Icecast Streaming Handbook

Icecast – Ezstream – Ices – Video Lan Client

Created By
David Childers

SCVI.NET

www.scvi.net

License

Icecast Streaming Handbook researched and created by David Childers

Icecast documentation written by members of the Xiph foundation community.

Ezstream documentation written by members of the Xiph foundation community.

Ezstream Man Page written by Moritz Grimm.

IceS 2.0 documentation written by members of the Xiph foundation community.

IceS 0.4 documentation was mostly written by Alexander Haväng.

Video Lan Client documentation written by members of the Video Lan Client community.

Old style microphone image retrieved from : http://www.clker.com/clipart-26834.html

Index

Sourcing Multimedia With The Video Lan Client (VLC) - Introduction - Installing VLC With libshout Support - Command Line Syntax For Streaming Content To An Icecast Server - VLC Icecast Output (Command Line Options)	

Foreword

I have created a comprehensive reference for users of the Icecast multimedia streaming server software. Icecast is a very robust Open Source software application that can be used on various operating systems and can be used to stream both audio and video multimedia.

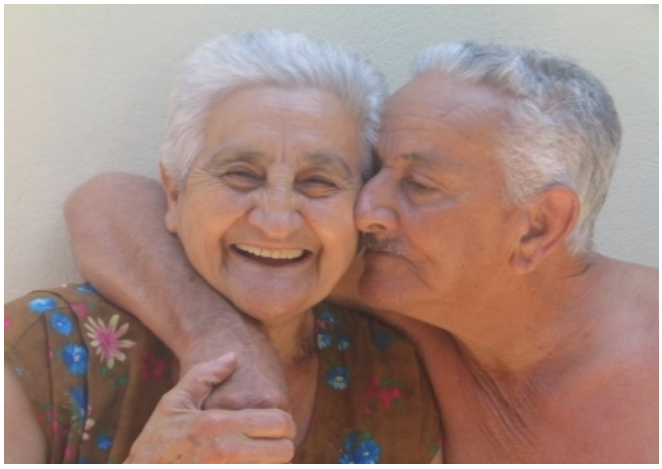
I would like to thank the Xiph Corporation for creating the wonderful Icecast multimedia server software, Scarlet Coker for providing assistance with the review of the manuscript and to James Davey at Broadcasting World for allowing me the opportunity to create this handbook.

It is my sincere hope that the reader finds this guide a valuable resource.

David Childers
www.scvi.net
July, 2009

Sustaining Member

Society Motion Picture
and Television Engineers



O sorriso de Deus estara sempre contigo
Voce estara sempre em nossos coracoes

Give a person a fish and you feed them for a day;
teach that person to use the Internet and they won't bother you for weeks.

Author Unknown

Icecast 2.3.2

Changes

Version 2.3

New Features

Streaming support for ogg speex, ogg flac, ogg midi
intro file support - per mount settable
on-demand relays, global and per-relay settable
fallback to file, extends on the intro file handling.
new mount-level settings

1. public, type/subtype, genre settings, stream description,
2. stream url, stream name, bitrate (override what is sent from the source client)
3. mp3 metadata interval
4. on-[dis]connect scripts can be stated per-mount, invoked at source start/stop and take 1 arg which is the mountpoint.

New URL listener authenticator .included is an example php-based application that can be used in conjunction with the url authenticator to manage a simple subscription-based broadcast.
HTPasswd authenticator uses in-memory structures now.
On demand files now can be fed through an authenticator
Update to admin/web xslt interface

Fixes

real/helix works
win32 access log correct
stats client is stable now (curl -X STATS http://admin@host:port/)
show mountpoints on stats that are inactive but have an active fallback
more updates over HUP possible

Introduction

What Is Icecast ?

Icecast is a streaming media server which currently supports Ogg Vorbis and MP3 audio streams. It can be used to create an Internet radio station or a privately running jukebox and many things in between. It is very versatile in that new formats can be added relatively easily and supports open standards for communication and interaction.

There are two major parts to most streaming media servers: the component providing the content (what we call source clients) and the component which is responsible for serving that content to listeners (this is the function of icecast).

What Platforms Are Supported ?

Currently the following Unix platforms are supported:

- Linux (Most flavors including Redhat and Debian)
- FreeBSD
- OpenBSD
- Solaris

Currently the following Windows platforms are supported:

- Windows NT
- Windows 2000
- Windows XP

Where Do I Go For Questions?

There are many ways to contact the icecast development team
Best Ways

- Icecast mailing list http://www.xiph.org/archives
- Icecast Developers mailing list http://www.xiph.org/archives
- Icecast IRC chat room - irc.freenode.net : #icecast

Alternate Ways

- team@icecast.org

Basic Setup

Basic Requirements

This section will describe the essential requirements in setting up a simple Internet radio station. It is by no means a complete list but should give you enough to get started.

There are two major components involved: the streaming server (icecast in this case) and the source client. The icecast server will be the place where all listeners of your station will connect. The source client (in general) runs on a separate machine than icecast, but does not necessarily need to. Source clients send the content to icecast and provide the stream data (encoded audio) that is then relayed out to listeners by icecast.

It is important to note that not all source clients work with icecast2. You will need to check to make sure that icecast2 is supported by your chosen source client.

The Basics

Each icecast server can house multiple broadcasts (or mountpoints) each containing a separate stream of content. A 'mountpoint' is a unique name on your server identifying a particular stream - it looks like a filename, such as '/stream.ogg'. A listener can only listen to a single mountpoint at a time. This means you can have a single icecast server contain either multiple broadcasts with different content, or possibly the same broadcast but with streams of different bitrates or qualities. In this case each broadcast or stream is a separate mountpoint.

At this point, the steps outlined here related to the Unix version or Win32 console version of icecast. Icecast is also available in a Win32 GUI version, and the steps are similar in setup, but not quite the same.

The first step in the process is to find and install the icecast2 server itself. How to do this is not contained within this documentation. After installation you should have and icecast binary and 3 directories

conf - Contains the icecast configuration file (icecast.xml) which defines all the configuration parameters for the server.
admin - Contains xslt files which are used by the icecast server to provide a web-based front end to the administration capabilities of the server.
logs - This is a blank directory which (if specified in the config file) will contain all the logs (there are 2) for icecast.

The next step is to edit the icecast.xml file and set the appropriate values. Most of the specified values in the samples are fine, for a basic setup the following entries should be specified, and if neccessary, changed to suite your situation :

<hostname> - DNS name or IP address used for stream directory listings.
<source-password> - will be used for the source client authentication
<admin-password> - will be used for authenticating admin features of icecast
<listen-socket> (both port and bind-address)
<logdir> - directory where log files will be placed
<webroot> - directory for non admin content (file serving root), status page is provided
<adminroot> - directory containing admin xslt files

Most will not need to change adminroot/webroot as those are only read by icecast but icecast needs to create files in the logdir which MAY mean you want to be more selective. The other settings above will need to be provided by the icecast administrator

Once the configuration file is modified, you should be able to start the server with the following command

icecast -c /path/to/icecast.xml

If no error messages are generated, then check the error.log file for the 'server started' message, it will look something like :-

[2003-10-31 13:04:49] INFO main/main.c Icecast 2.3.0 server started

You may notice slight variations to the line above, the time will no doubt be different, and on some platforms the main.c is just main, but the key thing here is that the server is started, logging is working and the version is shown.

You can also verify that it started by visiting the following URL : http://yourip:port/admin/stats.xml. You should be prompted for a username and password. Enter the username "admin" and the password you entered for <admin-password>. If all is well, you should see an small XML tree which represents icecast statistics (more about that later).

Now that the icecast server is started you must now configure your source client. The information you will need for the source client is the following :

IP address and Port of the icecast server - both of these come from <listen-socket>
source password - from <source-password>

Additionally, you will need to choose a mountpoint and specify this in the source client. Icecast does not need to know about each mount point (although you can configure settings for specific mountpoint - this is covered under Advanced configuration) there are, however, some points to mention regarding mountpoints. All Ogg Vorbis streams should have mountpoints that end in .ogg (i,e. /mystream.ogg). This is due to the lazy way most media players infer the type of stream. MP3 streams usually do not contain an extension (/mystream). Mount points also should not contain any spaces or odd characters (again due to the lazy way many of the media players are coded).

Once you have configured your source client, you should be able to connect it to the icecast server. Verify that it is connected by hitting the stats.xml URL that was mentioned above.

Now that you have the source connnected, listening to the stream involves simply opening the appropriate following URL in a browser: http://yourip:port/mounpointyouspecified.m3u. So, for instance, if you attached your source client to an icecast server located at 192.168.1.10:8000 with a mountpoint of /mystream.ogg, then you would open : http://192.168.1.10:8000/mystream.ogg.m3u. Note that the .m3u extention will serve up a link that opens most media players. Also it is important to note that m3u need not contain only MP3 stream, it can contain streams of arbitrary content-type and is used by icecast to serve a playlist that represents your broadcast to listening clients. Alternatively you can open up the stream URL directly within your media player (http://192.168.1.10:8000/mystream.ogg in this case)

Config File

Overview

This section will describe each section of the config file and is grouped into the following sections:

- Limits
- Authentication
- Stream Directory Settings
- Misc Server settings
- Relay settings
- Mount Specific settings
- File path settings
- Logging
- Security

Limits

```
<limits>
    <clients>100</clients>
    <sources>2</sources>
    <queue-size>102400</queue-size>
    <client-timeout>30</client-timeout>
    <header-timeout>15</header-timeout>
    <source-timeout>10</source-timeout>
    <burst-on-connect>1</burst-on-connect>
    <burst-size>65536</burst-size>
</limits>
```

This section contains server level settings that, in general, do not need to be changed. Only modify this section if you are know what you are doing.

clients

Total number of concurrent clients supported by the server. Listeners are considered clients, but so are accesses to any static content (i.e. fileserved content) and also any requests to gather stats. These are max *concurrent* connections for the entire server (not per mountpoint).

sources

Maximum number of connected sources supported by the server. This includes active relays and source clients.

queue-size

This is the maximum size (in bytes) of the stream queue. A listener may temporarily lag behind due to network congestion and in this case an internal queue is maintained for the listeners. If the queue grows larger than this config value, then it is truncated and any listeners found will be removed from the stream.
This will be the default setting for the streams which is 512k unless overridden here. You can override this in the individual mount settings which can be useful if you have a mixture of high bandwidth video and low bitrate audio streams.

client-timeout

This does not seem to be used.

header-timeout

The maximum time (in seconds) to wait for a request to come in once the client has made a connection to the server. In general this value should not need to be tweaked.

source-timeout

If a connected source does not send any data within this timeout period (in seconds), then the source connection will be removed from the server.

burst-on-connect

This setting is really just an alias for burst-size. When enabled the burst-size is 64 kbytes and disabled the burst-size is 0 kbytes. This option is deprecated, use burst-size instead.

burst-size

The burst size is the amount of data (in bytes) to burst to a client at connection time. Like burst-on-connect, this is to quickly fill the pre-buffer used by media players. The default is 64 kbytes which is a typical size used by most clients so changing it is not usually required. This setting applies to all mountpoints unless overridden in the mount settings.

Authentication

```
<authentication>
    <source-password>hackme</source-password>
    <relay-user>relay</relay-user>
    <relay-password>hackme</relay-password>
    <admin-user>admin</admin-user>
    <admin-password>hackme</admin-password>
</authentication>
```

This section contains all the usernames and passwords used for administration purposes or to connect sources and relays.

source-password

The unencrypted password used by sources to connect to icecast2. The default username for all source connections is 'source' but this option allows to specify a default password. This and the username can be changed in the individual mount sections.

relay-user

Used in the master server as part of the authentication when a slave requests the list of streams to relay. The default username is 'relay'

relay-password

Used in the master server as part of the authentication when a slave requests the list of streams to relay.

admin-user
admin-password

The username/password used for all administration functions. This includes retrieving statistics, accessing the web-based administration screens, etc. A list of these functions can be found in the "Administration" section of the

manual.

Stream Directory Settings

```
<directory>
    <yp-url-timeout>15</yp-url-timeout>
    <yp-url>http://dir.xiph.org/cgi-bin/yp-cgi</yp-url>
</directory>
```

This section contains all the settings for listing a stream on any of the Icecast2 YP Directory servers. Multiple occurrences of this section can be specified in order to be listed on multiple directory servers.

yp-url-timeout

This value is the maximum time icecast2 will wait for a response from a particular directory server. The recommended value should be sufficient for most directory servers.

yp-url

The URL which icecast2 uses to communicate with the Directory server. The value for this setting is provided by the owner of the Directory server.

Misc Server Settings

Server wide settings.

```
<hostname>localhost<hostname>
<fileserve>1</fileserve>
<server-id>icecast 2.3</server-id>
```

hostname

This is the DNS name or IP address that will be used for the stream directory lookups or possibily the playlist generation if a Host header is not provided. While localhost is shown as an example, in fact you will want something that your listeners can use.

fileserve

This flag turns on the icecast2 fileserver from which static files can be served. All files are served relative to the path specified in the <paths><webroot> configuration setting. By default the setting is enabled so that requests for the images on the status page are retrievable.

server-id

This optional setting allows for the administrator of the server to override the default server identification. The default is icecast followed by a version number and most will not care to change it however this setting will change that.

The following shows how you can specify the listening settings for the server.

The first shows an example of a common and simple way to define a listening socket

```
<listen-socket>
    <port>8000</port>
</listen-socket>
```

Using this as a basis we can extend this with an <bind-address> setting to limit which address icecast will listen on. Most will not need to use bind-address and often get confused by using it when there is no need. Another possibility is to use an <ssl> boolean setting which informs icecast that a secured connection is to be used. A common use for using a secure connection would be for admin page access.

The following shows how we can extend a single listen-socket to work with shoutcast style source clients. There are two issues shoutcast source clients have over icecast source clients, one is the lack of mountpoint and the second is the requirement of two ports. Both of these issues are handled by a simple addition in the listen-socket.

```
<listen-socket>
    <port>8000</port>
    <shoutcast-mount>/live.mp3</shoutcast-mount>
</listen-socket>
```

As before the port specified is allocated but this time the shoutcast-mount implicity defines a second listening socket whose port number is always one higher than the port defined, this also informs icecast of which mountpoint the shoutcast source client on this socket will be using. Using this approach you can allow multiple shoutcast source clients to connect at the same time.

The following is just to show the longer approach to defining shoutcast compatability.

```
<shoutcast-mount>/live.nsv</shoutcast-mount>

<-- You may have multiple <listen-socket> elements -->
<listen-socket>
   <port>8000</port>
</listen-socket>

<listen-socket>
   <port>8001</port>
   <shoutcast-compat>1</shoutcast-compat>
</listen-socket>
```

Note that multiple listen-socket sections may be configured in order to have icecast2 listen on multiple network interfaces or multiple ports. If a bind-address is not specified for a particular listen-socket, then the socket will be bound to all interfaces (including IPv6 if available). For most people, the bind-address option will not be required and often confuses people.

port

The TCP port that will be used to accept client connections.

bind-address

An optional IP address that can be used to bind to a specific network card. If not supplied, then it will bind to all interfaces.

shoutcast-mount

An optional mountpoint setting to be used when shoutcast DSP compatible clients connect. The default global setting is /stream but can be overridden here to use an alternative name which may include an extension that some clients require for certain formats.

Defining this within a listen-socket group tells icecast that this port and the subsequent port are to be used for shoutcast compatible source clients. This is an alternative to the shoutcast-compat approach as this implicitly defines the second listening socket and allows for specifying multiple sockets using different mountpoints for

shoutcast source clients. The shoutcast-mount outside of a listen-socket group is the global setting of the mountpoint to use.

shoutcast-compat

This optional flag will indicate that this port will operate in 'shoutcast-compatibility' mode. Due to major differences in the source client connection protocol, if you wish to use any of the shoutcast DJ tools, you will need to configure at least one socket as shoutcast-compatible. Note that when in this mode, only source clients (and specifically shoutcast source clients) will be able to attach to this port. All listeners may connect to any of the ports defined without this flag. Also, for proper Shoutcast DSP compatibility, you must define a listen socket with a port one less than the one defined as 'shoutcast-compat'. This means if you define 8001 as shoutcast-compat, then you will need to define a listen port of 8000 and it must not also be defined as shoutcast-compat. See the example config file in the distribution for more info.

Relaying Streams

This section contains the servers relay settings. The relays are implemented using a pull system where the receiving server connects as if its a listener to the sending server. There are two types of relay setups: a "Master server relay" or a "Specific Mountpoint relay."
Master Relay

A Master server relay is only supported between icecast2 servers and is used to relay a number of mountpoints from a remote icecast2 server.

<master-server>127.0.0.1</master-server>
<master-server-port>8001</master-server-port>
<master-update-interval>120</master-update-interval>
<master-username>relay</master-username>
<master-password>hackme</master-password>
<relays-on-demand>0</relays-on-demand>

The following diagram shows the basics of using a Master relay. Please note that the slave is configured with the <master-server>, <master-server-port>, etc settings and the master is the icecast server from which the slave will pull mountpoints and relay them. Using a Master server relay, all non-hidden mountpoints on the master can be relayed using this mechanism.

A server is configured as a Master Server relay by specifying the <master-server>, <master-server-port>,<master-update-interval>,<master-password> values in the config file. The server that is being relayed does not need any special configuration.

master-server

This is the IP for the server which contains the mountpoints to be relayed (Master Server).

master-server-port

This is the TCP Port for the server which contains the mountpoints to be relayed (Master Server).

master-update-interval

The interval (in seconds) that the Relay Server will poll the Master Server for any new mountpoints to relay.

master-username

This is the relay username on the master server. It is used to query the server for a list of mountpoints to relay. If not specified then 'relay' is used.

master-password

This is the relay password on the Master server. It is used to query the server for a list of mountpoints to relay.

relays-on-demand

Global on-demand setting for relays. Because you do not have individual relay options when using a master server relay, you still may want those relays to only pull the stream when there is at least one listener on the slave. The typical case here is to avoid surplus bandwidth costs when no one is listening.

Specific Mountpoint Relay

If only specific mountpoints need to be relayed, then you can configure Icecast with a "Specific Mountpoint Relay". Note that the relaying Icecast is configured with the <relay> settings and will pull the specified mountpoint(s) and relay them to the listeners. Using a Specific Mountpoint Relay, only those mountpoints specified will be relayed.

Specific Mountpoint Relays can be configured to relay from an Icecast 2 server, as well as Icecast 1.x and Shoutcast. A server is configured as a Specific Mountpoint Server relay by specifying a <relay> XML chunk in the config file for each mountpoint to be relayed. The server that is being relayed does not need any special configuration.

```
<relay>
    <server>127.0.0.1</server>
    <port>8001</port>
    <mount>/example.ogg</mount>
    <local-mount>/different.ogg</local-mount>
    <username>joe</username>
    <password>soap</password>
    <relay-shoutcast-metadata>0</relay-shoutcast-metadata>
    <on-demand>1</on-demand>
</relay>
```

server

This is the IP for the server which contains the mountpoint to be relayed.

port

This is the TCP Port for the server which contains the mountpoint to be relayed.

mount

The mountpoint located on the remote server. If you are relaying a shoutcast stream, this should be a '/' or '/;name'.

local-mount

The name to use for the local mountpoint. This is what the mount will be named on the relaying server. By default the remote mountpoint name is used.

username

The source of the relay may require authentication itself, if so state the username here.

password

The source of the relay may require authentication itself, if so state the password here.

relay-shoutcast-metadata

If you are relaying a Shoutcast stream, you may want to specify this indicator to also relay the metadata (song titles) that are part of the Shoutcast data stream (1=enabled, 0=disabled). By default this is enabled but it is up to the remote server on whether it sends any.

on-demand

An on-demand relay will only retrieve the stream if there are listeners requesting the stream. 1=enabled, 0=disabled (default is <relays-on-demand>). This is useful in cases where you want to limit bandwidth costs when no one is listening.

Mount Specific Settings

```
<mount>
    <mount-name>/example-complex.ogg</mount-name>
    <username>othersource</username>
    <password>hackmemore</password>
    <max-listeners>1</max-listeners>
    <max-listener-duration>3600</max-listener-duration>
    <dump-file>/tmp/dump-example1.ogg</dump-file>
    <intro>/intro.ogg</intro>
    <fallback-mount>/example2.ogg</fallback-mount>
    <fallback-override>1</fallback-override>
    <fallback-when-full>1</fallback-when-full>
    <charset>ISO8859-1</charset>
    <public>1</public>
    <stream-name>My audio stream</stream-name>
    <stream-description>My audio description</stream-description>
    <stream-url>http://some.place.com</stream-url>
    <genre>classical</genre>
    <bitrate>64</bitrate>
    <type>application/ogg</type>
    <subtype>vorbis</subtype>
    <hidden>1</hidden>
    <burst-size>65536</burst-size>
    <mp3-metadata-interval>4096</mp3-metadata-interval>
    <authentication type="htpasswd">
        <option name="filename" value="myauth"/>
        <option name="allow_duplicate_users" value="0"/>
    </authentication>
    <on-connect>/home/icecast/bin/source-start</on-connect>
    <on-disconnect>/home/icecast/bin/source-end</on-disconnect>
</mount>
```

This section contains the settings which apply only to a specific mountpoint and applies to an incoming stream whether it is a relay or a source client. The purpose of the mount definition is to state certain information that can override either global/default settings or settings provided from the incoming stream.

A mount does not need to be stated for each incoming source although you may want to specific certain settings like the maximum number of listeners or a mountpoint specific username/password. As a general rule, only define what you need to but each mount definition needs at least the mount-name. Changes to most of these will apply across a configuration file re-read even on active streams, however some only apply when the stream starts or ends.

mount-name

The name of the mount point for which these settings apply.

username

An optional value which will set the username that a source must use to connect using this mountpoint.

password

An optional value which will set the password that a source must use to connect using this mountpoint.

max-listeners

An optional value which will set the maximum number of listeners that can be attached to this mountpoint.

max-listener-duration

An optional value which will set the length of time a listener will stay connected to the stream. An auth component may override this.

dump-file

An optional value which will set the filename which will be a dump of the stream coming through on this mountpoint.

intro

An optional value which will specify the file those contents will be sent to new listeners when they connect but before the normal stream is sent. Make sure the format of the file specified matches the streaming format. The specified file is appended to webroot before being opened.

fallback-mount

This optional value specifies a mountpoint that clients are automatically moved to if the source shuts down or is not streaming at the time a listener connects. Only one can be listed in each mount and should refer to another mountpoint on the same server that is streaming in the same streaming format.

If clients cannot fallback to another mountpoint, due to a missing fallback-mount or it states a mountpoint that is just not available, then those clients will be disconnected. If clients are falling back to a mountpoint and the fallback-mount is not actively streaming but defines a fallback-mount itself then those clients may be moved there instead. This multi-level fallback allows clients to cascade several mountpoints.

A fallback mount can also state a file that is located in webroot. This is useful for playing a pre-recorded file in the case of a stream going down. It will repeat until either the listener disconnects or a stream comes back available and takes the listeners back. As per usual, the file format should match the stream format, failing to do so may cause problems with playback.

Note that the fallback file is not timed so be careful if you intend to relay this. They are fine on slave streams but don't use them on master streams, if you do then the relay will consume stream data at a faster rate and the

listeners on the relay would eventually get kicked off.

fallback-override

When enabled, this allows a connecting source client or relay on this mountpoint to move listening clients back from the fallback mount.

fallback-when-full

When set to 1, this will cause new listeners, when the max listener count for the mountpoint has been reached, to move to the fallback mount if there is one specified.
no-yp (deprecated)

Setting this option prevents this mountpoint from advertising on YP. The default is 0 so YP advertising can occur however you may want to prevent it here if you intend listeners to connect to a local relay instead. Deprecated option, replaced by <public>

charset

For non-Ogg streams like MP3, the metadata that is inserted into the stream often has no defined character set. We have traditionally assumed UTF8 as it allows for multiple language sets on the web pages and stream directory, however many source clients for MP3 type streams have assumed Latin1 (ISO 8859-1) or leave it to whatever character set is in use on the source client system.

This character mismatch has been known to cause a problem as the stats engine and stream directory servers want UTF8 so now we assume Latin1 for non-Ogg streams (to handle the common case) but you can specify an alternative character set with this option.

The source clients can also specify a charset= parameter to the metadata update URL if they so wish.

public

The default setting for this is -1 indicating that it is up to the source client or relay to determine if this mountpoint should advertise. A setting of 0 will prevent any advertising and a setting of 1 will force it to advertise. If you do force advertising you may need to set other settings listed below as the YP server can refuse to advertise if there is not enough information provided.
stream-name

Setting this will add the specified name to the stats (and therefore YP) for this mountpoint even if the source client/relay provide one.

stream-description

Setting this will add the specified description to the stats (and therefore YP) for this mountpoint even if the source client/relay provide one.

stream-url

Setting this will add the specified URL to the stats (and therefore YP) for this mountpoint even if the source client/relay provide one. The URL is generally for directing people to a website.

genre

Setting this will add the specified genre to the stats (and therefore YP) for this mountpoint even if the source client/relay provide one. This can be anything be using certain key words can help searches in the YP directories.

bitrate

Setting this will add the specified bitrate to the stats (and therefore YP) for this mountpoint even if the source client/relay provide one. This is stated in kbps.

type

Setting this will add the specified mime type to the stats (and therefore YP) for this mountpoint even if the source client/relay provide one. It is very unlikely that this will be needed.

subtype

Setting this will add the specified subtype to the stats (and therefore YP) for this mountpoint. The subtype is really to help the YP server to identify the components of the type. An example setting is vorbis/theora do indicate the codecs in an Ogg stream

burst-size

This optional setting allows for providing a burst size which overrides the default burst size as defined in limits. The value is in bytes.

mp3-metadata-interval

This optional setting specifies what interval, in bytes, there is between metadata updates within shoutcast compatible streams. This only applies to new listeners connecting on this mountpoint, not existing listeners falling back to this mountpoint. The default is either the hardcoded server default or the value passed from a relay.

hidden

Enable this to prevent this mount from being shown on the xsl pages. This is mainly for cases where a local relay is configured and you do not want the source of the local relay to be shown

authentication

This specifies that the named mount point will require listener authentication. Currently, we only support a file-based authentication scheme (type=htpasswd). Users and encrypted password are placed in this file (separated by a :) and all requests for this mountpoint will require that a user and password be supplied for authentication purposes. These values are passed in via normal HTTP Basic Authentication means (i.e. http://user:password@stream:port/mountpoint.ogg). Users and Passwords are maintained via the web admin interface. A mountpoint configured with an authenticator will display a red key next to the mount point name on the admin screens. You can read more about listener authentication here.

on-connect

State a program that is run when the source is started. It is passed a parameter which is the name of the mountpoint that is starting. The processing of the stream does not wait for the script to end. This option is not available on win32

on-disconnect

State a program that is run when the source ends. It is passed a parameter which is the name of the mountpoint that has ended. The processing of the stream does not wait for the script to end. This option is not available on win32

Path Settings

```
<paths>
    <basedir>./</basedir>
    <logdir>./logs</logdir>
    <pidfile>./icecast.pid</pidfile>
    <webroot>./web</webroot>
    <adminroot>./admin</adminroot>
    <allow-ip>/path/to/ip_allowlist</allow-ip>
    <deny-ip>/path_to_ip_denylist</deny-ip>
    <alias source="/foo" dest="/bar"/>
</paths>
```

This section contains paths which are used for various things within icecast. All paths (other than any aliases) should not end in a '/'.

basedir

This path is used in conjunction with the chroot settings, and specified the base directory that is chrooted to when the server is started. This feature is not supported on win32.

logdir

This path specifies the base directory used for logging. Both the error.log and access.log will be created relative to this directory.

pidfile

This pathname specifies the file to write at startup and to remove at normal shutdown. The file contains the process id of the icecast process. This could be read and used for sending signals icecast.

webroot

This path specifies the base directory used for all static file requests. This directory can contain all standard file types (including mp3s and ogg vorbis files). For example, if webroot is set to /var/share/icecast2, and a request for http://server:port/mp3/stuff.mp3 comes in, then the file /var/share/icecast2/mp3/stuff.mp3 will be served.

adminroot

This path specifies the base directory used for all admin requests. More specifically, this is used to hold the XSLT scripts used for the web-based admin interface. The admin directory contained within the icecast distribution contains these files.

allow-ip

If specified, this specifies the location of a file that contains a list of IP addresses that will be allowed to connect to icecast. This could be useful in cases where a master only feeds known slaves. The format of the file is simple, one IP per line.

deny-ip

If specified, this specifies the location of a file that contains a list of IP addressess that will be dropped immediately. This is mainly for problem clients when you have no access to any firewall configuration. The format of the file is simple, one IP per line.

alias source="/foo" dest="/bar"

Aliases are used to provide a way to create multiple mountpoints that refer to the same mountpoint.

Logging Settings

```
<logging>
    <accesslog>access.log</accesslog>
    <errorlog>error.log</errorlog>
    <playlistlog>playlist.log</playlistlog>
    <loglevel>4</loglevel> <-- 4 Debug, 3 Info, 2 Warn, 1 Error -->
</logging>
```

This section contains information relating to logging within icecast. There are two logfiles currently generated by icecast, an error.log (where all log messages are placed) and an access.log (where all stream/admin/http requests are logged).

Note that on non-win32 platforms, a HUP signal can be sent to icecast in which the log files are re-opened for appending giving the ability move/remove the log files.

accesslog

Into this file, all requests made to the icecast2 will be logged. This file is relative to the path specified by the <logdir> config value.

errorlog

All icecast generated log messages will be written to this file. If the loglevel is set too high (Debug for instance) then this file can grow fairly large over time. Currently, there is no log-rotation implemented.

playlistlog

Into this file, a log of all metadata for each mountpoint will be written. The format of the logfile will most likely change over time as we narrow in on a standard format for this. Currently, the file is pipe delimited. This option is optional and can be removed entirely from the config file.

logsize

This value specifies (in Kbytes) the maxmimum size of any of the log files. When the logfile grows beyond this value, icecast will either rename it to logfile.old, or add a timestamp to the archived file (if logarchive is enabled).

logarchive

If this value is set, then icecast will append a timestamp to the end of the logfile name when logsize has been reached. If disabled, then the default behavior is to rename the logfile to logfile.old (overwriting any previously saved logfiles). We disable this by default to prevent the filling up of filesystems for people who don't care (or know) that their logs are growing.

loglevel

Indicates what messages are logged by icecast. Log messages are categorized into one of 4 types, Debug, Info, Warn, and Error.

The following mapping can be used to set the appropriate value :

- loglevel = 4 - Debug, Info, Warn, Error messages are printed
- loglevel = 3 - Info, Warn, Error messages are printed
- loglevel = 2 - Warn, Error messages are printed
- loglevel = 1 - Error messages only are printed

Security Settings

```
<security>
    <chroot>0</chroot>
    <changeowner>
        <user>nobody</user>
        <group>nogroup</group>
    </changeowner>
</security>
```

This section contains configuration settings that can be used to secure the icecast server by performing a chroot to a secured location. This is currently not supported on win32.

chroot

An indicator which specifies whether a chroot() will be done when the server is started. The chrooted path is specified by the <basedir> configuration value.

changeowner

This section indicates the user and group that will own the icecast process when it is started. These need to be valid users on the system.

Admin Interface

Overview

This section contains information about the admin interface of icecast. Through this interface the user can manipulate many server features. From it you can gather statistics, move listeners from mountpoint to mountpoint, disconnect connected sources, disconnect connected listeners, and many other activities. Each function is enumerated here as well as an example usage of the function.

Each of these functions requires HTTP authentication via the appropriate username and password. For mount-specific functions, you may use either the <admin-username> and <admin-password> specified in the icecast config file, or the username and password specified for that mountpoint (if any). For general functions (not specific to a single mountpoint), you must use the admin username and password. It is also important to note that in all the examples 192.168.1.10 is used as the example host and 8000 is used as the example port for the icecast server.

Admin Functions (Mount Specific)

All these admin functions are mount specific in that they only apply to a particular mountpoint (as opposed to applying to the entire server). Each of these functions requires a mountpoint to be specified as input.

Metadata Update

Description:
This function provides the ability for either a source client or any external program to update the metadata information for a particular mountpoint.

Example:
http://192.168.1.10:8000/admin/metadata?mount=/mystream&mode=updinfo&song=ACDC+Back+In+Black

Fallback Update

Description:
This function provides the ability for either a source client or any external program to update the "fallback mountpoint" for a particular mountpoint. Fallback mounts are those that are used in the even of a source client disconnection. If a source client disconnects for some reason that all currently connected clients are sent immediately to the fallback mountpoint.

Example:
http://192.168.1.10:8000/admin/fallbacks?mount=/mystream.ogg&fallback=/myfallback.ogg

List Clients

Description:
This function lists all the clients currently connected to a specific mountpoint. The results are sent back in XML form.

Example:
http://192.168.1.10:8000/admin/listclients?mount=/mystream.ogg

Move Clients (Listeners)

Description:
This function provides the ability to migrate currently connected listeners from one mountpoint to another. This

function requires 2 mountpoints to be passed in: mount (the *from* mountpoint) and destination (the *to* mountpoint). After processing this function all currently connected listeners on mount will be connected to destination. Note that the destination mountpoint must exist and have a sounce client already feeding it a stream.

Example:
http://192.168.1.10:8000/admin/moveclients?mount=/mystream.ogg&destination=/mynewstream.ogg

Kill Client (Listener)

Description:
This function provides the ability to disconnect a specific listener of a currently connected mountpoint. Listeners are identified by a unique id that can be retrieved by via the "List Clients" admin function. This id must be passed in to the request. After processing this request, the listener will no longer be connected to the mountpoint.

Example:
http://192.168.1.10:8000/admin/killclient?mount=/mystream.ogg&id=21

Kill Source

Description:
This function will provide the ability to disconnect a specific mountpoint from the server. The mountpoint to be disconnected is specified via the variable "mount".

Example:
http://192.168.1.10:8000/admin/killsource?mount=/mystream.ogg

Admin Functions (General)

Stats

Description:
This admin function provides the ability to query the internal statistics kept by the icecast server. Almost all information about the internal workings of the server such as the mountpoints connected, how many client requests have been served, how many listeners for each mountpoint, etc, are available via this admin function.
Note that this admin function can also be invoked via the http://server:port/admin/stats.xml syntax, however this syntax should not be used and will eventually become deprecated.

Example:
http://192.168.1.10:8000/admin/stats

List Mounts

Description:
This admin function provides the ability to view all the currently connected mountpoints.

Example:
http://192.168.1.10:8000/admin/listmounts

Web-Based Admin Interface

As an alternative to manually invoking these URLs, a web-based admin interface was developed. This interface provides the same functions that were identified and described above but presents them in a little nicer way. The Web-Based Admin Interface to icecast is shipped with icecast provided in the "admin" directory and comes ready to use. All the user needs to do is set the path to this directory in the config file via the <adminroot> config variable.

The Web-Based Admin Interface is a series of XSLT files which are used to display all the XML obtained via the URL admin interface. This can be changed and modified to suit the user's need. Knowledge of XSLT and transformations from XML to HTML are required in order to make changes to these scripts.

The main URL for the Web-Based Admin Interface is

http://192.168.1.10:8000/admin/stats.xsl

From this URL all of the other admin functions can be exercised.

Modification of existing XSLT transforms in /admin is allowed, but new files cannot be created here. Creation of new XSLT transforms as well as modification of existing transforms is allowed in /web. These work using the document returned by /admin/stats.xml. To see the XML document that is applied to each admin XSLT, just change the .xsl to .xml in your request (i.e. /admin/listclients.xml). You can then code your XSLT transform accordingly.

Server Statistics

Overview

This section contains information about the server statistics available from icecast. An example stats XML tree will be shown and each element will be described. The following example stats tree will be used:

```
<?xml version="1.0"?>
<icestats>
    <client_connections>13</client_connections>
    <connections>14</connections>
    <source_connections>1</source_connections>
    <sources>1</sources>
    <source mount="/test.ogg">
        <artist></artist>
        <audio_info>ice-samplerate=32000;ice-bitrate=Quality -1;ice-channels=1</audio_info>
        <ice-bitrate>Quality -1</ice-bitrate>
        <ice-channels>1</ice-channels>
        <ice-samplerate>32000</ice-samplerate>
        <listeners>0</listeners>
        <public>0</public>
        <title></title>
        <type>Ogg Vorbis</type>
    </source>
</icestats>
```

General Statistics

client-connections

Client connections are basically anything that is not a source connection. These include listeners (not concurrent, but cumulative), any admin function accesses, and any static content (file serving) accesses.

source-connections

Source connections are the number of times (cumulative not currently connected) a source has connected to icecast.

connections

The total of client + source connections.

sources

The total of currently connected sources (mountpoints).

Source-Specific Statistics

artist

Artist of the current song (metadata set by source client).

title

Title of the current song (metadata set by source client).

audio-info

Information about the bitrate/samplerate/quality of the stream (set by source client). Also used for YP entries.

ice-bitrate
ice-samplerate
ice-channels

Information about the bitrate/samplerate/quality of the stream (set by source client).

listeners

The number of currently connected listeners.

public

Flag that indicates whether this mount is being listed on a YP (sey by source client).

type

Media type of the stream.

Relaying

Overview

Relaying is the process by which one server mirrors one or more streams from a remote server. The servers need not be of the same type (i.e. icecast can relay from Shoutcast). Relaying is used primarily for large broadcasts that need to distribute listening clients across multiple physical machines.

Type Of Relays

There are two types of relays that icecast supports. The first type is when both master and slave servers are icecast2 servers. In this case, a "master-slave" relay can be setup such that all that needs to be done is configure the slave server with the connection information (serverip:port) of the master server and the slave will mirror all mountpoints on the master server. The slave will also periodically check the master server to see if any new mountpoints have attached and if so will relay those as well. The second type of relay is a "single-broadcast" relay. In this case, the slave server is configured with a serverip+port+mount and only the mountpoint specified is relayed. In order to relay a broadcast stream on a Shoutcast server, you must use the "single-broadcast" relay and specify a mountpoint of "/".

Setting Up A Master-Slave Relay

In order to setup a relay of this type both servers (the one you wish to relay and the one doing the relaying) need to be icecast2 servers. The following configuration snippet is used as an example:

```
<master-server>192.168.1.11</master-server>
<master-server-port>8001</master-server-port>
<master-update-interval>120</master-update-interval>
<master-password>hackme</master-password>
```

In this example, this configuration is setup in the server which will be doing the relaying (slave server). The master server in this case need not be configured (and actually is unaware of the relaying being performed) as a relay. When the slave server is started, it will connect to the master server located at 192.168.1.11:8001 and will begin to relay all mountpoints connected to the master server. Additionally, every master-update-interval (120 seconds in this case) the slave server will poll the master server to see if any new mountpoints have connected, and if so, the slave server will relay those as well. Note that the names of the mountpoints on the slave server will be identical to those on the master server.

Setting Up A Single-Broadcast Relay

In this case, the master server need not be an icecast2 server. Supported master servers for a single-broadcast relay are Shoutcast, Icecast1.x, and of course Icecast2. The following configuration snippet is used as an example:

```
<relay>
    <server>192.168.1.11</server>
    <port>8001</port>
    <mount>/example.ogg</mount>
    <local-mount>/different.ogg</local-mount>
    <relay-shoutcast-metadata>0</relay-shoutcast-metadata>
</relay>
```

In this example, this configuration is also setup in the server which will be doing the relaying (slave server). The master server in this case need not be configured (and actually is unaware of the relaying being performed) as a

relay. When the slave server is started, it will connect to the master server located at 192.168.1.11:8001 and will begin to relay only the mountpoint specified (/example.ogg in this case). Using this type of relay, the user can override the local mountpoint name and make it something entirely different than the one on the master server. Additionally, if the server is a Shoutcast server, then the <mount> must be specified as /. And if you want the Shoutcast relay stream to have metadata contained within it (Shoutcast metadata is embedded in the stream itself) then the <relay-shoutcast-metadata> needs to be set to 1.

Listing In A YP Directories

Overview

A YP (Yellow Pages) directory is a listing of broadcast streams. Icecast2 has it own YP directory located at http://dir.xiph.org. Currently icecast2 can only be listed in an icecast2-supported YP directory. This means that you cannot list your stream in the Shoutcast YP directory.

In the icecast2 configuration file are all the currently available YP directory servers. Listing your stream in a YP is a combination of settings in the icecast configuration file and also in your source client.

Configuring Icecast2 For YP Support

First of all, icecast must have been built with YP support. This is automatically done if you have libcurl installed. If libcurl is not detected when icecats is compiled, then YP support is disabled.

If icecast has been built with YP support, then the following configuration options control the YP directory settings:

```
<directory>
   <yp-url-timeout>15</yp-url-timeout>
   <yp-url>http://dir.xiph.org/cgi-bin/yp-cgi</yp-url>
</directory>
```

Multiple directory XML chunks can be specified in order to be listed in multiple directories.

Configuring Your Source Client For YP Support

This is usually covered in the source client documentation. More specifically, the source client needs to provide the HTTP header ice-public:1 on connect in order to enable YP listing of the stream.

If a mountpoint is being listed on a YP, then you will see some additional statistics relating to the YP such as last-touch, currently-playing, etc.

Listener Authentication

Listener Authentication

Listener authentication is a feature of icecast which allows you to secure a certain mountpoint such that in order to listen, a listener must pass some verification test. With this feature, a simple pay-for-play operation (eg user/pass), or some filtering based on the listener connection can be performed. This section will show you the basics of setting up and maintaining this component.

To define listener authentication, a group of tags are specified in the <mount> group relating to the mountpoint. This means that authentication can apply to listeners of source clients or relays.

The following authentication mechanisms can apply to listeners

- HTPASSWD - lookup a named file for a matching username and password
- URL - issue web requests (eg PHP) to match authentication

The listener authentication within a specified mount in the icecast XML configuration can apply to either to a stream from a source client, relay or a webroot based file. They do apply to intro files or fallback streams.

HTPASSWD Listener Authentication

In order to use listener authentication, you MUST configure a mount specific option. This means that you have to provide a <mount> section in the main icecast config file. The following is an example :

```
<mount>
   <mount-name>/example.ogg</mount-name>
   <authentication type="htpasswd">
         <option name="filename" value="myauth"/>
         <option name="allow_duplicate_users" value="0"/>
   </authentication>
</mount>
```

To support listener authentication you MUST provide at a minimum <mount-name> and <authentication>. The mount-name is the name of the mountpoint that you will use to connect your source client with and authentication configures what type of icecast2 authenticator to use. Currently, only a single type "htpasswd" is implemented. New authenticators will be added later. Each authenticator has a variable number of options that are required and these are specified as shown in the example. The htpasswd authenticator requires a few parameters. The first, filename, specifies the name of the file to use to store users and passwords. Note that this file need not exist (and probably will not exist when you first set it up). Icecast has built-in support for managing users and passwords via the web admin interface. More on this later in this section. The second option, allow_duplicate_users, if set to 0, will prevent multiple connections using the same username. Setting this value to 1 will enable mutltiple connections from the same username on a given mountpoint. Note there is no way to specify a "max connections" for a particular user.

Icecast supports a mixture of streams that require listener authentication and those that do not. Only mounts that are named in the config file can be configured for listener authentication.

Configuring Users And Passwords

Once the appropriate entries are made to the config file, connect your source client (using the mountpoint you named in the config file). To configure users and passwords for this stream you must use the web-based admin interface. Navigate to http://server:ip/admin/stats.xsl to begin. If you have configured everything properly, you

should see this page (Icecast Status Page): http://server:ip/admin/stats.xsl

You will see a lock in front of all mountpoint configured for listener authentication. Also note that this page will only show CONNECTED mountpoints.

To manage users and passwords for this mountpoint, click on the lock or follow the "Manage Authentication" link. You should see this page (Show defined Users).

This screen will show all the users configured for this mountpoint. Adding users is as simple as entering a username and password in the fields and clicking "Add New User". Note that usernames MUST be unique and there are NO restrictions on passwords. You can delete users by clicking the appropriate delete link next to each user.

Finishing It All Off

Ok, so you've created your users, and you have everything setup properly, how do your users login ? Well, we've provided a simple login form that you can use for this purpose. This page (Authorization Page) http://server:port/auth.xsl will bring up a form that users can use to enter their username and password.

This page will serve a m3u with the username and password and in most cases should open the correct media player and begin playing your stream

URL

Authenticating listeners via the URL method involves icecast, when a listener connects, issuing requests to a web server and checking the response headers. If a certain header is sent back then the listener connecting is allowed to continue, if not, an error is sent back to the listener.

The URLs specified will invoke some web server scripts like PHP to do any work that they may choose to do. All that is required of the scripting language is that POST information can be handled and response headers can be sent back. libcurl is used for the requesting so https connections may be possible, but be aware of the extra overhead involved.

The useragent sent in each curl request will represent the icecast server version. The response headers will depend on whether the listener is to be accepted. In the case of rejection, a response header

icecast-auth-message: reason

should also be returned for placing in the log files.

In order to use URL based listener authentication, you MUST configure a mount specific option. This means that you have to provide a <mount> section in the main icecast config file. The following shows the list of options available :

```
<mount>
  <mount-name>/example.ogg</mount-name>
  <authentication type="url">
      <option name="mount_add" value="http://myauthserver.com/stream_start.php"/>
      <option name="mount_remove" value="http://myauthserver.com/stream_end.php"/>
      <option name="listener_add" value="http://myauthserver.com/listener_joined.php"/>
      <option name="listener_remove" value="http://myauthserver.com/listener_left.php"/>
      <option name="username" value="user"/>
      <option name="password" value="pass"/>
      <option name="auth_header" value="icecast-auth-user: 1"/>
```

```
        <option name="timelimit_header" value="icecast-auth-timelimit:"/>
    </authentication>
</mount>
```

The options are described below in more detail, each of which is optional, but in each case, within the POST data, the value for each setting is encoded.

mount_add

This URL is for informing the auth server of a stream starting. No listener information is passed for this, but can be used to initialise any details the auth server may have.

POST details are

action=mount_add&mount=/live&server=myserver.com&port=8000

Here the details indicate the server name (<hostname>) and mountpoint starting up

mount_remove

This URL is for informing the auth server of a stream finishing, like the start option, no listener details are passed.

POST details are

action=mount_remove&mount=/live&server=myserver.com&port=8000

like the start option, server name and mountpoint are provided

listener_add

This is most likely to be used if anything. When a listener connects, before anything is sent back to them, this request is processed. The default action is to reject a listener unless the auth server sends back a response header which may be stated in the 'header' option

POST details are

action=listener_add&server=myserver.com&port=8000&client=1&mount=/live&user=&pass=&ip=127.0.0.1&agent="My%20player"

There are more details with this, client is the unique ID for the client within icecast, user and pass may be blank but come from the HTTP basic auth that the listener states, ip is the listeners IP address and agent is the Useragent from the listeners player.

The mount here (unlike the start/end options) states the requested url including any query parameters, so for instance the requested URL can be /stream.ogg&session=xyz, but note that each option data is escaped before being passed via POST

listener_remove

This URL is for when a listener connection closes.

POST details are

action=listener_remove&server=myserver.com&port=8000&client=1&mount=/live&user=&pass=&duration=3600

Again this is similar to the add option, the difference being that a duration is passed reflecting the number of seconds the listener was connected for

auth_header

The expected response header to be returned that allows the authencation to take place may be specified here. The default is

icecast-auth-user: 1

but it could can anything you like, for instance

HTTP 200 OK

timelimit_header

Listeners could have a time limit imposed on them, and if this header is sent back with a figure (which represents seconds) then that is how long the client will remain connected for.

A Note About Players And Authentication

We do not have an exaustive list of players that support listener authentication. We use standard HTTP basic authentication, and in general, many media players support this if they support anything at all. Winamp and Foobar2000 support HTTP basic authentication on windows, and XMMS supports it on unix platforms. Winamp/XMMS as least support the passing of query parameters, other players may also do

Win32 Specific Documentation

The win32 port of icecast2 is simply a UI framework around the core icecast2 server. The win32 version of icecast2 directly uses the main executable of icecast (statically included) and simply provides a GUI interface to icecast2.

Most of the features of icecast2 are available in the win32 port.

Server Status Tab

The server status tab contains information regarding statistics that are global to the server. There are two types of statistics in icecast2: source level and global statistics. Global statistics are cumulative stats from all sources offered by the server. Source level statistics are stats which apply only to a single source attached to the server.

Examples of global statistics are:

The number of current sources connected
The number of sources that have attempted connections
Total number of attempted connections to the server

The Server Status tab contains at a minimal the global stats for the server. Additionally, you may add source specific stats to this tab. The intent is to provide a single "dashboard view" of what's going on in the server. To add source statistics to the Server Status tab, see the section on the Stats tab.

Adding stats to the window title.

Any stat that is contained on the Server Status tab can be displayed as the icecast2 window title. This provides yet another mechanism by which you can view activities on the server. To enable this feature, right click on any stat in the Server Status tab.

Removing source level stats from the Server Status Tab.

To remove a source level stat that you have inserted onto the Server Status Tab, simple right click that statistic and select "Delete from Global Stats". The stat will be deleted from the Server Status tab, but will still remain on the source level Stats tab.

Editing The Icecast Config File

Editing the icecast2 configuration file is a very simple process. For a description of what each field means, see the main icecast documenation. Changes to the icecast2 configuration can only be done while the server is stopped. To edit the current server configuration file, select "Configuration/Edit Configuration" from the main menu.

Stats Tab

The stats tab contains a view of all the connected mountpoints and the statistics that go along with them. Each connected mountpoint is displayed in the left pane of the window, and all stats for the selected mountpoint are displayed in the right pane of the window.

Server Status Tab

Overview

The server status tab contains information regarding statistics that are global to the server. There are two types of statistics in icecast2, source level and global statistics. Global statistics are those that are accumilations of stats from all sources offered by the server. Source level statistics are stats which apply only to a single source attached to the server.

Examples of global statistics are :

The number of current sources connected
The number of sources that have attempted connections
Total number of attempted connections to the server

The Server Status tab contains at a minimal the global stats for the server. Additionally, you may add source specific stats to this tab. The intent is to provide a single "dashboard view" of what's going on in the server.

Adding Stats To The Window Title

Any stat that is contained on the Server Status tab can be displayed as the icecast2 window title. This provides yet another mechanism by which you can view activities on the server. To enable this feature, right click on any stat in the Server Status tab.

Removing Source Level Stats From The Server Status Tab

To remove a source level stat that you have inserted onto the Server Status Tab, simple right click that statistic and select "Delete from Global Stats". The stat will be deleted from the Server Status tab, but will still remain on the source level Stats tab.

Glossary

source client

A source client is an external program which is responsible for sending content data to icecast. Some source clients that support icecast2 are Oddcast, ices2, ices0.3, and DarkIce.

slave server (Relay)

The slave server in a relay configuration is the server that is pulling the data from the master server. It acts as a listening client to the master server.

master server (Relay)

The master server in a relay configuration is the server that has the stream that is being relayed.

mountpoint

A mountpoint is a resource on the icecast server that represents a single broadcast stream. Mountpoints are named similar to files (/mystream.ogg, /mymp3stream). When listeners connect to icecast2, they must specify the mountpoint in the request (i.e. http://192.168.1.10:8000/mystream.ogg). Additionally, source clients must specify a mountpoint when they connect as well. Statistics are kept track of by mountpoint. Mountpoints are a fundamental aspect of icecast2 and how it is organized.

fallback mountpoint

A fallback mountpoint is configured with a parent mountpoint. In the event of the parent mountpoint losing connection with icecast, Icecast will then move all clients currently connected to the now defunct mountpoint to it's fallback mountpoint.

Frequently Asked Questions

General Questions

What is Icecast?

Icecast, the project, is a collection of programs and libraries for streaming audio over the Internet. This includes:

- Icecast, a program that streams audio data to listeners
- Libshout, a library for communicating with Icecast servers
- IceS, a program that sends audio data to Icecast servers

A source client is an external program which is responsible for sending content data to icecast. Some source clients that support icecast2 are Oddcast, ices2, ices0.3, and DarkIce.

What is icecast, the program?

icecast streams audio to listeners, and is compatible with Nullsoft`s Shoutcast.

What is libshout ?

From the README:

libshout is a library for communicating with and sending data to an icecast server. It handles the socket connection, the timing of the data, and prevents bad data from getting to the icecast server.

What is IceS?

IceS is a program that sends audio data to an icecast server to broadcast to clients. IceS can either read audio data from disk, such as from Ogg Vorbis files, or sample live audio from a sound card and encode it on the fly.

How can I view the stream status page?

Check your icecast configuration file for an element called <webroot>. This directory contains web stuff. In it, place a file called "status.xsl" that transforms an XML file containing stream data into a web page (either XHTML or HTML).

There are sample XSL stylesheets available in icecast/web/ in the CVS distribution of icecast.

In addition, the web directory can hold multiple status transforms, if you can't decide which one you want.

What can I use to listen to an Icecast stream?

We maintain a list of Icecast-compatible audio players at http://www.icecast.org/

Ezstream

About Ezstream

Ezstream is a command line source client for Icecast media streaming servers. It began as the successor of the old "shout" utility, and has since gained a lot of useful features.

In its basic mode of operation, it streams media files or data from standard input without re-encoding and thus requires only very little CPU resources. It can also use various external decoders and encoders to re-encode from one format to another, and stream the result to an Icecast server. With re-encoding enabled, ezstream is a very flexible source client.

Supported media formats for streaming are MP3, Ogg Vorbis and Ogg Theora. Ezstream natively supports metadata in MP3 (ID3v1 only) and Ogg Vorbis, or many more formats when it is built with the TagLib option.

Prerequisites

Ezstream depends on:

- libshout 2.2.x
(http://www.icecast.org/)

- libxml 2.x
(http://xmlsoft.org/)

Ezstream optionally uses:

For reading metadata from Ogg Vorbis files:
- TagLib 1.x (1.4 or newer recommended, will be used via the libtag_c wrapper)
(http://developer.kde.org/~wheeler/taglib.html)

OR:

- libvorbis 1.x
(http://www.vorbis.com)

Using TagLib is recommended, as it allows ezstream to read metadata from many additional media file types.

For basic non-ASCII charset support in metadata and filenames:
- Libiconv, if iconv() is not available in the system libc.
(http://www.gnu.org/software/libiconv/)

Installation

The ezstream software uses the GNU auto-tools to configure, build and install on a variety of systems. Aside from the standard autoconf options of the configure script, a couple of additional options are available:

--enable-examplesdir=DIR
example configuration files installation directory
(default: DATADIR/examples/ezstream)

--enable-debug enable memory debugging (default: no)

--with-libvorbis=PFX prefix where the Vorbis library header files and library are installed (default: autodetect)

--with-libvorbis-includes=DIR
directory where Vorbis library header files are installed (optional)

--with-libvorbis-libs=DIR
directory where the Vorbis libraries are installed
(optional)

--with-taglib=PFX prefix where the TagLib header files and library are installed (default: autodetect)

--with-taglib-includes=DIR
directory where TagLib header files are installed
(optional)

--with-taglib-libs=DIR directory where TagLib is installed (optional)

The compilation and installation process then boils down to the usual

```
$ ./configure --help | less        # Skim over the available options
$ ./configure [options] && make && [sudo] make install
# Configure, build and install
# [as root] the software
```

If this procedure is unfamiliar to you, please consult the INSTALL file for more detailed instructions.

When the configuration keeps failing despite having all dependencies installed, take note of the more verbose error messages in config.log. If necessary, it is possible to directly customize many build flags through environment variables. See the "influential environment variables" list in the --help output.

Usage

Once ezstream is successfully installed, type "man ezstream" (without quotes) on the command line for a comprehensive manual. This distribution package also comes with example configuration files that can be used as a guide to configure ezstream.

Note that all by itself, ezstream is not particularly useful. It requires a running Icecast server to stream to, which then relays the stream to many listeners. If this comes as a surprise, browse to http://www.icecast.org/ for a lot more information, resources, and other source clients.

External Decoders/Encoders

Ezstream should be able to work with any media decoder and encoder that fulfills the following requirements:

1. It needs to be executable on the command line and not require a graphical display to function.
2.1. A decoder needs to be capable of sending RAW data to standard output.
2.2. An encoder needs to be capable of reading RAW data from standard input.
2.3. A combined de-/encoder needs to be capable of sending media data that can be streamed to standard output.

Media formats that ezstream does not support directly are passed through unaltered. Whether they work or not depends on the level of support offered by the version of libshout ezstream is linked with.

The following incomplete list of programs shows a few that are known to work.
These are also used in the example configuration files:

MP3
- Decoder: madplay (http://mad.sf.net/)
- Encoder: lame (http://lame.sf.net/)

Ogg Vorbis:
- Decoder: oggdec
- Encoder: oggenc
Both utilities are in the vorbis-tools package (http://www.vorbis.com/).

FLAC:
- Decoder: flac (http://flac.sf.net/)
- Encoder: (None. Not supported by libshout at the time of writing, and thus cannot be used by ezstream.)

Ogg Theora:
- Decoder/Encoder: ffmpeg2theora (http://v2v.cc/~j/ffmpeg2theora/)

Operating System Specific Notes

Ezstream and SunPRO cc/c99 on Solaris:

Ezstream may not build with SunPRO cc/c99 "out of the box" if a threaded libshout was built with gcc. This known issue results in the following error message from the linker:

ld: fatal: option -h and building a dynamic executable are incompatible

This is related to gcc and GNU ld using different compiler/linker flags, related to POSIX threads, than the SunPRO compilers. These are being passed on to ezstream, where cc or c99 ultimately chokes on them.

Ezstream compiles with both cc/c99 and gcc if libshout was built with Sun's compiler. If libshout was built with gcc, compile ezstream with gcc as well.

Information About The Binary Ezstream Distribution For Windows

ZIP Archive Contents

File	Description
\COPYING.txt	License for using, modifying and distributing Ezstream.
\README.txt	This file.
\ezstream.1.pdf	The ezstream manual in PDF format.
\ezstream.exe	The ezstream executable file.
\ezstream-X.Y.Z.zip	The ezstream source distribution, from which ezstream.exe was built.
\examples*.xml	Ezstream example configuration files.
\examples*.sh	Example playlist and metadata scripts, for demonstrational purposes (these are shell scripts.)

Installation

As of version 0.3.0, ezstream no longer comes with an installer. If you have a previous version of ezstream installed, you might want to simply uninstall it.

The ezstream.exe file can be run from any location. To install it, simply copy it to a location of your choosing. To make it available anywhere on the system, add the installation folder to your PATH environment variable.

If it doesn't start, e.g. running
> ezstream.exe -h
does not show the command line help, the Microsoft Visual C++ 2008 runtime libraries are missing. Download and install vcredist_x86.exe, which can be easily found on www.microsoft.com via
http://www.google.com/search?q=vc+2008+redist

Limited Functionality

A few useful features are missing in the Windows version of ezstream:
- Runtime control via signals is not possible.
- Some useful external utilities (encoders, decoders) may not be available on Windows.

Source Code

Ezstream uses:
- zlib (http://www.zlib.net/, BSD-like license)
- libiconv (http://www.gnu.org/software/libiconv/, LGPL)
- libxml2 (http://xmlsoft.org/, BSD-like license)
- libogg, libvorbis, libvorbisfile (http://www.vorbis.com/, BSD-like license)
- libFLAC (http://flac.sourceforge.net/, BSD-like license)
- libtheora (http://www.theora.org/, BSD-like license)
- libshout (http://www.icecast.org, LGPL)
- TagLib (http://ktown.kde.org/~wheeler/taglib.html, LGPL)

These libraries are statically linked into the ezstream.exe file.

Ezstream itself is licensed under the GPL version 2 (see COPYING.txt for details.)

Support

For information on how to report issues with ezstream, please visit the ezstream home page at http://www.icecast.org/ezstream.php

Ezstream Metadata - Example XML Configuration File

```
<!--
    EXAMPLE: Ogg Vorbis stream WITHOUT reencoding, using an external playlist program and an external metadata program, with a custom format string.

  This example streams (only) Ogg Vorbis files that are provided by an external playlist script. The script must not return filenames of non-.ogg files. Metadata for this stream is not acquired from the media file itself but from another external program 'meta.sh'.
 -->
<ezstream>
  <url>http://localhost:8000/vorbis.ogg</url>
  <sourcepassword>hackme</sourcepassword>
  <format>VORBIS</format>
  <!-- The playlist program name is provided here: -->
  <filename>play.sh</filename>
  <!-- Tell ezstream that <filename/> is a playlist program: -->
  <playlist_program>1</playlist_program>
  <!-- Use the meta.sh script to get metadata -->
  <metadata_progname>meta.sh</metadata_progname>
  <!-- Set the metadata string according to this format: -->
  <metadata_format>@s@: @a@ - @t@</metadata_format>
  <!--
   The following settings are used to describe your stream to the server.
   It's up to you to make sure that the bitrate/quality/samplerate/channels
   information matches up with your input stream files.
  -->
  <svrinfoname>My Stream</svrinfoname>
  <svrinfourl>http://www.oddsock.org</svrinfourl>
  <svrinfogenre>RockNRoll</svrinfogenre>
  <svrinfodescription>This is a stream description</svrinfodescription>
  <svrinfobitrate>96</svrinfobitrate>
  <svrinfoquality>2.0</svrinfoquality>
  <svrinfochannels>2</svrinfochannels>
  <svrinfosamplerate>44100</svrinfosamplerate>
  <!-- Allow the server to advertise the stream on a public YP directory: -->
  <svrinfopublic>1</svrinfopublic>
</ezstream>
```

Ezstream Mp3 - Example XML Configuration File

```
<!--
  EXAMPLE: MP3 playlist stream WITHOUT reencoding

  This example streams a playlist that contains only MP3 files. No other file formats may be listed. Since ezstream will not be doing any reencoding, the resulting stream format (bitrate, samplerate, channels) will be of the respective input files.
 -->
<ezstream>
   <url>http://localhost:8000/stream</url>
   <sourcepassword>hackme</sourcepassword>
   <format>MP3</format>
   <filename>playlist.m3u</filename>
   <!-- Once done streaming playlist.m3u, exit: -->
   <stream_once>1</stream_once>
   <!--
      The following settings are used to describe your stream to the server.  It's up to you to make sure that the bitrate/samplerate/channels information matches up with your input stream files. Note that <svrinfoquality /> only applies to Ogg Vorbis streams.
    -->
   <svrinfoname>My Stream</svrinfoname>
   <svrinfourl>http://www.oddsock.org</svrinfourl>
   <svrinfogenre>RockNRoll</svrinfogenre>
   <svrinfodescription>This is a stream description</svrinfodescription>
   <svrinfobitrate>128</svrinfobitrate>
   <svrinfochannels>2</svrinfochannels>
   <svrinfosamplerate>44100</svrinfosamplerate>
   <!--
    Prohibit the server to advertise the stream on a public YP directory:
    -->
   <svrinfopublic>0</svrinfopublic>
</ezstream>
```

Ezstream Re-encode Mp3 - Example XML Configuration File

```
<!--
  EXAMPLE: MP3 stream using an external playlist program, WITH reencoding

   This example streams a playlist that may contain MP3, Ogg Vorbis and FLAC files. Ezstream will use external decoders to read the media files, and re-encode them to MP3 using the lame MP3 encoder. The output stream settings are controlled via the paramters to lame.
 -->
<ezstream>
   <url>http://localhost:8000/stream</url>
   <sourcepassword>hackme</sourcepassword>
   <!--
     Since the reencoding feature is enabled below, <format /> sets the output format of the stream.
    -->
   <format>MP3</format>
   <filename>playlist.pl</filename>
   <!--
      Indicate that <filename> contains an executable program or script, which prints a single line with a filename to standard output:
    -->
   <playlist_program>1</playlist_program>
   <!--
      The following settings are used to describe your stream to the server. It's up to you to make sure that the bitrate/samplerate/channels information matches up with your lame encoder settings below.
    -->
   <svrinfoname>My Stream</svrinfoname>
   <svrinfourl>http://www.oddsock.org</svrinfourl>
   <svrinfogenre>RockNRoll</svrinfogenre>
   <svrinfodescription>This is a stream description</svrinfodescription>
   <svrinfobitrate>128</svrinfobitrate>
   <svrinfochannels>2</svrinfochannels>
   <svrinfosamplerate>44100</svrinfosamplerate>
   <!--
    Prohibit the server to advertise the stream on a public YP directory:
   -->
   <svrinfopublic>0</svrinfopublic>
   <reencode>
      <!-- Enable the reencoding feature: -->
      <enable>1</enable>
      <!--
       Each <encdec /> element specifies a pair of programs to be used for decoding and encoding, respectively, and which file extension and output stream format they apply to.

       All the configuration of the output stream is usually done by using the appropriate command line parameters of the encoders in the <encode /> elements.

       New <encdec /> sections can be added for new input/output formats.

       Distorted audio, or audio playing at the wrong speed/pitch may be caused by conflicting sample rates in the various <decode /> and <encode /> sections, byte order (endianness) issues and mono input files. See the documentation on the various de-/encoders for the options that need to be used to create a consistent stream of raw samples.
      -->
      <encdec>
```

```
        <!-- Support for FLAC decoding: -->
        <format>FLAC</format>
        <match>.flac</match>
        <decode>flac -s -d --force-raw-format --sign=signed --endian=little -o - "@T@"</decode>
        <!-- <encode>Not supported Yet</encode> -->
      </encdec>
      <encdec>
        <!--
          Support for MP3 decoding via madplay, and encoding via LAME:
         -->
        <format>MP3</format>
        <match>.mp3</match>
        <!-- Note: madplay uses host byte order for raw samples. -->
        <decode>madplay -b 16 -R 44100 -S -o raw:- "@T@"</decode>
        <encode>lame --preset cbr 128 -r -s 44.1 --bitwidth 16 - -</encode>
      </encdec>
      <encdec>
        <!--
          Support for Vorbis decoding via oggdec, and encoding via oggenc:
         -->
        <format>VORBIS</format>
        <match>.ogg</match>
        <decode>oggdec -R -b 16 -e 0 -s 1 -o - "@T@"</decode>
        <encode>oggenc -r -B 16 -C 2 -R 44100 --raw-endianness 0 -q 1.5 -t "@M@" -</encode>
      </encdec>
    </reencode>
</ezstream>
```

Ezstream Vorbis - Example XML Configuration File

```
<!--
  EXAMPLE: Ogg Vorbis playlist stream WITHOUT reencoding

   This example streams a playlist that contains only Ogg Vorbis files. No other file formats may be listed. Since ezstream will not be doing any reencoding, the resulting stream format (quality/bitrate, samplerate, channels) will be of the respective input files.
 -->
<ezstream>
   <url>http://localhost:8000/vorbis.ogg</url>
   <sourcepassword>hackme</sourcepassword>
   <format>VORBIS</format>
   <filename>playlist.m3u</filename>
   <!-- For demonstrational purposes, explicitly set continuous streaming: -->
   <stream_once>0</stream_once>
   <!--
      The following settings are used to describe your stream to the server. It's up to you to make sure that the bitrate/quality/samplerate/channels information matches up with your input stream files.
    -->
   <svrinfoname>My Stream</svrinfoname>
   <svrinfourl>http://www.oddsock.org</svrinfourl>
   <svrinfogenre>RockNRoll</svrinfogenre>
   <svrinfodescription>This is a stream description</svrinfodescription>
   <svrinfobitrate>96</svrinfobitrate>
   <svrinfoquality>2.0</svrinfoquality>
   <svrinfochannels>2</svrinfochannels>
   <svrinfosamplerate>44100</svrinfosamplerate>
   <!-- Allow the server to advertise the stream on a public YP directory: -->
   <svrinfopublic>1</svrinfopublic>
</ezstream>
```

Ezstream Re-encode Vorbis - Example XML Configuration File

```
<!--
  EXAMPLE: Ogg Vorbis playlist stream WITH reencoding and random playback

   This example streams a playlist that may contain MP3, Ogg Vorbis and FLAC files. Ezstream will use external decoders to read the media files, and re-encode them to Ogg Vorbis using the oggenc encoder. The output stream settings are controlled via the paramters to oggenc.
 -->
<ezstream>
   <url>http://localhost:8000/vorbis.ogg</url>
   <sourcepassword>hackme</sourcepassword>
   <!--
     Since the reencoding feature is enabled below, <format /> sets the output format of the stream.
    -->
   <format>VORBIS</format>
   <filename>playlist.m3u</filename>
   <!-- Enable playlist shuffling: -->
   <shuffle>1</shuffle>
   <!--
      The file in <filename> is a regular playlist and not a program. For demonstrational purposes, explicitly state this here:
    -->
   <playlist_program>0</playlist_program>
   <!--
      The following settings are used to describe your stream to the server.  It's up to you to make sure that the bitrate/quality/samplerate/channels information matches up with your oggenc encoder settings below.
    -->
   <svrinfoname>My Stream</svrinfoname>
   <svrinfourl>http://www.oddsock.org</svrinfourl>
   <svrinfogenre>RockNRoll</svrinfogenre>
   <svrinfodescription>This is a stream description</svrinfodescription>
   <svrinfobitrate>88</svrinfobitrate>
   <svrinfoquality>1.5</svrinfoquality>
   <svrinfochannels>2</svrinfochannels>
   <svrinfosamplerate>44100</svrinfosamplerate>
   <!-- Allow the server to advertise the stream on a public YP directory: -->
   <svrinfopublic>1</svrinfopublic>
   <reencode>
      <!-- Enable the reencoding feature: -->
      <enable>1</enable>
      <!--
      Each <encdec /> element specifies a pair of programs to be used for decoding and encoding, respectively, and which file extension and output stream format they apply to.

      All the configuration of the output stream is usually done by using the appropriate command line parameters of the encoders in the <encode /> elements.

      New <encdec /> sections can be added for new input/output formats.

      Distorted audio, or audio playing at the wrong speed/pitch may be caused by conflicting sample rates in the various <decode /> and <encode /> sections, byte order (endianness) issues and mono input files. See the documentation on the various de-/encoders for the options that need to be used to create a consistent stream of raw samples.
      -->
```

```
    <encdec>
      <!-- Support for FLAC decoding: -->
      <format>FLAC</format>
      <match>.flac</match>
      <decode>flac -s -d --force-raw-format --sign=signed --endian=little -o - "@T@"</decode>
      <!-- <encode>Not supported Yet</encode> -->
    </encdec>
    <encdec>
      <!--
        Support for MP3 decoding via madplay, and encoding via LAME:
       -->
      <format>MP3</format>
      <match>.mp3</match>
      <!-- Note: madplay uses host byte order for raw samples. -->
      <decode>madplay -b 16 -R 44100 -S -o raw:- "@T@"</decode>
      <encode>lame --preset cbr 128 -r -s 44.1 --bitwidth 16 - -</encode>
    </encdec>
    <encdec>
      <!--
        Support for Vorbis decoding via oggdec, and encoding via oggenc:
       -->
      <format>VORBIS</format>
      <match>.ogg</match>
      <decode>oggdec -R -b 16 -e 0 -s 1 -o - "@T@"</decode>
      <encode>oggenc -r -B 16 -C 2 -R 44100 --raw-endianness 0 -q 1.5 -t "@M@" -</encode>
    </encdec>
  </reencode>
</ezstream>
```

Ezstream Re-encode Theora - Example XML Configuration File

```
<!--
  EXAMPLE: Ogg Theora playlist stream WITH reencoding and sequential playback

   This example streams a playlist that may contain .avi and MPEG files. Ezstream will use the ffmpeg2theora program to both decode and re-encode the video files to Ogg Theora. The output stream settings are controlled via the paramters to ffmpeg2theora.
 -->
<ezstream>
   <url>http://localhost:8000/theora.ogg</url>
   <sourcepassword>hackme</sourcepassword>
   <!--
     Since the reencoding feature is enabled below, <format /> sets the output format of the stream.
    -->
   <format>THEORA</format>
   <filename>playlist.m3u</filename>
   <!--
     Playlist shuffling is disabled, when the <shuffle /> element does not exist or is commented out:
    -->
   <!-- <shuffle>1</shuffle> -->
   <!--
      The following settings are used to describe your stream to the server.  It's up to you to make sure that the bitrate/quality/samplerate/channels information matches up with your oggenc encoder settings below.
    -->
   <svrinfoname>My Stream</svrinfoname>
   <svrinfourl>http://www.oddsock.org</svrinfourl>
   <svrinfogenre>Documentary</svrinfogenre>
   <svrinfodescription>This is a stream description</svrinfodescription>
   <svrinfobitrate>200</svrinfobitrate>
   <svrinfochannels>2</svrinfochannels>
   <svrinfosamplerate>44100</svrinfosamplerate>
   <!--
     A missing or commented out <svrinfopublic /> element means no advertising on YP for this broadcast:
    -->
   <!-- <svrinfopublic>1</svrinfopublic> -->
   <reencode>
      <!-- Enable the reencoding feature: -->
      <enable>1</enable>
      <!--
       Each <encdec /> element specifies a pair of programs to be used for decoding and encoding, respectively, and which file extension and output stream format they apply to.

       All the configuration of the output stream is usually done by using the appropriate command line parameters of the encoders in the <encode /> elements.

       New <encdec /> sections can be added for new input/output formats.

       Distorted audio, or audio playing at the wrong speed/pitch may be caused by conflicting sample rates in the various <decode /> and <encode /> sections, byte order (endianness) issues and mono input files. See the documentation on the various de-/encoders for the options that need to be used to create a consistent stream of raw samples.
       -->
      <encdec>
         <!--
```

Support for THEORA. Ogg Theora streams are created differently with ffmpeg2theora, which does the complete reencoding on its own. Therefore, we only supply a <decode /> element and let ezstream pass through the resulting stream unaltered to Icecast.

```
        -->
        <format>THEORA</format>
        <match>.avi</match>
        <decode>ffmpeg2theora -x 192 -y 128 -a 0 -v 4 --title "@M@" -o - "@T@"</decode>
      </encdec>
      <encdec>
        <format>THEORA</format>
        <match>.mpg</match>
        <decode>ffmpeg2theora -x 192 -y 128 -a 0 -v 4 --title "@M@" -o - "@T@"</decode>
      </encdec>
    </reencode>
</ezstream>
```

Ezstream Man Page

NAME

ezstream - source client for Icecast with external de-/encoder support

SYNOPSIS

ezstream [-hnqVv] -c configfile

DESCRIPTION

The ezstream utility is a source client for the Icecast media streaming server. In its basic mode of operation, it streams media files and data from standard input "as-is" — such as Ogg Vorbis, Ogg Theora and MP3 — to a server. It can also use various external decoders and encoders to re- encode from one format to another, and stream the result. The only requirement is that the external programs support writing to or reading from standard input, and can be used from the command line.

Command line parameters

-c configfile
Use the XML configuration in configfile. (Mandatory.)

-h Print a summary of available command line parameters with short descriptions and exit.

-n Normalize metadata strings by removing excess whitespaces.

-q Be more quiet. Suppress the output that external programs send to standard error.

-V Print the ezstream version number and exit.

-v Produce more verbose output from ezstream. Use twice for even more verbose output.

When the -q and -v parameters are provided simultaneously, an additional line of information about the currently streamed file — playlist posi- tion, approximate playing time and bit rate — is displayed.

Runtime control
On POSIX systems, ezstream offers limited runtime control via signals. By sending a signal to the ezstream process, e.g. with the kill(1) util- ity, a certain action will be triggered.

SIGHUP
Rereads the playlist file after the track that is currently streamed. If the playlist is not to be shuffled, ezstream attempts to find the previously streamed file and continue with the one fol- lowing it, or restarts from the beginning of the list otherwise.

SIGUSR1
Skips the currently playing track and moves on to the next in playlist mode, or restarts the current track when streaming a single file.

SIGUSR2
Triggers rereading of metadata for the stream by running the program or script specified in <metadata_progname/> (see below.) This is the only meaningful signal when streaming from standard input.

The ezstream utility uses a simple XML configuration file format. It has a tree-like structure and is made up of XML elements. Of all the possi- ble XML features, only regular elements that contain text or other ele- ments, and comments, appear in an ezstream configuration file.

Each element in the configuration file consists of a start tag, its con- tent and an end tag. For example:

<filename>playlist.m3u</filename>
<!-- XML comments look like this. -->

In this section, each available element is listed and described. Note that for this purpose, elements are introduced in their short, i.e. empty form. In the configuration file, they need to be used as start tag + content + end tag, like in the introductory example shown above.

Root element
<ezstream />
(Mandatory.) The configuration file's root element. It contains all other configuration elements.

Global configuration elements
Each of the global configuration elements have the <ezstream/> element as their parent.

<url />
(Mandatory.) Specifies the location and mount point of the Icecast server, to which the stream will be sent. The content must be of the form http://server:port/mountpoint For example:

<url>http://example.com:8000/stream.ogg</url>

<sourcepassword />
(Mandatory.) Sets the source password for authentication with the Icecast server.

<format />
(Mandatory.) This element has two different meanings, depending on whether re-encoding is enabled or not. It specifies the output format of the stream if re-encoding is enabled. Otherwise, it spec- ifies the input format of all input files. Recognized and supported values for output stream formats are VORBIS, MP3 and THEORA. Other values will be ignored and cause ezstream to simply pass through the data, which may or may not work.

<filename />
(Mandatory.) Set the path and name of a single media file, a playlist, the name of an external program (see below), or the key- word stdin for streaming from standard input. Playlists are recog- nized by their filename extension and end with either .m3u or .txt.

A playlist consists of filenames, one entry per line. Comments in playlists are introduced by a '#' sign at the beginning of a line and ignored by ezstream.

<playlist_program />
(Optional.) Set to 1 (one) to indicate that the file in <filename/> is actually an executable program or script. If set to 0 (zero), <filename/> content is assumed to be a media file, playlist file or the keyword stdin (the default).

See the SCRIPTING section for details on how the playlist program must behave.

<shuffle />
(Optional.) Set to 1 (one) to randomly shuffle the entries of the playlist specified in <filename/>. Files are played sequentially if set to 0 (zero) or when the <shuffle/> element is absent. This option will be ignored if <playlist_program/> is set to 1 (one.)

<metadata_progname />
(Optional.) Set the path and name of an executable program or script that should be used by ezstream to set the

metadata of the stream.

The program is automatically queried when a new track is streamed, or whenever the SIGUSR2 signal is received.

If the <metadata_progname/> element is present in the configuration, no attempts will be made to read metadata from files that are being streamed. If this behavior is not desired, it should be removed or commented out in the configuration file.

See the SCRIPTING section for details on how the metadata program must behave.

<metadata_format />
(Optional.) Set the format of the string that should be used for the '@M@' placeholder when setting metadata with an external program or script via <metadata_progname/>.

See the METADATA section for details on how metadata is handled by ezstream.

<stream_once />
Set to 1 (one) in order to stream the content of <filename/> only once, and to 0 (zero) for continuous streaming (the default).

<reconnect_tries />
Set how many attempts should be made to reconnect to the Icecast server in case the connection is interrupted. The default is to try indefinitely, which is equal to setting this configuration option to 0 (zero).

<svrinfoname />
(Optional.) Set the name of the broadcast. Informational only.

<svrinfourl />
(Optional.) Set the URL of the web site associated with the broad- cast. Informational only.

<svrinfogenre />
(Optional.) Set the genre of the broadcast. Informational only, used for YP.

<svrinfodescription />
(Optional.) Set the description of the broadcast. Informational only, used for YP.

<svrinfobitrate />
(Optional.) Set the bit rate of the broadcast. This setting is also purely informational and only used for YP. The value is set by the user and not ezstream, and should match the bit rate of the stream.

<svrinfoquality />
(Optional.) Set the quality setting of an Ogg Vorbis broadcast. Informational only and needs to be set by the user, used for YP.

<svrinfochannels />
(Optional.) Set the number of audio channels in the broadcast, e.g. 1 (one) for mono or 2 for stereo. Informational only and needs to be set by the user, used for YP.

<svrinfosamplerate />
(Optional.) Set the sample rate of the broadcast. Informational only and needs to be set by the user, used for YP.

<svrinfopublic />
(Optional.) Set to 1 (one) if the broadcast may be listed in a pub- lic YP directory. If set to 0 (zero), the Icecast server will not submit this stream to a YP directory, which is also the default if the <svrinfopublic/> element is absent.

<reencode />
(Optional.) Element that contains child elements, which specify if and how re-encoding should be done.

Re-encoding settings
Each of the re-encoding configuration elements have the <reencode/> ele- ment as their parent.

<enable />
Set to 1 (one) to enable re-encoding. If set to 0 (zero), no re- encoding will be done, which is also the default if the <enable/> element is absent.

<encdec />
Element that contains child elements, which specify how to decode and encode a certain media file format for streaming. Each format is described by a separate <encdec/> element.

Decoder/Encoder settings
Each of the decoder/encoder configuration elements have the <encdec/> element as their parent.

<format />
This element is used by ezstream to find the appropriate encoder for the output stream format specified in the <format/> element inside the global configuration. For consistency reasons, it is recom- mended that this element is always supplied, even for currently unsupported output formats, with content such as VORBIS, MP3, THEORA, FLAC, et cetera.

<match />
Set the filename extension used to identify a given media file for- mat. This allows ezstream to find the appropriate decoder for a given file. Should be set to .mp3 for MP3, .flac for FLAC, .ogg for Ogg Vorbis, and so on.

<decode />
Set the command to decode the specified media file format to raw data and send it to standard output. During runtime, the place- holder '@T@' is replaced with the name of the media file, as it is specified in the <filename/> element or contained in a playlist file. It should always be enclosed in quotes, to prevent problems with filenames that contain whitespaces.

Metadata placeholders can be used in the <decode/> element as well, for combined de-/encoder programs that produce data that can be streamed. See the METADATA section for details on how metadata is handled by ezstream.

For example, to decode Ogg Vorbis files using the oggdec utility:

<decode>oggdec -R -o - "@T@"</decode>

<encode />
Set the command to encode raw data, received from standard input, to the specified stream format.

Metadata placeholders can be used in the <encode/> element. For details about using metadata in ezstream, see below in the METADATA section.

For example, to encode an Ogg Vorbis stream using the quality set- ting 1.5 with the oggenc utility:

<encode>oggenc -r -q 1.5 -t "@M@" -</encode>

SCRIPTING

The ezstream utility provides hooks for externally controlled playlist and metadata management. This is done by running external programs or scripts that need to behave in ways explained here.

Common Rules
- The program must be an executable file.
- The program must write one line to standard output and exit.
- The program must not require arbitrary command line options to func- tion. A wrapper script must be used if there is no other way.

Playlist Programs
- The program must return only filenames, with one filename per execu- tion.
- The program should not return an empty line unless ezstream is sup- posed to know that the end of the playlist has been reached. This is significant when the <stream_once/> option is enabled.

Metadata Programs
- The program must not return anything (just a newline character is okay) if it is called by ezstream with a command line parameter that the program does not support.
- When called without command line parameters, the program should return a complete string that should be used for metadata.
- When called with the command line parameter "artist", the program should return only the artist information of the metadata. (Optional.)
- When called with the command line parameter "title", the program should return only the title information of the metadata. (Optional.)
- The supplied metadata must be encoded in UTF-8.

METADATA

The main tool for handling metadata with ezstream is placeholders in decoder and encoder commands that are replaced with real content during runtime. The tricky part is that one of the placeholders has to be han- dled differently, depending on where the metadata comes from. This sec- tion will explain each possible scenario.

Metadata Placeholders
@T@ Replaced with the media file name. Required in <decode/> and is available in <metadata_format/>.

@M@ Replaced with a metadata string. See below for a detailed explana- tion. Available in <decode/> and <encode/>.

@a@ Replaced with the artist information. Available in <decode/>, <encode/> and <metadata_format/>.

@t@ Replaced with the title information. Available in <decode/>, <encode/> and <metadata_format/>.

@s@ Replaced with the string returned by <metadata_progname/> when called without any command line parameters. Available only in <metadata_format/>.

The @M@ Placeholder
While all other placeholders are simply replaced with whatever data they are associated with, '@M@' is context-sensitive. The logic used by ezstream is the following:

```
If ('@M@ is present')
   If ('<metadata_progname/>' AND '<metadata_format/>')
      Replace with format string result.
   Else
      If (NOT '<metadata_progname/>' AND '@t@ is present')
         Replace with empty string.
      else
         Replace with generated metadata string.
      Endif
   Endif
```

Endif

The generated metadata string for '@M@' is of the format "Artist - Title", if both artist and title information is available. If one of the two is missing, the available one is displayed without a leading or trailing dash, e.g. just "Artist". If neither artist nor title are available, the name of the media file — without its file extension — is used.

Metadata Caveats
It is possible to generate strange results with odd combinations of placeholders, external metadata programs and updates during runtime via SIGUSR2. If things start to become just confusing, simplify.

Metadata updates during runtime are done with a relatively broken feature of libshout. Additional metadata information that is already present in the stream sent via ezstream is usually destroyed and replaced with the new data. It is not possible to properly discern between artist and title information, which means that anything set with the SIGUSR2 feature will continue to end up entirely in the "Title" field of a stream.

Of all possible Ogg-based streams, only Ogg Vorbis can have its metadata manipulated by Icecast. Any attempt of ezstream to update other Ogg metadata is actually a no-op.

While ezstream tries to do its best with relaying metadata accurately to Icecast, and subsequently the listeners, different codesets and locales can pose a problem. Especially when streaming MP3 files, it may help to explicitly set a codeset to work with via the LC_CTYPE environment vari- able, as ezstream assumes ID3v1 tags to be in the user's current locale. Note that, even though support for different locales is provided by ezstream, Icecast itself and the listening clients also have a say in the matter. The only way to ensure consistent results with metadata in non- Ogg streams is to use the characters available in the ISO-8859-1 codeset.

External encoders may put additional, and possibly artificial, restric- tions on valid characters in metadata.

FILES

/usr/share/examples/ezstream
Directory containing example configuration files for various uses of ezstream, as well as example playlist and metadata scripts.

AUTHORS

ezstream was written by:

Ed Zaleski ⟨oddsock@oddsock.org⟩
Moritz Grimm ⟨mdgrimm@gmx.net⟩

This manual was written by Moritz Grimm.

IceS

IceS v2.0.1

IceS .04 is a source client for broadcasting in Ogg Vorbis format to an icecast2 server

- Introduction

- Basic Setup

- Config File

- Available Input Modules

- Frequently Asked Questions

Introduction

What Is IceS ?

IceS is a source client for a streaming server. The purpose of this client is to provide an audio stream to a streaming server such that one or more listeners can access the stream. With this layout, this source client can be situated remotely from the icecast server.

The primary example of a streaming server used is Icecast 2, although others could be used if certain conditions are met.

What Platforms Are Supported ?

Currently the following Unix platforms are supported:

- Linux (Most flavors including Redhat and Debian)
- FreeBSD
- OpenBSD
- Solaris

Where Do I Go For Questions?

IceS is developed and maintained by the same people involved with Icecast 2. There are several ways to contact the development team

Best Ways

- Icecast mailing list http://www.xiph.org/archives
- Icecast Developers mailing list http://www.xiph.org/archives
- Icecast IRC chat room - irc.freenode.net : #icecast

Alternate Ways

- team@icecast.org

Basic Setup

What Does IceS Require?

IceS v2 is not a graphical application, it's purpose is to stream whatever it is given into a stream for feeding to the Icecast streaming server. It does however require the following:

- libogg available at http://www.vorbis.com
- libvorbis available at http://www.vorbis.com
- libxml2 available at xmlsoft
- libshout 2 available at The Icecast site

Please note that in many cases, pre-built packages are split into two, a run-time package, typically consisting of the actual run-time library and a development package, consisting of the support files needed to compile and link the application.

The following are optional

- ALSA. driver and libs available at The ALSA site

What Does IceS Do ?

IceS reads audio data from an input and sends the audio data to one or more files or icecast servers. Before it's actually sent out, some processing maybe performed, typically resampling and/or downmixing to produce streams suited to various bandwidth requirements.

The streams produced are Ogg Vorbis streams so while icecast 2 is capable of streaming these streams, other streaming servers may not be.

What Input Can IceS Handle?

Several inputs currently exist, but some maybe dependant on certain platforms or if certain drivers or libraries are available.

- OSS - Open Sound System. Typically used on linux based systems to get live input from a soundcard.
- ALSA - Advanced Linux Sound Architecture. Like OSS but with various improvements for linux based systems.
- stdinpcm - Uses standard input to receive PCM audio.
- playlist - Uses a playlist to read audio files for processing.
- sun - like OSS, but for Sun Solaris, also works for OpenBSD

How Do You Start IceS?

The configuration of IceS is done via an XML based config file. Which you supply as an argument to the program at invocation time. For example

ices /etc/ices.xml

Config File

The ices 2 configuration file is in XML format, which is described in detail below. There are some sample XML files provided in the distribution under the conf directory which show the main way ices is used.

live audio streaming, takes audio from the soundcard which can be captured from say the Mic, line-In, CD or a combination of these.

- ices-oss.xml
- ices-alsa.xml

Playlist audio streaming, takes pre-encoded Ogg Vorbis files and either sends them as-is or re-encodes them to different settings

- ices-playlist.xml

General Layout

```
<?xml version="1.0"?>
<ices>
    general settings
    stream section
</ices>
```

General Settings

These apply to IceS as a whole. The example below gives a useful example to work to

```
<background>0</background>
<logpath>/var/log/ices</logpath>
<logfile>ices.log</logfile>
<logsize>2048</logsize>
<loglevel>3</loglevel>
<consolelog>0</consolelog>
<pidfile>/var/log/ices/ices.pid</pidfile>
```

background

Set to 1 if you want IceS to put itself into the background.

logpath

A directory that can be written to by the user that IceS runs as. This can be anywhere you want but as log files are created, write access to the stated must be given.

logfile

The name of the logfile created. On log re-opening the existing logfile is renamed to <logfile>.1

logsize

When the log file reaches this size (in kilobytes) then the log file will be cycled (the default is 2Meg)

loglevel

A number that represents the amount of logging performed.

1 - Only error messages are logged
2 - The above and warning messages are logged
3 - The above and information messages are logged
4 - The above and debug messages are logged

consolelog

A value of 1 will cause the log messages to appear on the console instead of the log files. Setting this to 1 is generally discouraged as logs are cycled and writing to screen can cause stalls in the application, which is a problem for timing critical applications.

pidfile

State a filename with path to be created at start time. This file will then contain a single number which represents the process id of the running IceS. This process id can then be used to signal the application of certain events.

Stream Section

This describes how the input and outgoing streams are configured.

```
<stream>
    Metadata
    Input
    Instance
</stream>
```

Metadata

```
<metadata>
    <name>My Stream</name>
    <genre>Rock</genre>
    <description>A short description of your stream</description>
    <url>http://mysite.org</url>
</metadata>
```

This section describes what metadata information is passed in the headers at connection time to icecast. This applies to each instance defined within the stream section but maybe overridden by a per-instance <metadata> section.

Input

This section deals with getting the audio data into IceS. There are a few ways that can happen. Typically it's either from a playlist or via a soundcard.

The layout is consistent between the different input modules. Within the input section a module tag is needed to identify the module in question. The rest are made up of param tags specific to the module. There can be several param tags supplied to a module. Details of the module parameters are shown later.

Instance

Multiple instances can be defined to allow for multiple encodings, this is useful for encoding the same input to

multiple bitrates. Each instance defines a particular set actions that occur on the passed in audio. Any modifications to the input data is isolated to the instance.

```
<instance>
    hostname
    port
    password
    mount
    yp
    resample
    downmix
    savefile
    encode
</instance>
```

hostname

State the hostname of the icecast to contact, this can be a name or IP address and can be ipv4 or ipv6 on systems that support IPv6. The default is localhost.

port

State the port to connect to, this will be the port icecast is listening on, typically 8000 but can be any.

password

For providing a stream, a username/password has to be provided, and must match what icecast expects.

mount

Mountpoints are used to identify a particular stream on a icecast server, they must begin with / and for the sake of certain listening clients should end with the .ogg extension.

yp

By default streams will not be advertised on a YP server unless this is set, and only then if the icecast if configured to talk to YP servers.

Resample

```
<resample>
   <in-rate>44100</in-rate>
   <out-rate>22050</out-rate>
</resample>
```

When encoding or re-encoding, there is a point where you take PCM audio and encode to Ogg Vorbis. In some situations a particular encoded stream may require a lower samplerate to achieve a lower bitrate. The resample will modifiy the audio data before it enters the encoder, but does not affect other instances.

The most common values used are 48000, 44100, 22050 and 11025, and is really only used to resample to a lower samplerate, going to a higher rate serves no purpose within IceS.

Downmix

```
<downmix>1</downmix>
```

Some streams want to reduce the bitrate further, reducing the number of channels used to just 1. Converting stereo to mono is fairly common and when this is set to 1 the number of channels encoded is just 1. Like resample, this only affects the one instance it's enabled in.

Savefile

<savefile>/home/ices/dump/stream1.ogg</savefile>

Sometimes the stream transmitted wants to be saved to disk. This can be useful for live recordings.

encode

```
<encode>
    <quality>0</quality>
    <nominal-bitrate>65536</nominal-bitrate>
    <maximum-bitrate>131072</maximum-bitrate>
    <minimum-bitrate>-1</minimum-bitrate>
    <managed>0</managed>
    <samplerate>22050</samplerate>
    <channels>1</channels>
    <flush-samples>11000</flush-samples>
</encode>
```

quality

State a quality measure for the encoder. The range goes from -1 to 10 where -1 is the lowest bitrate selection (default 3), and decimals can also be stated, so for example 1.5 is valid. The actual bitrate used will depend on the tuning in the vorbis libs, the samplerate, channels and the audio to encode. A quality of 0 at 44100hz and 2 channels is typically around the 64kbps mark.

nominal-bitrate

State a bitrate that the encoder should try to keep to. This can be used as an alternative to selecting quality.

managed

State 1 to enable full bitrate management in the encoder. This is used with nominal-bitrate, maximum-bitrate and minimum-bitrate to produce a stream with more strict bitrate requirements. Enabling this currently leads to higher CPU usage.

maximum-bitrate

State bitrate in bits per second to limit max bandwidth used on a stream. Only applies if managed is enabled.

minimum-bitrate

State bitrate in bits per second to limit minimum bandwidth used on a stream. Only applies if managed is enabled, this option has very little use so should not really be needed.

samplerate

State the samplerate used for the encoding, this should be either the same as the input or the result of the resample. Getting the samplerate wrong will cause the audio to be represented wrong and therefore sound like it's running too fast or too slow.

channels

State the number of channels to use in the encoding. This will either be the number of channels from the input or 1 if downmix is enabled.

flush-samples

This is the trigger level at which Ogg pages are created for sending to the server. Depending on the bitrate and compression achieved a single ogg page can contain many seconds of audio which may not be wanted as that can trigger timeouts.

Setting this to the same value as the encode samplerate will mean that a page per second is sent, if a value that is half of the encoded samplerate is specified then 2 ogg pages per second are sent.

Available Input Modules

Several input modules are available, depending on the platform, drivers and libraries available. The general layout is defined as

```
<input>
   <module>module name</module>
   <param name="name1">value</param>
   <param name="name2">value</param>
</input>
```

For live input you may want to look into various resources on the web for information on sound input. You may find that ALSA for instance supports a particular soundcard better than the Open Sound System.

Open Sound

```
<module>oss</module>
<param name="rate">44100</param>
<param name="channels">2</param>
<param name="device">/dev/dsp</param>
<param name="metadata">1</param>
<param name="metadatafilename">/home/ices/metadata</param>
```

This module is for reading live input from the Open Sound System drivers, often found on linux systems but are available on others. This will read audio from the DSP device in a format specified in the parameters provided.

The following can be used to configure the module

rate

The value is in hertz, 44100 is the samplerate used on CD's, but some drivers may prefer 48000 (DAT) or you may want to use something lower.

channels

The number of channels to record. This is typically 2 for stereo or 1 for mono

device

The device to read the audio samples from, it's typically /dev/dsp but there maybe more than one card installed.

metadata

Check for metadata arriving, if any are present then the data is marked for an update. The metadata is in the form of tag=value, and while Ogg Vorbis can handle any supplied tags, most players will only do anything with artist and title.

metadatafilename

The name of the file to open and read the metadata tags from, with this parameter missing standard input is read. Using a file is often the better approach. When using the file access the procedure is usually to populate the file contents then send a SIGUSR1 to the IceS process.

The format of the file itself is a simple one comment per line format, below is a trivial example of the file, other

tags can be used but players tend to only look for artist and title for displaying. The data must be in UTF-8 (this is not checked by ices, however).

artist=Queen
title=We Will Rock You

ALSA

The Advanced Linux Sound Architecture (ALSA) is a completely different sound system on linux but provides OSS compatability so the OSS driver should work with it as well. To use ALSA natively a separate module is used

```
<module>alsa</module>
<param name="rate">44100</param>
<param name="channels">2</param>
<param name="device">hw:0,0</param>
<param name="periods">2</param>
<param name="buffer-time">500</param>
<param name="metadata">1</param>
<param name="metadatafilename">/home/ices/metadata</param>
```

The parameters to ALSA are mostly the same for OSS, as it performs the same task, ie captures audio from the DSP. device

This is the device name as used in ALSA. This can be a physical device as in the case of "hw:0,0" or a virtual device like one with dsnoop.

periods

This specifies how many interrupts will be generated (default: 2)

buffer-time

The size of the buffer measured in mS (default 500)

Sun

The Sun Solaris DSP input is similar to OSS. It allows for reading from a soundcard on a Sun Solaris UNIX. OpenBSD also has a sound driver that is similar to solaris and as such should be able to use this module.

```
<module>sun</module>
```

The parameters are the same as the OSS and ALSA modules.

StdinPCM

```
<module>stdinpcm</module>
<param name="rate">44100</param>
<param name="channels">2</param>
<param name="metadata">1</param>
<param name="metadatafilename">/home/ices/metadata</param>
```

This module should always be available, and as you can see the parameters are almost the same except for the

device. The PCM audio comes from the standard input so it can be generated from some external app feeding into a pipe.

As it's raw PCM being fed in, it's impossible to determine the samplerate and channels so make sure the stated parameters match the incoming PCM or the audio will be encoded wrongly.

Playlist

The playlist module is used to get audio from some pre-encoded Ogg Vorbis files. IceS currently checks to see if the same file gets played in succession and skips it, this means that having a playlist repeat with only one ogg file listed won't work. The method of file selection is determined by the playlist type. The current types are basic and script.

Basic

```
<param name="type">basic</param>
<param name="file">/path/to/playlist</param>
<param name="random">0</param>
<param name="once">0</param>
<param name="restart-after-reread">1</param>
```

file

State a path to a file which will contain a list of Ogg Vorbis filenames to play. One per line with lines beginning with '#' being treated as comments. If a line has a single '-' then standard input is read, which provides a way of getting some external Ogg Vorbis stream into ices.

random

When set to 1, the playlist will be randomised when the playlist is loaded. By default random is off

once

When set to 1, the playlist is gone through once and then ends, this will cause ices to exit. By default once is off.

restart-after-reread

If the playlist is re-read mid way through, which may occur if the playlist was updated then this will restart at the beginning of the playlist. By default it's off.

Script

```
<param name="type">script</param>
<param name="program">/path/to/program</param>
```

Program

State a path to a program which when run will write to it's standard output a path to an Ogg Vorbis file. The program can be anything from an executable to a shell script as long as it starts, writes the filename to it's standard output and then exits.

Frequently Asked Questions

This for those questions people have asked as it wasn't covered in the documentation

Can ices play mp3 files?

No, there hasn't been much interest in handling MP3 with ices 2. The older version ices 0.x maybe of interest in such cases. If you really want to encode the Vorbis stream from non-vorbis files then you can play them with an external application, eg xmms, and use ices 2 to capture from the soundcard, but be aware that any conversion from one lossy format to another is bad so make sure the original material is high quality.

How do I encode from Line-In?

IceS will read from the DSP on the soundcard, but where that gets the audio from depends on the mixer settings. Use a utility like aumix/alsamixer to see the settings and change the capture or recording device. Usually the default is the Mic

When I start ices 2 it seems to get stuck not doing anything?

If you are using live input, check to see if something else is holding the recording device (typically /dev/dsp) open. A good candidate is esd. What happens is that the driver only allows one application to have the device open at any one time, a second attempt will just block.

Some OSS drivers allow multiple opens but on ALSA you can configure a virtual device in asound.conf, type dsnoop/dmix, which allows access for multiple apps.

Ices reports a message about failing to set samplerate on live input?

Some hardware/drivers are limited in the settings they support. Sometimes they only support one samplerate like 48khz. You have to experiment if the documentation for the device is not specific.

Can I do crossfading with the playlist?

Ices does not do much in the way manipulating the audio itself, resampling and downmixing are available as that has a direct effect on encoding an outgoing stream. Ices can still be used in conjunction with other applications such as xmms by making ices read from the say the dsp (eg oss, alsa etc), that way anything that is played by that other application is encoded and sent to icecast.

My player connects but I don't hear anything?

If you are getting data through to the player and the settings show like the samplerate and chnnels then it is probably the mixer settings which are set incorrectly and ices can only read silence. A common example, ALSA may have many levels in the mixer and by default they are all muted.

My player seems unable to play the stream?

If the stream looks to be getting to the player then it will be how the player is handling it. The usual causes of this are

- Missing ".ogg" extension. Both ices and icecast do not care about the extension however some apps use the extension to determine which plugin to use.

- Missing Ogg Vorbis plugin. The winamp lite versions were known for this.

- Are you running Winamp 3. This is a discontinued product and had problems with the vorbis plugin, either use the later v2.9 series or v5.

The sound quality is poor

Ogg Vorbis is a lossy compression technology, so quality of the sound is reduced, however with live input the source of audio can be poor depending on the soundcard and the system it's in. As an initial test just record a wav file from the DSP (using eg rec, arecord etc) and listen to the quality of the audio recorded. If the source of audio is poor then encoding it to Ogg Vorbis is not going to improve it.

The reasons for poor audio from the DSP can be difficult to resolve, search for information on audio quality. It could be driver related or maybe some interference from some other device.

Here are some links where further information can be found:
http://www.djcj.org/LAU/guide/index.php
http://www.linuxdj.com/audio/lad/index.php3

IceS v0.4

IceS v.04 is a source client for broadcasting in MP3 format to an icecast2 server

- Introduction

- What's It For?

- What Can It Do?

- Configuring

- Licensing

- Developers Resources

Introduction

For a very long time, the only good streaming tool for command line systems was shout. Shout had a lot of issues, it was a quick and dirty hack, it was buggy, it was sending data too fast or too slow, and it was just something no one wanted to fix. So we rewrote it from scratch, and the next generation streamer, 'ices', is here. 'ices' is short for 'icesource', a source for the icecast server. 'ices' should be pronounced 'isis' like the egyptian godess of fertility. For more information about icecast, I suggest you check out the icecast webpage at http://www.icecast.org/.

What's It For

Ices, armed with a list of mp3 files, sends a continuous stream of mp3 data to an icecast server. The server is then responsible for accepting client connections and feeding the mp3 stream to them. But the stream originates in the streamer, 'ices'. The terms 'encoder', 'streamer' and 'source' are used equivalently throughout this document, and throughout all the documentation in the streaming system.

What Can It Do

Cue File

The cue file holds information on the file that ices is currently feeding to the server. This is neat for you people out there who like running scripts. I myself, use the cue file in a tcl script, running from a eggdrop bot, on irc. That way I can ask the bot what song is currently playing, how long it is, how much of it has been played, and get information about the next songs on the playlist.

The file currently has the following lines, (in this order).
- filename
- size (in bytes)
- bitrate (in kbits/s)
- minutes:seconds (total song length)
- percent played (i.e 25, no %-sign)
- playlist line index (i.e 3, if we're playing the 4:th line in the internal playlist. Logical, huh?) Also, for you scripting people, when ices starts, it writes its process id to the file ices.pid.
- ID3 Artist
- ID3 Title

Signal Handling

- Sending SIGINT to ices will make it exit.
- Sending SIGHUP to ices will make it close and reopen the logfile and playlist, and reload and restart the playlist script if you are using one.
- Sending SIGUSR1 to ices will make it skip to the next track.

Re-encoding

If compiled with support for reencoding using liblame, and you supply the -R command line option or set the Stream/Reencode to 1 in the XML config file, then ices will start reencoding your files on the fly to the bitrate you specified with the -b option or the Stream/Bitrate tag.

If you are re-encoding and ices was compiled with vorbis support, you may also re-encode Ogg Vorbis files as MP3 on the fly. This gives you the opportunity to convert your source files to Ogg Vorbis at your convenience while still supporting as many listeners as possible. Likewise, ices can transcode FLAC and MP4 (AAC) files if you've compiled it with FLAC and FAAD libraries, respectively.

The sample rate, number of channels, etc, will be chosen on the fly by lame itself, unless you specify something using the -H and -N options. I think you should be fine with whatever lame chooses though.

Also, please make sure that your files are ok before you start reencoding them with ices. This is because the mpglib part of lame (what does the decoding) is rather unstable and will call exit(0) when errors are encountered.

This will make ices exit, which is kinda bad :)

Crossfading

If you've compiled with support for reencoding, you can crossfade between tracks (blend the end of one into the start of the next). This is controlled by the -C command line option or the Playlist/Crossfade parameter in the configuration file. Both of these take an integer argument, which is the number of seconds to crossfade. Songs less than twice the length of the crossfade requested will not be faded. This is handy for eg station IDs.

Multiple Streams

You can feed the same playlist simultaneously to different mountpoints, by specifying multiple Stream sections in the config file or passing multiple -moptions on the command line. This is especially useful in conjunction with reencoding because you can stream the same music at a high bitrate for broadband listeners and simultaneously at a low bitrate for POTS listeners.

Playlist Handling

About 96% of all emails I got about shout was people asking me to add small changes to shout playlist handling to suit their specific needs. This is course is not how I want to spend my life :) Shout had a feature to call an external program with a system() call, before each song, and that could possibly modify the playlist. This was rather ugly, but did the trick. In ices, we take this a step further and include scripting support inside the program. You can write your own playlist handler in perl or python, whatever you prefer.

Your script module has to define at least a function named ices_get_next, which should return a path to a file or FIFO containing MP3 data.

In addition you may define the functions ices_init and ices_shutdown which will be called by ices once before asking for the first song and before shutting down, respectively.

You may also define ices_get_lineno, which specifies the line number of the current track in the cue file. If you don't use the cue file it is safe to omit this function.

Finally you can define ices_get_metadata to return a string you want to use for title streaming. Ices will call this function once per track after calling ices_get_next. If this function is not defined or returns null, ices will use whatever it can get out of the file itself, either tags or the file name.

I suggest you take a look in the distributed module files and just expand on that.

Configuring

Ices can do everything shout could do, and more. It can be configured through hard coded defaults, a configfile, and command line options. The configfile is in XML, but don't get scared and run off. Just edit the distributed configfile and change the values you need. The command line options should be familiar to old shout users, although some options have been renamed.

Command Line Options

Options:

-B (Background (daemon mode))
-b <stream bitrate>
-C <crossfade seconds>
-c <configfile>
-D <base directory>
-d <stream description>
-f <dumpfile on server>
-F <playlist>
-g <stream genre>
-h <host>
-i (use icy headers)
-M <interpreter module>
-m <mountpoint>
-n <stream name>
-p <port>
-P <password>
-r (randomize playlist)
-s (private stream)
-S <perl|python|builtin>
-u <stream url>
-N <Re-encoded number of channels>
-H <Re-encoded sample rate>

Note that each time you specify a mount point with -m you are creating a new stream, and subsequent stream options will apply only to it.

Configuration File (ices.conf)

Here's a sample configuration file. It's the same as the ices.conf.dist that is included in the ices distribution. You can specify multiple stream sections with different mountpoints, names, and reencoding options.

```
<?xml version="1.0"?>
<ices:Configuration xmlns:ices="http://www.icecast.org/projects/ices">
<Playlist>
      <File>apan.txt</File>
         <Randomize>1</Randomize>
         <Type>builtin</Type>
         <Module>ices</Module>
        <Crossfade>0;<Crossfade>
</Playlist>
<Server>
        <Hostname>localhost</Hostname>
        <Port>8000</Port>
        <Password>letmein</Password>
```

```
        <Protocol>xaudiocast</Protocol>
</Server>
<Execution>
        <Background>0</Background>
        <Verbose>1</Verbose>
        <Base_Directory>/tmp</Base_Directory>
</Execution>
<Stream>
        <Name>Cool ices default name from XML</Name>
        <Genre>Cool ices genre from XML</Genre>
        <Description>Cool ices description from XML</Description>
        <URL>Cool ices URL from XML</URL>
        <Bitrate>128</Bitrate>
        <Public>1</Public>
       <Reencode>0</Reencode>
       <Samplerate>-1</Samplerate>
       <Channels>-1</Channels>
</Stream>
</ices:Configuration>
```

Configurations options

This describes all the different options in ices.

- Server Hostname
Command line option: -h <host>
Config file tag: Server/Hostname

This is the name, or ip, of the host ices should connect to. It has to run a streaming server, capable of the xaudiocast or icy protocol.

This value defaults to localhost.

- Server Port
Command line option: -p <port>
Config file tag: Server/Port
This is the port the server is listening on, by default 8000.

- Server Password
Command line option: -P <password>
Config file tag: Server/Password

The encoder password for the server. If this is not correct, then ices cannot log in on the server, and ices will exit.

- Server Protocol
Command line option: -i for icy-headers
Config file tag: Server/Protocol

Either xaudiocast or icy. Use xaudiocast if you can, and icy if you must.

- Execution Background
Command line option: -B
Config file tag: Execution/Background
This will launch ices in the background, as a daemon.

- Execution Verbose

Command line option: -v
Config file tag: Execution/Verbose

Normally ices outputs what stream is playing and a small amount of extra information. With verbose turned on, you get a whole lot of debugging information and lots of track info.

- Execution Base Directory
Command line option: -D <directory>
Config file tag: Execution/Base_directory

Ices uses this directory for cue files, log files and temporary playlist files. You need write permissions in this directory. The default is /tmp

- Stream Mountpoint
Command line option: -m <mountpoint>
Config file tag: Stream/Mountpoint

This is the mountpoint of the stream on the icecast server, if using the x-audiocast protocol.

- Stream Name
Command line option: -n <stream name>
Config file tag: Stream/Name

This is the name of the stream, not to be confused with the name of the song playing.

- Stream Genre
Command line option: -g <stream genre>
Config file tag: Stream/Genre

This is the genre of your stream, e.g Jazz or Static.

- Stream Description
Command line option: -d <stream description>
Config file tag: Stream/Description

This option is a description of your stream.

- Stream URL
Command line option: -u <URL>
Config file tag: Stream/URL

This should be a URL describing your stream.

- Stream Bitrate
Command line option: -b <bitrate>
Config file tag: Stream/Bitrate

If you turn on reencoding then this will be the bitrate of the stream, otherwize the actual bitrate of the stream is the bitrate your files are encoded at, and this value is for displaying purposes only.

Read the last 3 lines again, please.

- Stream Public
Command line option: -s (makes stream private)
Config file tag: Stream/Public

This regulates whether the icecast server will display your stream on a directory server. Default is 1 (yes).

- Stream Re-encode
Command line option: -R (turns re-encoding on)
Config file tag: Stream/Reencode

When you turn this option on, ices (if compiled with libmp3lame support) will re-encode your mp3 files on the fly to whatever bitrate you specify with the Stream Bitrate option, unless the file bitrate is the same as the stream bitrate.

PLEASE NOTE: that if your files are corrupt, this might crash ices because the library used to decode (mpglib) is not very stable. I suggest you check your files with mp3check or some other mp3 verification program before you add them to your playlist. Just popping them into your favourite player and checking the sound is definitely not enough.

For legal reasons, this option is not available on the binary distributions.

- Stream Samplerate
Command line option: -H <samplerate>
Config file tag: Stream/Samplerate

Use this to force reencoding to output mp3 data with this samplerate.

- Stream Channels
Command line option: -N <number of channel>
Config file tag: Stream/Channels

Use this to force reencoding to output mp3 data with this many channels.

- Playlist File
Command line option: -F <file>
Config file tag: Playlist/File

This is the file where ices originally looks for files to play.
When using playlist modules in perl or python, this argument is ignored.

- Playlist Randomize
Command line option: -r (randomizes file)
Config file tag: Playlist/Randomize

This option is passed to the playlist handler, and tells it to randomize the playlist.

- Playlist Type
Command line option: -S <perl|python|builtin>
Config file tag: Playlist/Type

By default, ices using a builtin playlist handler. It handles randomization and not much more. Most people want sophisticated playlist handlers that interface with databases and keep track of god knows what. ices handles embedded python and embedded perl, so now you can write your own modules, without modifying ices, that do just about anything. Use this option to change the playlist handler type from builtin (default), to python or perl.

- Playlist Module
Command line option: -M <module>
Config file tag: Playlist/Module

Use this option to execute a different python or perl module than the default.

Default for python is ices.py and default for perl is ices.pm, although do NOT specify the file extension for the module. Use 'whatever' instead of 'whatever.pm' or 'whatever.py'

- Crossfade
Command line option: -C <seconds>
Config file tag: Playlist/Crossfade

If this option is specified and reencoding is enabled, ices will crossfade seconds between tracks.

Licensing

Ices is licensed under the Gnu General Public License, and for more info about that I suggest you read the file named COPYING.

Developers resources

If you want to write your own streaming software, or perhaps a nice streaming mp3 client, go to http://developer.icecast.org.

This document was mostly written by Alexander Haväng [eel@icecast.org].

Sourcing Multimedia Content With The Video Lan Client (VLC)

- Introduction

- Installing VLC With libshout Support

- Command Line Syntax For Streaming Content To An Icecast Server

- VLC Icecast Output (Command Line Options)

Introduction

The Video Lan Client (VLC) can be used to source multimedia to an Icecast server. The multimedia content that is being sourced must be compatible with the Ogg, MP3 format.

You must enable VLC to support the Icecast Output method to allow VLC to source multimedia to an Icecast server.

This is done with the addition and enabling of the libshout library. Pre-compiled packages do not automatically enable the libshout library.

Installing VLC With libshout Support

(For Debian Linux)

Install libshout.
#sudo apt-get install libshout-dev

Install VLC.
#sudo apt-get build-dep vlc

Configure VLC to enable libshout support.
#./configure --prefix=/opt/vlc-0.8.6x/ --exec-prefix=/opt/vlc-0.8.6x/ --enable-theora --enable-shout --enable-v4l –enable-dvb

Compile.
#make
#make install

Check VLC to ensure that libshout support has been enabled.
#cd /opt/vlc-0.8.6x/bin
#./vlc -l | grep shout

If libshout support has been properly enabled you will get this print out.
#VLC media player 0.8.6a Janus
#access_output_shout IceCAST output
#playlist New winamp 5.2 shoutcast import
#shout Shoutcast radio lijsten
#shout Shoutcast TV listings

http://forum.videolan.org/viewtopic.php?f=4&t=14476&start=30

Command Line Syntax For Streaming Content To An Icecast Server

stream single multimedia file
:duplicate{dst=std{access=shout,mux=ogg,dst=source:hackme@192.168.1.6:8000/audio.ogg}}

stream multimedia from input
:std{access=shout{name="mystream"},mux=ogg,dst=source:hackme@192.168.1.6:8000/audio.ogg}"

source = server_ password @ server_ip_address : port number / stream_name . ogg

http://forum.videolan.org/viewtopic.php?f=4&t=14476&start=30

Note:

You must also include information additional information in the command line syntax, such as the source of the files or input device and if you intend to transcode the multimedia content into another codec.

http://www.videolan.org/doc/streaming-howto/en/ch03.html

VLC Icecast Output (Command Line Options)

--sout-shout-name=<string> (Stream name)
Name to give to this stream/channel on the shoutcast/icecast server.

--sout-shout-description=<string> (Stream description)
Description of the stream content or information about your channel.

--sout-shout-mp3, --no-sout-shout-mp3 (Stream MP3 (default disabled))
You normally have to feed the shoutcast module with Ogg streams. It is also possible to stream MP3 instead, so you can forward MP3 streams to the shoutcast/icecast server. (default disabled)

--sout-shout-genre=<string> (Genre description)
Genre of the content.

--sout-shout-url=<string> (URL description)
URL with information about the stream or your channel.

--sout-shout-bitrate=<string> (Bitrate)
Bitrate information of the transcoded stream.

--sout-shout-samplerate=<string> (Samplerate)
Samplerate information of the transcoded stream.

--sout-shout-channels=<string> (Number of channels)
Number of channels information of the transcoded stream.

--sout-shout-quality=<string> (Ogg Vorbis Quality)
Ogg Vorbis Quality information of the transcoded stream.

--sout-shout-public, --no-sout-shout-public (Stream public (default disabled))
Make the server publicly available on the 'Yellow Pages' (directory listing of streams) on the icecast/shoutcast website. Requires the bitrate information specified for shoutcast. Requires Ogg streaming for icecast. (default disabled)

http://wiki.videolan.org/VLC_command-line_help