap,name="TestStream"},
dst=rtp{mux=ts,dst=192.168.1.2}}'
```

Display the input stream, transcode it and send it to two unicast IP addresses:

```
% vlc -vvv input_stream --sout '#duplicate{dst=display,
dst="transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}:
duplicate{dst=rtp{mux=ts,dst=192.168.1.2},
dst=rtp{mux=ts,dst=192.168.1.12}"}'
```

Send the input stream to a multicast IP address and the transcoded stream to another multicast IP address with the associated SAP announces:

```
% vlc -vvv input_stream --sout '#duplicate{dst=
rtp{mux=ts,dst=239.255.1.2,sap,name="OriginalStream"},
dst="transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}:
rtp{mux=ts,dst=239.255.1.3,sap,name="TranscodedStream"}"}'
```

HTTP streaming

Stream in HTTP:

- on the server, run:

  ```
  % vlc -vvv input_stream --sout '#standard{access=http,mux=ogg,dst=server.example.org:8080}'
  ```

- on the client(s), run:

  ```
  % vlc http://server.example.org:8080
  ```

Transcode and stream in HTTP:

```
% vlc -vvv input_stream --sout '#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128}:
standard{access=http,mux=ogg,dst=server.example.org:8080}'
```

For example, if you want to stream an audio CD in Ogg/Vorbis over HTTP:

```
% vlc -vvv cdda:///dev/cdrom --sout '#transcode{acodec=vorb,ab=128}:
standard{access=http,mux=ogg,dst=server.example.org:8080}'
```

RTP streaming

Stream in RTP:

- on the server, run:

  ```
  % vlc -vvv input_stream --sout '#rtp{dst=192.168.0.12,port=1234,sdp=rtsp://server.example.org:8080/test.sdp
  ```

- on the client(s), run:

  ```
  % vlc rtsp://server.example.org:8080/test.sdp
  ```

  Note: You can remplace the *rtsp* URL by a *http* one, a file path (prefixed with *file://*) or *sap://* to export the sdp file using one of these methods instead of rtsp.

RTSP

See *VLM - Multiple streaming and Video on demand*

MMS / MMSH streaming to Windows Media Player

```
% vlc -vvv input_stream --sout '#transcode{vcodec=DIV3,vb=256,scale=1,acodec=mp3,ab=32,channels=2}:std{access
```

VLC media player can connect to this by using the following url : mmsh://server_ip_address:8080 .

Windows Media Player can connect to this by using the following url : mms://server_ip_address:8080 .

Use the *es* module

Separate audio and video in two PS files:

```
% vlc -vvv input_stream --sout
'#es{access=file,mux=ps,url_audio=audio-%c.%m,url_video=video-%c.%m}'
```

Extract the audio track of the input stream to a TS file:

```
% vlc -vvv input_stream --sout '#es{access_audio=file,mux_audio=ts,url_audio=audio-%c.%m}'
```

Stream in unicast the audio track on a port and the video track on another port (NOTE: This will not only work with VLC 0.8.6 or older - FIXME?)

- on the server side:

  ```
  % vlc -vvv input_stream --sout
  '#es{access=rtp,mux=ts,url_audio=192.168.1.2:1212,url_video=192.168.1.2:1213}'
  ```

- on the client side: to receive the audio:

  ```
  % vlc rtp://@:1212
  ```

  to receive the video:

  ```
  % vlc rtp://@:1213
  ```

Stream in multicast the video and dump the audio in a file:

```
% vlc -vvv input_stream --sout '#es{access-video=udp,mux-video=ts,dst-video=239.255.12.42,
access-audio=file,mux-audio=ps,dst-audio=audio-%c.%m}'
```

Note: You can also combine the *es* module with the other modules to set-up even more complex solution.

Chapter 5. VLM - Multiple streaming and Video on demand

VLM

VideoLAN Manager is a small media manager designed to control multiple streams with *only one instance of VLC*. It allows multiple streaming and video on demand (VoD). This manager being a new feature, it can only be controlled by the telnet interface or the http interface.

Interfaces

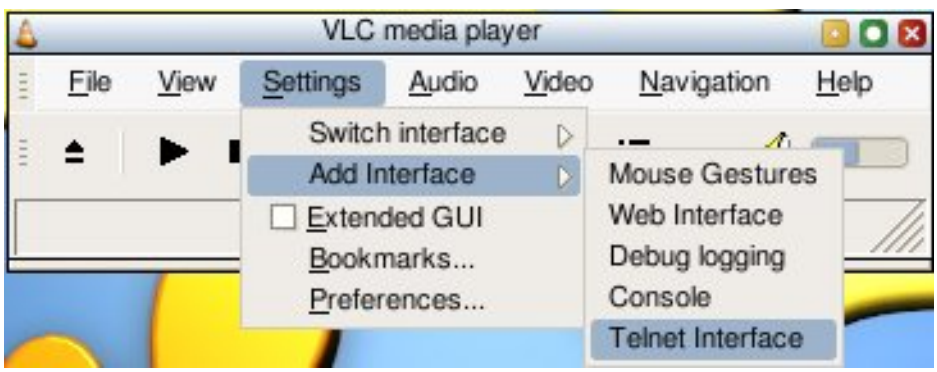
Telnet interface

You can launch the telnet interface as a common interface using the command line :

```
% vlc --intf telnet
```

```
% vlc --extraintf telnet
```

The telnet interface can also be launched in the wxWindows interface :



Launching the Telnet interface - wxWindows interface

The default port is 4212. The default password is "admin". These can be changed using **--telnet-port <integer>** and **--telnet-password <string>** command line options. They can also be changed in the preferences panel when using the wxWindows interface in the *Modules->interface-> telnet* section (check the *Advanced options* checkbox).

HTTP interface

Launching the HTTP interface is described in the Play-with-VLC Howto. (http://www.videolan.org/doc/)

To access the vlm section of the http interface, use the following URL: *http://host:port/vlm.html* (*http://host:port/vlm/* for VLC 0.8.4 and older).

VLM Elements

Medias

A *Media* is composed with a list of inputs (the video and audio streams you want to stream), an output (how and where you want to stream them) and some options.

There are two types of medias:

- *vod*: A vod media is commonly used for Video on Demand. It will be launched only if a vod client asks for it.
- *broadcast*: A broadcast media is very close to a TV program or channel. It is launched, stopped or paused by the administrator and may be repeated several times. The client has no control over this media.

Schedules

A *Schedule* is a script with a date. When the schedule date is reached, the script is launched. There are several options available like a period or a number of repetitions.

Command line syntax

Command lines

- **help** : Displays an exhaustive command lines list
- **new (name) vod|broadcast|schedule [properties]** : Create a new vod, broadcast or schedule element. Element names must be unique and cannot be "media" or "schedule". You can specify properties in this command line or later on by using the **setup** command.
- **setup (name) (properties)** : Set an elements property. See *Media Properties*.
- **show [(name)|media|schedule]** : Display current element states and configurations.
  - **show (name)** - Specify an element's name to show all information concerning this element.
  - **show media** displays a summary of media states.
  - **show schedule** displays a summary of schedule states.
- **del (name)|all|media|schedule** : Delete an element or a group of elements. If the element wasn't stopped, it is first stopped before being deleted.
  - **del (name)** - Delete the (name) element.
  - **del all** - Delete all elements
  - **del media** - Delete all media elements.
  - **del schedule** - Delete all schedule elements
- **control (name) [instance_name] (command)** : Change the state of the (instance_name) instance of the (name) media. If (instance_name) isn't specified, the control command affects the default instance. See *Control Commands* for available control commands.
- **save (config_file)** : Save all media and schedule configurations in the specified config file. The config file path is relative to the directory in which vlc was launched. If the file exists it will be overwritten. Note that states, such as playing, paused or stop, are not saved. See *Configuration Files* for more info.

- **load (config_file)** : Load a configuration file. The config file path is relative to the directory in which vlc was launched. See *Configuration Files* for more info.

Media Properties

> **Note:** Except the "append" property, all properties can be followed by another one.

- **input (input_name)** : Add an input to the end of the media's input list.
- **output (output_name)** : Define the media's output. The syntax is the same as the vlc ":sout=..." vlc option but you do not have to put the ":sout=..." string. See *Advanced streaming using the command line* for more information concerning stream outputs (sout).

> **Note:** You do not have to specify an output for vod elements.

- **option (option_name)[=value]** : Adds the (option_name) to the media option list. The syntax is equivalent to the ":(option)=..." option , but you do not have to put the ":" string. Options are global: they are applied to all inputs of the media.
- **enabled|disabled** : Enable or Disable the media. If a media is disabled, it cannot be streamed, paused, launched by a schedule, or played as VoD.
- **loop|unloop (broadcast only)** : If a media with the "loop" option receives the "play" command, it will automatically restart to play the input list once the end of the input list is reached.

> **Note: loop|unloop** is only used for broadcast media types.

- **mux (mux_name)** : This option should only be specified if you want the elementary streams to be sent encapsulated instead of raw. The (mux_name) should be specified as a four characters length identifier such as mp2t for MPEG TS or mp2p for MPEG PS. See *Streaming, Muxers and Codecs*.

> **Note:** The **mux** property is only used for vod media types.

Schedule Properties

- **enabled|disabled** : A disabled schedule will never be launched.
- **append (command_until_rest_of_the_line)** : Add a command to the command line lit. The command line can be every command VLM can understand.

> **Note:** The rest of the line will be considered as part of the command line. You cannot put another option after the **append** one.

- **date (year)/(month)/(day)-(hour):(minutes):(seconds)|now** : Specify the first date the schedule should be launched. You can specify a date using the **(year)/(month)/(day)-(hour):(minutes):(seconds)** format (example: 2004/11/16-00:43:12) or using the **now** keyword. If **now** is used, the schedule will be launched as soon as possible (i.e. as soon as it is enabled) and the current date will be used as the first date of the schedule.
- **period (years_aka_12_months)/(months_aka_30_days)/(days)-(hours):(minutes):(sec onds)** : Specify the period of time a schedule must wait for launching itself another time. (Months are considered as 30 days, Years as 12 months) If a period is specified without a **repeat** property, the schedule will be launched endlessly.
- **repeat (number_of_repetitions)** : Specify the number of times the schedule will be launched again. For example, if a schedule has **repeat 11**, it will be launched 12 times.

Control Commands

- **play** : Stat a broadcast media. The media begins to launch the first item of the input list, then launches the next one and so on. (like a play list)
- **pause** : Put the broadcast media in paused status.
- **stop** : Stop the broadcast media.
- **seek (percentage)** : Seek in the current playing item of the input list.

Configuration Files

A VLM configuration file is a list of command lines : one line corresponds to one command line.

To create a configuration file, just edit a text file and type a list of VLM commands. Beware of recursive calls: you an put a **load (file)** in a configuration file which can lead to recursive inclusion of the same file and result in VLC's crash.

As of versions > 0.8.1, any line where the first non white space character is a # is considered as a comment.

Examples

This section provides several small vlm configuration files.

Multiple streaming

Simple broadcasting

```
new channel1 broadcast enabled
setup channel1 input http://host.mydomain/movie.mpeg
setup channel1 output #rtp{mux=ts,dst=239.255.1.1,sap,name="Channel 1"}

new channel2 broadcast enabled
setup channel2 input rtp://@239.255.12.42
setup channel2 output #rtp{mux=ts,dst=239.255.1.2,sap,name="Channel 2"}

control channel1 play
control channel2 play
```

Scheduled broadcasting

```
new my_media broadcast enabled
setup my_media input my_video.mpeg input my_other_movie.mpeg
setup my_media output #standard{mux=ts,access=udp,dst=239.255.1.1,sap,name="My Media"}

new my_sched schedule enabled
setup my_sched date 2012/12/12-12:12:12
setup my_sched append control my_media play
```

Video On Demand

Basic example

First launch the vlc

```
% vlc --ttl 12 -vvv --color -I telnet --telnet-password videolan --rtsp-host 0.0.0.0:5554
```

where:

- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).
- *telnet* launches the telnet interface of the vlc.
- *videolan* is the password to connect to the telnet interface.
- *0.0.0.0* is the host address.
- *5554* is the port on which you stream.

Then you connect to the vlc telnet interface and create the vod object

```
new Test vod enabled
setup Test input my_video.mpg
```

You can access to the stream with:

```
% vlc rtsp://server:5554/Test
```

where:

- *server* is the address of the streaming server (IP or DNS)

Chapter 6. Receive and save a stream

Receive a stream with VLC

Receive an unicast stream

```
% vlc -vvv rtp://
```

Receive a multicast stream

```
% vlc -vvv rtp://@239.255.12.42
```

where *239.255.12.42* is the multicast IP address you want to join.

Receive an HTTP/FTP/MMS stream

Use one of the following command lines:

- `% vlc -vvv http://example/stream.xyz`

  where *http://example/stream.xyz* is the HTTP address of the stream;
- `% vlc -vvv ftp://example/stream.xyz`

  where *ftp://example/stream.xyz* is the FTP address of the stream;
- `% vlc -vvv mms://viptvr.yacast.fr/encoderfranceinfo`

  where *mms://viptvr.yacast.fr/encoderfranceinfo* is the MMS address of the stream.

Receive a RTP stream available through RTSP

```
% vlc -vvv rtsp://www.hardradio.com/tonbeme.mov
```

where *rtsp://www.hardradio.com/tonbeme.mov* is the address of the stream.

Receive a stream described by an SDP file

```
% vlc -vvv http://server.example.org/stream.sdp
```

Save a stream with VLC

VLC can save the stream to the disk. In order to do this, use the Stream Output of VLC : you can do it via the graphical interface, or you can add to the command line the following argument:

```
--sout file/muxer:stream.xyz
```

where:

- *muxer* is one of the formats supported by VLC's stream output, i.e. :
  - *ogg* for OGG format,
  - *ps* for MPEG2-PS format,
  - *ts* for MPEG2-PS format.

- and `stream.xyz` is the name of the file you want to save the stream to, with the right extension.

Receive a stream with a set-top-box

Some set-top-boxes with Ethernet cards can receive MPEG2-TS streams over UDP and support multicast.

Set-top-boxes known to work with VLC are :

- Pace (http://www.pace.co.uk) set top boxes. (Pace Micro DSL 4000)
- Aminocom (http://www.aminocom.com) set top boxes. (all the models with mpeg2)
- tuxia / gct-allwell (mpeg4 and mpeg2) sigma designs8174 chipset
- i3micro mood200 (mpeg4 and mpeg2 in transport streams)

Chapter 7. Stream a file

Stream a file with VLC

```
% vlc -vvv video1.xyz --sout udp:192.168.0.42 --ttl 12
```

where:

- `video1.xyz` is the file you want to stream,
- *192.168.0.42* is either:
  - the IP address of the machine you want to unicast to;
  - or the DNS name the machine you want to unicast to;
  - or a multicast IP address.

- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).

If you want to stream the file continuously, add the *--loop* option.

Chapter 8. Stream a DVD

Note: Under Unix/Linux, you must have write access to the device corresponding to your DVD drive. For that, you should be in the *disk* or *cdrom* group (look at the permissions in `/dev`). If you're not, add yourself to the group:

```
# adduser your_login disk_or_cdrom
```

and then restart your session.

Stream a DVD with VLC

```
% vlc -vvv --color dvdsimple:///dev/dvd --sout '#rtp{mux=ts,dst=192.168.0.12}' --ttl 12 --sout-all
```

where:

- `/dev/dvd` is the name of your DVD drive (put `D:` under Windows if `D` is the letter of your DVD drive) or the directory where you copied your DVD ,
- *192.168.0.42* is either:
  - the IP address of the machine you want to unicast to;
  - or the DNS name the machine you want to unicast to;
  - or a multicast IP address.
- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).
- *sout-all* allows you to stream all soundtracks and subtitles

If you want to stream the DVD continuously, add the *--loop* option.

Chapter 9. Stream a DVB channel (satellite or digital terrestial TV)

Note: This is possible under GNU/Linux only.

Install the DVB drivers

If you want to be able to stream from a DVB card (a satellite card or a digital terrestial TV card), you need to install the DVB drivers:

- if you use a Linux 2.6.x kernel, you just need to select the right modules in your kernel configuration.
- if you are using a Linux 2.4.x kernel, you must download the latest release of the DVB drivers from the DVB drivers download page (http://www.linuxtv.org/download/dvb/) of the LinuxTV (http://www.linuxtv.org/) Project.

The following sections assume that you have a working linux-dvb installation, either from stock kernel 2.6 or from kernel 2.4 with DVB patches. If you have any problem with the linux-dvb drivers, please report the problem to the maintainers of the drivers, not to us. Thanks.

Stream with VLS

Note: VLS is currently deprecated and hasn't been maintained for years. It is strongly advised to use VLC instead, which now supports the same features as VLS, and many more. The only advantage of VLS is to support the dvbrc file syntax, and it requires a bit less CPU horsepower. However, we do not support VLS any longer.

Put a `.dvbrc` file containing the DVB channels (satellite or digital terrestial TV channels) you want to stream in your home directory (some are provided in the *libdvb* tarball for the satellite channels).

Run VLS with the following command line :

```
% vls -vv -d udp:192.168.0.42 dvb:"EUROSPORT" --ttl 12
```

where:

- *"EUROSPORT"* is the channel you want to stream as written in your `~/.dvbrc` file ,
- *192.168.0.42* is either :
  - the IP address of the machine you want to unicast to;
  - or the DNS name the machine you want to unicast to;
  - or a multicast IP address.
- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).

Stream with VLC

> **Note:** VLC has many more features than VLS. First you can use the advanced stream output options such as transcoding and all kinds of output supports. Second VLC can take advantage of the Common Interface supported by some DVB adapters to descramble one or several services. Currently released versions of VLC only support the low-level API so some adapters won't work (budget-ci cards work, twinhan doesn't). Some CAM modules aren't compatible with some DVB cards, check the linux-dvb documentation for more information. So-called "professional" CAM modules are able to descramble up to twelve services, whereas customer-oriented modules are often limited to one or two services unless otherwise specified.

VLC must be compiled with --enable-dvb and you need the linux-dvb headers installed in your system. An example command-line is as follows :

```
%
vlc -vvv --color --ttl 12  --ts-es-id-pid --programs=8508,8505 dvb: \
--dvb-frequency=11739000 --dvb-srate=27500000 --dvb-voltage=13 \
--sout-standard-access=udp --sout-standard-mux=ts --sout \
'#duplicate{dst=std{dst=address1},select="program=8508",dst=std{dst=address2},select="program=8505"}'
```

The example above shows the minimum set of options needed to stream out two services. Here is a list of frontend options, depending on the frontend type:

- *common options*
  - *dvb-adapter* : specifies the adapter to use in case you have several adapters in your machine (by default use adapter 0)
  - *dvb-device* : specifies the name of the DVB device to use (should not be needed with a standard linux-dvb installation)
  - *dvb-srate* : specifies the symbol rate of the modulated signal, in symbols/s
  - *dvb-inversion* : specifies whether the signal is inverted or not (default is automatic detection)
  - *dvb-budget-mode* : enters a special mode where all PIDs are retrieved by the driver; it should no longer be necessary as VLC should filter wanted PIDs

- *satellite frontend (QPSK)*
  - *dvb-frequency* : specifies the frequency to tune to in kHz; according to the frequency range, VLC auto-detects the band to use: S (2.5-2.7 GHz), C-lower (3.4-4.2 GHz), C-higher (4.5-4.8 GHz), Ku (10.7-13.25 GHz) or direct BIS frequency (0.95-2.15 GHz); it is mandatory to supply the *dvb-srate* option to satellite frontends
  - *dvb-voltage* : specifies the voltage to apply on the IF; most LNBs behave differently when supplied with 13 V or 18 V; universal LNBs select vertical polarity with 13 V and horizontal with 18 V; you can also select 0 V if your LNB has another power supply (default is 13 V)
  - *dvb-tone* : specifies whether to send a 22 kHz pulse tone to the LNB; universal LNBs switch to high-band when this pulse is sent; by default VLC automatically adopts the correct behaviour if the frequency supplied is in the Ku band (other bands do not need this)
  - *dvb-fec* : specifies the code-rate to use for Forward Error Correction; type in the first number of the code-rate, for 2/3 use --dvb-rate=2, etc. (default is 9, meaning automatic detection)
  - *dvb-high-voltage* : enables a special mode of the DVB adapter to compensate for the voltage loss in very long cables (AFAIK it is present in the API, but no DVB adapter actually implements it)
  - *dvb-lnb-lof1, dvb-lnb-lof2, dvb-lnb-slof* : specifies the frequencies of the first and second local oscillators, and the frequency at which the 22 kHz pulse should be activated to enable the second oscillator; by default VLC uses the values for universal LNBs if the frequency supplied is in the Ku band (other bands do not need this)

- *cable frontend (QAM)*
  - *dvb-frequency* : specifies the frequency to tune to in Hz; it is mandatory to supply the *dvb-srate* option to cable frontends
  - *dvb-modulation* : specifies the modulation of the analog signal; valid values are -1 (QPSK), 0 (automatic QAM, default), 16 (QAM16), 32 (QAM32), 64 (QAM64) 128 (QAM128), 256 (QAM256)

- *terrestrial frontend (OFDM)*
  - *dvb-frequency* : specifies the frequency to tune to in Hz; it is mandatory to supply the *dvb-bandwidth* option, all other parameters are optional
  - *dvb-bandwidth* : specifies the bandwidth of the OFDM channel (6, 7 or 8 MHz depending on the country)
  - *dvb-hierarchy* : specifies if the OFDM channel uses hierarchic information; allowed values are -1 (no hierarchy), 0 (automatic, default), 1, 2 and 4
  - *dvb-code-rate-hp, dvb-code-rate-lp* : specifies the code-rate to use for higher and lower hierarchies respectively (default auto, same syntax as *dvb-fec*)
  - *dvb-guard* : specifies the guard interval; valid values are 0 (automatic, default), 4 (1/4), 8 (1/8), 16 (1/16) and 32 (1/32)
  - *dvb-transmission* : specifies the transmission mode; valid values are 0 (automatic, default), 2 (2K) and 8 (8K)

We also ought to explain the other non-dvb-specific options of the example command-line:

- *ts-es-id-pid* : this option is necessary if you use the *#duplicate* stream output filter to split the multiplex in several outputs; there is no need to use *#duplicate* neither *ts-es-id-pid* if you have one program only
- *programs, program, sout-all* : there are several ways of specifying the services to select (and optionally descramble):
  - *programs* : used to specify one or serveral programs to select; VLC selects all known elementary streams of these programs; this is the currently recommended way
  - *program* : used to specify one program to select; it differs from using *programs* with only one program in that this option only select the first audio stream, and no subtitle stream; it should be used if you plan to switch programs and audio with a GUI
  - *sout-all* : tells VLC to select all programs; this is discouraged because of the extra CPU load needed to demultiplex unwanted programs, and because it is not compatible with CAM descrambling
- The other options are standard stream output options and are described in the other chapters of this documentation.

Chapter 10. Stream from encoding cards and other capture peripherals

Hardware encoding cards

Note: This is possible under GNU/Linux only.

VideoLAN supports two kinds of MPEG-2 encoding cards:

- Hauppauge WinTV-PVR-250 and WinTV-PVR-350,
- Visiontech Kfir.

The Hauppauge WinTV-PVR-250/350 gives much better results and is cheaper than the Visiontech Kfir.

Stream with the Hauppauge WinTV-PVR-250/350 card

Install the drivers

First, you will have to patch your kernel (version 2.4) to support the v4l2 API (Video 4 Linux version 2). The patch is available on the Video4Linux HQ (http://bytesex.org/v4l/). If you use a 2.6 kernel, you only need to build I2C support and the BT848 Video For Linux module.

Once your kernel is ready, install the CK version (currently in development) of the Linux drivers for the Hauppauge WinTV-PVR-250/350. They are hosted on ivtv ck (http://67.18.1.101/~ckennedy/ivtv). You will need to patch your kernel to use it with a 2.4. You can also use the CVS version available here: ivtv.sourceforge.net (http://ivtv.sourceforge.net/) (this version is not developped anymore). Then, you will have to create the device and load the modules; for this, please refer to the documentation shipped with the drivers.

Stream with VLC

Note: You must add **--enable-pvr** to **./configure** to use this feature.

```
% vlc -vvv --color pvr:///dev/video0:norm=secam:size=720x576:frequency=576250:
bitrate=3000000:maxbitrate=4000000 --cr-average 1000 --sout '#rtp{mux=ts,dst=192.168.0.42}' --ttl 12
```

where :

- */dev/video0* is the device corresponding to the encoding card ,
- *norm=secam* is name of the standard of the analogic signal (possible values are pal, secam, and ntsc) ,
- *size=720x576* is the size of the video you want to stream ,
- *frequency=567250* is the frequency in kHz of the channel you want to stream ,
- *bitrate=3000000* is the average bitrate of the stream ,
- *maxbitrate=4000000* is the maximum bitrate of the stream ,

- *1000* is a secret value to work around a bug of the card.
- *192.168.0.42* is either :
  - the IP address of the machine you want to unicast to ;
  - or the DNS name the machine you want to unicast to ;
  - or a multicast IP address.

- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).

Stream with the Visiontech Kfir card

Install the drivers

If you want to be able to stream from a Visiontech Kfir card, you need to install its Linux drivers. Download the latest release of the drivers from the drivers download page (http://www.linuxtv.org/download/mpeg2/) of the LinuxTV web site (http://www.linuxtv.org/).

Uncompress the tarball and follow the instructions written in the `INSTALL` file to compile and install the drivers.

> **Note:** If you have a VIA chipset, you need to disable USB in the BIOS.

Stream

```
% vlc -vvv --color kfir:///dev/video --sout '#rtp{mux=ts,dst=192.168.0.42}' --ttl 12
```

where :

- `/dev/video` is the device corresponding to the Kfir card ,
- *192.168.0.42* is either :
  - the IP address of the machine you want to unicast to ;
  - or the DNS name the machine you want to unicast to ;
  - or a multicast IP address.

- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).

Software encoding cards

Under GNU/Linux

Install the Video for Linux drivers

If you want to stream from an acquisition card or a webcam, a video4linux driver must be available for it. You can find more information about video4linux and supported devices here (http://www.exploits.org/v4l).

Compile the right module for your device, and insert it into your kernel. Some video4linux modules are shipped with the 2.4.x and 2.6.x Linux kernels, the patch is available on the Video4Linux HQ (http://bytesex.org/v4l).

You can test your device by using any of the listed programs in the *Video: TV and PVR/DVR* section of this page (http://www.exploits.org/v4l/).

Note that v4l2 modules will also work with VLC.

Stream with VLC

Note: You must add **--enable-v4l** to **./configure** to use this feature.

```
% vlc -vvv --color v4l:///dev/video:norm=secam:frequency=543250:size=640x480:channel=0:adev=/dev/dsp:audio=0
--sout '#transcode{vcodec=mp4v,acodec=mpga,vb=3000,ab=256,venc=ffmpeg{keyint=80,hurry-up,vt=800000},deinterla
```

Note: You can find all transcode options on this page : *Advanced streaming using the command line* .

where:

- `/dev/video` is the device corresponding to your acquisition card or your webcam,
- *norm=secam* is name of the standard of the analogic signal (possible values are pal, secam, and ntsc) ,
- *frequency=543250* is the frequency of the channel in kHz (*Warning :* for VLC < 0.6.1, Frequency is channel frequency in MHz multiplied by 16),
- *size=640x480* is the size of the video you want (you can also put the standard size like *subqcif* (128x96), *qsif* (160x120), *qcif* (176x144), *sif* (320x240), *cif* (352x288) or *vga* (640x480)),
- *channel=0* is the number of the channel (usually 0 is for tuner, 1 for composite and 2 for svideo),
- *adev=/dev/dsp* is the audio device,
- *audio=1* is the number of the audio channel (usually 0 is for mono and 1 for stereo),
- *vcodec=mp4v* is the video format you want to encode in (*mp4v* is MPEG-4, *mpgv* is MPEG-1, and there is also *h263*, *DIV1*, *DIV2*, *DIV3*, *I420*, *I422*, *I444*, *RV24*, *YUY2*),
- *acodec=mpga* is the audio format you want to encode in (*mpga* is MPEG audio layer 2, *a52* is A52 i.e. AC3 sound),
- *vb=3000* is the video bitrate in Kbit/s
- *ab=256* is the audio bitrate in Kbit/s
- *venc=ffmpeg* allows to set the encoder to use, where:
  - *keyint=80* is the maximal amount of frames between two key frames

  - *hurry-up* allows the encoder to decrease the quality of the stream if the CPU can't keep up with the encoding rate
  - *vt=800000* is the tolerance in kbit/s for the bitrate of the outputted video

- *deinterlace* tells VLC to deinterlace the video on the fly,
- *192.168.0.42* is either:
  - the IP address of the machine you want to unicast to;
  - or the DNS name the machine you want to unicast to;
  - or a multicast IP address.

- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).

Stream with DirectShow

Note: This is only possible under Windows

Install your peripheral drivers

You need to install your peripherals under Windows with the appropriate drivers. Nothing else is necessary.

Stream with VLC in command line

```
% C:\Program Files\VideoLAN\VLC\vlc.exe -vvv -I rc --ttl 12 dshow:// vdev="VGA USB Camera" adev="USB Camera"
```

Note: You need to precise the complete path to find vlc program or to launch the command from the correct directory.

- *vvv* is to activate the verbose mode
- *rc* is to activate the remote control interface (MS/DOS console)
- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers),
- *vdev="VGA USB Camera"* is the name of the video peripheral that DirectShow will use (this is only an exemple),
- *adev="USB Camera"* is the name of the audio peripheral,
- *size="640x480"* is the resolution (you can also put the standard size like *subqcif* (128x96), *qsif* (160x120), *qcif* (176x144), *sif* (320x240), *cif* (352x288) or *vga* (640x480)).
- *239.255.42.12* is either:

- the IP address of the machine you want to unicast to
- or the DNS name the machine you want to unicast to
- or a multicast IP address.

Chapter 11. Stream from a DV camcorder

Note: This is possible under GNU/Linux only.

Install the libraw1394 and libavc1394

If you want to be able to stream from a DV camcorder, then you need to install the libraries libraw1394 and libavc1394:

- if you use a Fedora Core distribution then you just need to install the libraries using:

```
%yum update
%yum install libraw1394 libavc1394
```

- if you want to install the libraries from the source then you must download them from the libraw1394 (http://www.linux1394.org/) and libavc1394 (http://sourceforge.net/projects/libavc1394) from their projects website.
- if you have a distribution that uses udev (http://kernel.org/pub/linux/utils/kernel/hotplug), then you must add/change the following line to the file 50-udev.rules in your /etc/udev/rules.d directory.

```
%vi /etc/udev/rules.d/50-udev.rules
# IEEE1394 (firewire) devices (must be before raw devices below)
KERNEL=="raw1394",              NAME="%k"
KERNEL=="dv1394",               NAME="dv1394/%k"
KERNEL=="video1394*",           NAME="video1394/%n"
```

The following sections assume that you have a working linux installation with the IEEE 1394 (Firewire) libraries installed, either manually from the source code or through your distributions upgrade mechanism.

Stream with DV

Connect the DV camcorder with a Firewire cable to your computer, and check the creation of the file `/dev/raw1394`.

Run VLC with the following command line :

```
% vlc -vvv dv/rawdv:///dev/raw1394 --dv-caching 10000-sout '#transcode{vcodec=WMV2,vb=512,scale=1,acodec=mp3,
```

where:

- *dv/rawdv://* is the DV input and `/dev/raw1394` the device file ,
- *dv-caching* is the delay is miliseconds (ms) (start with a high value, 10s or so, and lower it later) ,
- *sout* is the stream output chain that is used to stream the DV camcorder as a multimedia stream over the network. The *transcode* syntax is explained in the chapter about transcoding. The example as given above generates a multimedia stream that is compatible with Windows Media Player ,
- *sout-transcode-fps* is the number of pictures per second *25.0* that the transcode module should generate of the requested audio/video codec.

Chapter 12. Streaming over IPv6

Streaming over IPv6

This chapter covers the specifics of streaming over IPv6. You should still read the previous chapters if you are not confortable with streaming in general.

Requirements

You will obviously need an IPv6-aware operating system. That includes Windows XP/2003, Linux 2.6, Mac OS X (starting from version 10.2). Windows 2000 and Linux 2.4 are supported too, but their IPv6 stacks are not as good, so upgrade if you can. IPv6 must be properly configured and working on your system and network.

On Linux, the *ipv6* kernel module must be loaded (or compiled-in). On Windows, the IPv6 protocols suite can be installed by running "ipv6 install" from the command line, or through the Network configuration panel.

Note: Under Windows 2000, you must add by hand a default multicast IPv6 route, with the following command:

```
# ipv6 rtu ff::/8 4
```

where the last number (*4* in this example) is the number of your true IPv6 interface. To have a list of your IPv6 interfaces, run **ipv6 if**.

Warning

Under Windows XP SP1, you may have problems with a hidden IPv6 firewall. To solve the problem, go to the list of Windows Services and stop the IPv6 firewalling service. You should consider upgrading to Service Pack 2 which provides an integrated IPv4/IPv6 firewall that can be configured through the GUI.

Warning

If you are using VMware under Linux, you will have to stop VMware and unload the VMware kernel modules, because we noticed it prevented IPv6 streaming !

Limitations

There are still some features of the VLC media player which do not support IPv6. In particular, it is not possible to use RTSP over IPv6 because the underlying library, Live.com, does not support IPv6 at the time of writing.

Additionnaly, note that at the moment, VLC defaults to using IPv4 mostly every, as it is what most people uses. That might be changed to something more transparent in future versions.

Streaming with VLC

With the Streaming Wizard (GUI)

The streaming wizard accepts IPv6 addresses between braces, for example: **`[2002:8ac3:802d:1242:211:11ff:fe25:e6b4]`**. If you specify a link-local address, you will most likely need to specify the networking interface to use. On Unix, that can be done this way: **`[fe80::211:11ff:fe25:e6b4%eth0]`** to attach to eth0. Similarly, on Windows, you may specify **`[fe80::211:11ff:fe25:e6b4%1]`** where 1 is the number of the network interface as defined by the operating system.

If you're streaming over HTTP, note that IPv6 is automatically used by default (so that both IPv6 and IPv4 clients will be allowed).

If you want to specify DNS hostname, keep in mind that the VLC defaults to IPv4 resolution. You must either specify hostnames that only resolves to IPv6 addresses, or enable the "Force IPv6" *advanced* option in *Preferences / General Settings / Input*.

From the command-line

The **`--ipv6`** command line option force the use of IPv6 by default (ie. IPv6 is always attempted before IPv4).

```
% vlc -vvv video1.xyz --sout '#rtp{mux=ts,dst=ff08::1}' --ttl 12
```

where:

- `video1.xyz` is the file you want to stream (you can also put **dvdsimple:///dev/dvd** to stream a DVD or any other input configuration) ,
- *ff08::1* is either :
  - the IPv6 address of the machine you want to unicast to;
  - or the multicast IPv6 address.

- *12* is the value of the TTL (Time To Live) of your IP packets (which means that the stream will be able to cross 11 routers).

  Note: You may have to specify the output network interface:

  ```
  % vlc -vvv video1.xyz --sout '#rtp{mux=ts,dst=ff02::1%eth0}' --ttl 12
  ```

  where *eth0* is the name of the network interface (under Linux the network interfaces are named *ethX*, under Mac OS X it's *enX* and under Windows it's *X*, where *X* is the appropriate number).

Receiving an IPv6 stream

With the graphical user interface

Select File / Open Network Stream. To receive an UDP/RTP unicast stream sent to your system, you should select the Force IPv6 option (and possibly adjust the destination UDP port). To receive an UDP multicast stream, select the

UDP/RTP Multicast option, and specify the multicast address to subscribe to inside square brackets. The IPv6 addresses syntax is the same as that explained in the *Streaming over IPv6* section of this chapter.

From the command line

IPv6 is used automatically if available (with version 0.8.6 or more).

```
% vlc -vvv --ipv6 rtp://@[ff08::1]
```

Note: Under Unix/Linux, you may have to protect the square brackets around the IPv6 address:

```
% vlc -vvv -rtp:@//\[ff08::1\]
```

Note: You may have to specify the output network interface:

```
% vlc -vvv rtp://@[ff02::1%eth0] --ttl 12
```

where *eth0* is the name of the network interface (under Linux the network interfaces are named *ethX*, under Mac OS X it's *enX* and under Windows it's *X*, where *X* is the appropriate number).

Appendix A. GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program " means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification ".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a

work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License.

   Exception:: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED

WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.