The SHOUTcast Streaming Standard [Technical]

Jay

The SHOUTcast Streaming Standard [Technical]

Wed Feb 20, 2002 2:00 pm

Intro
Not much is available about how our streaming technology works, more specifically MP3 streaming. When you get right down to it, it is actually very simple to understand. You will especially understand it if you know anything about HTTP's Protocol, which is simply a communication method for a client to a server. This article is mainly geared for those hard core into this stuff or those who want to develop applications similar to those utilized at this site.

Tools used
The tools used for streaming are quite simple:

- Source (usually a dsp in a player)
- Server (dispatches the source's mp3 stream to the client)
- Client (is used to listen to the audio coming down from the server

Source to Server
In order for a server to allow connections from a client, it needs a source. When a source connection is made then the server will pass the data along to the clients when they connect.

The dialog goes something like this (I will use SHOUTcast as the example)

1. The source makes a connection to the service port (shoutcast's is the port +1)
2. The source then sends the password like so password\r\n
3. If the password is correct, the server will reply with OK2\r\nicy-caps:11\r\n\r\n, this basically informs the source that the server has authorized the dsp to be the source and it is ready for data. If the password is incorrect, the server sends invalid password\r\n.
4. If the source receives the OK2, it then begins sending information about the stream to the server. Usually in this form:

CODE: SELECT ALL
```
icy-name:Unnamed Server\r\n
icy-genre:Unknown Genre\r\n
icy-pub:1\r\n
icy-br:56\r\n
icy-url:http://www.shoutcast.com\r\n
icy-irc:%23shoutcast\r\n
icy-icq:0\r\n
```

```
icy-aim:N%2FA\r\n
\r\n
```

Then The source will begin sending the mp3 encoded stream
* icy-name is the name of the stations
* icy-genre is the genre that the station resides in
* icy-pub is basically a switch to either allow the server to publish itself in the directory or not (1 meaning yes and 0 meaning no)
* icy-br is the bitrate of the stream
* icy-url is the homepage for the stream
* icy-irc is yp shoutcast specific (used for contact information)
* icy-icq is yp shoutcast specific (used for contact information)
* icy-aim is yp shoutcast specific (used for contact information)

You can also pass this optional data:

CODE: **SELECT ALL**

```
content-type: mime/type\r\n
icy-reset: 1\r\n
icy-prebuffer: ??\r\n
```

* content-type is the data type to expect from this stream. (HTTP spec header)
* icy-reset tells the server whether it should clear out the buffer. (necessary for NSV/NSA streams.
* icy-prebuffer, we aren't quite certain what this is for, how to use it or even whether it works, but it exists.

The optional params are not necessarily passed to the client, content-type is of course but as for the others it is not clear.

This is just a simple walk through of how the source communicates with the server. No other information is passed on this port as far as I am aware.

Title streaming from source to server
This is a simple one, the server receives the title of the song and the URL of the page simply by having the source make the URL call

CODE: **SELECT ALL**

```
http://www.host.com:portnumber/admin.cgi?pass=Server%20Password&mode=updinfo
```

When this gets called by the source or a browser even, the title of the song changes in the clients which support shoutcast style title streaming. This communication always happens on the public port (by default 8000) never on the service port as it is used for strictly sending the stream to the server.

You also must make sure that when you make your HTTP calls that it comes from a browser or program that specifies the User-Agent: header as Mozilla.

Client to Server
The Client to Server communication is handle in a similar fashion to the way that a browser communicates with a webpage server. This is known as the HTTP protocol. However SHOUTcast and icecast do not handle in exactly the same manner, the headers are different. I have yet to pin point exactly what is so different. I think it may have something to do with the notification error, as it is HTTP/1.0 200 OK on all web servers using HTTP, this may confuse some clients and causes the headers to not exist.
1. The client connects to the server and sends information about itself, if it can handle title streaming it sends and extra field like so:

CODE: SELECT ALL

```
icy-metadata:1\r\n
```

In addition to the normal headers sent. This tag signifies that the client has the ability to stream the title streaming tags from the stream, therefore the server will send the extra title information, if this were not possible, some clients would hiccup when the title information is sent
2. The server then responds with

CODE: SELECT ALL

```
icy-genre:Unknown Genre\r\n (what genre the server falls under)

icy-url:http://www.shoutcast.com\r\n (homepage for the server)

Content-Type:audio/mpeg\r\n (Content type of the stream to follow)

icy-pub:1\r\n (whether the server is public or not)

icy-br:56\r\n (bitrate of the server)

icy-metaint:8192\r\n (if icy-metadata:1 was signified this was shown I will

\r\n (end of header)
```

3. At this point the server begins sending the audio data.

SHOUTcast Meta Title Streaming
Earlier we discussed how the server gets the title of the song from the source, but we didn't quite get into how the client gets the title of the song.

When the client signifies that it is title streaming compatible, the shoutcast server adds an extra header tag set like so

CODE: SELECT ALL

```
icy-metaint:8192\r\n
```

this tells the client exactly how many bytes of data to read out of the stream before it can expect the beginning of the Meta-Data (which is where the title is stored) It also always starts counting at the beginning of the stream (not the header)
After this the client then reads 1 byte, this byte tells the client how large the Meta-Data Tag is divided by 16, so if the byte was 4 then the client would know that the meta-data tag was 64 bytes long. But, you ask, not all titles are going to equal 64 bytes or 48 bytes etc...? Well the simple answer is that SHOUTcast places blanks or "\0" in the unused space until it equals the length, after that is read, then it is back to the mpeg data to start the process all over again.

Pretty simple huh?

In Closing
I am sure that this technology will change, and I will try my best to keep this article up to the specs as I know them. If you find anything incorrect in this article, or any oversights, then please email me

Feel free to leave a comment if you feel that I have missed something. Do not reply with questions. Questions should go in the Audio Streaming forum. Questions will be split from this thread and moved to appropriate forums.

Last edited by Jay on Mon Jan 16, 2006 9:41 am, edited 6 times in total.

testicalls

Passing the mp3 binary.

Wed Oct 27, 2004 11:04 am

Lets say that I don't want to bother with a DSP. What if I knew what mp3 i wanted to pass to the server but without the need of opening winamp, loading the dsp, and running the file.

I assume the following process:
Make a script/program, that loads into memory the mp3 that needs to be passed. Lame, or mp3pro, encode then pass the necessary informationt to the server in order to initiate a transfer. When the server returns icy 200 (ok) then what happens? How would I pass the mp3 to the server, I assume a socket connection but is there a specific method?

Thanks for the little lecture, hope I understood.
-Testi

ThuGie

YP Protocol

Fri Dec 31, 2004 5:25 am

I downloaded today the new shoutcast server and i was thinking i'm gone scan what is sending to the yp so i did and this is what i got
not sure of anything changed.

First what you need to send
GET /addsrv?v=1&br=24&p=8000&m=3&t=Station Name&g=Genre&url=http://shoutcast.com&content=audio/mpeg&irc=#shoutcast&icq=0&aim =N/A HTTP/1.0
Host: yp.shoutcast.com
Accept: */*

this is what you send 😃
ok v=unknown to me sorry
br=bitrate
p=port
m=max users i gues
t=station name
g=genre
url=website url
content=type of stream
irc=irc channel
aim= gues 😉

it always sends this or it got updated 😉:
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2004 11:07:27 GMT
Server: Apache/1.3.33 (Unix) PHP/4.3.10
X-Powered-By: PHP/4.3.10
Connection: close
Content-Type: shoutcast/crapola

oke when you send it but did not enter a station name you get this error:

icy-response: nak

icy-error: please give your station a name (in the dsp plugin YP tab)

oke when you did not send the genre:
icy-response: nak
icy-error: please identify this station's genre (in the dsp plugin YP tab)

icy-response: nak = failed adding i gues

when sending and succes.
icy-response: ack
icy-id: 75568272
icy-tchfrq: 3

icy-response: ack = succes i gues
icy-id: ? = the id on the yp also used to remove yourself
icy-tchfrq: 3 = unknown sorry

oke now to remove yourself:
GET /remsrv?id=?&p=8000 HTTP/1.0
Host: yp.shoutcast.com
Accept: */*

id=the id you got when succes adding.
p=port of stream.

when succes removing you get:
icy-response: ack

this is all i got :p i hope you can use it i dont think so but still :p.
i was bored 😃 theres more i gues but this only took me 30sec to scan so 😃.

Jay

Mon Jan 31, 2005 11:45 pm

There is definately alot more to it and there are certain restrictions as well. I will not publish our knowledge of their yp due to security concerns (hacking the directory is a big problem at shoutcast.com due to the heavy traffic shoutcast.com recieves).

- Jay

Andrew

Mon Feb 07, 2005 2:19 pm

icy-tchfrq: 3

means that shoutcast will send updates to the yp.shoutcast.com every 3 minutes.

As the yp sends this back they can lengthen or shorten the update time per server if they so wish.

icy-id: xxxxx

is a unique Id within the shoutcast directory for the station entry.

icy-backup:

can also be sent back ... for some reason ... 😉

Andrew

djspinnercee

Excellent Info!

Wed Jun 29, 2005 12:18 pm

I have been playing with this too.... a project I\'ve been playing with is a alternate directory app that behaves like the SHOUTcast directory....

There is some info that I can add that may be useful:

I think that the icy-backup: will specify an DNAS id or IP that is used to help create clusters -- so when this is sent to a DNAS it will \"know\" that backups exist to it can redirect when full.

So far, I have found three commands that the DNAS sends to the YP:

In forming its request, the important HTTP header for all is:
Content-Type: shoutcast/crapola(0d)(0a)(0d)(0a)

The DNAS always initiates (establishes) the connections to the YP.

Then the icy-* things are in the content part of the request....

[yp-add]
GET /addsrv?
v=1 (don\'t know this, maybe 1=public, 0=private)
br=bitrate
p=port
m=DNAS MaxListeners
t=stream title
g=genre
url=URL
irc=
icq=
aim=

for this OP, the \"directory checker\" is called while the connection is still active -- it will try to extablish both a \"listen\" connection and a station page connection to determine accessability, validity, and I think to also get the current track info.

if the directory checker returns an \"all good,\" it will finish this request with:
icy-response: ack(0a)
icy-id: nnnnnnn(0a)
icy-tchfrq: 3(0a)
....and close the connection.

[yp-rem] is as above....
GET /remsrv?
id=nnnnnnnn
p=port

...the YP response body will only contain:
icy-response: ack(0a)

[yp-tch] Track title update...
GET /tchsrv?
id=nnnnnnnn
p=port
li=current listeners
alt=average listen time
ct=now playing track title

The YP responds:
icy-tchfrq: 5(0a)
icy-response: ack(0a)

Errors are returned by the YP as:
icy-response: nak(0a)
icy-error: ...the appropriate message ...(0a)

I assume that a NAK error will tell the DNAS to retry the failed OP at some other time.

****

I have a simple YP sim running to test this.... because the DNAS does not allow config of the target YP (IP) and only the port (YPort=), I create a HOSTS file entry for yp.shoutcast.com

nnn.nnn.nnn.nnn yp.shoutcast.com

nnn.nnn.nnn.nnn is the raw IP found from a ping or tracert to www411.servehttp.com (the IP is dynamic, so it can change)

in the DNAS config, the YPort=11427

Feel free to try it on your own --- so far the interaction is all that is going on, so it always returns ack -- I\'m working on the database, and the directorry checker with intentions of being able to build a directory....

I\'m looking for feedback, or additional info that may be helpful....

Jay

Wed Jun 29, 2005 4:12 pm

If you are interested in building a directory you might want to watch
http://sourceforge.net/projects/yp-php/

I will be publishing my version of an Icecast2 directory there which will be able to list both Steamcast and Icecast2 streams.

- Jay

webranger

Sat Jul 02, 2005 9:48 pm

AOL may have lost it's soul, but I'm glad you didn't 😀

Image
[size=0]New! City of Roses Radio - Rock, Blues, Jazz More[/size]

donotdespisethesnake

Mon Jul 11, 2005 1:46 pm

Hi there

I am trying to understand exactly what the client sends (and get back). If the client gets a playlist like

CODE: SELECT ALL

```
[playlist]
numberofentries=5
File1=http://64.236.34.97:80/stream/1005
Title1=(#1 - 521/15388) Smoothjazz.Com - The worlds best Smooth Jazz - Live
Length1=-1
```

What does the HTTP request look like? If I telnet to 64.236.34.97 on port 80 and send

CODE: SELECT ALL

```
GET /stream/1005 HTTP/1.1
Accept: audio/mpeg
```

te connection is dropped. I didn\\\\\\\'t really expect it to work but it is the quickest way to try. Can anyone suggest what the request should look like (and perhaps a better way to test it). I am probably missing something obvious (like a brain, 😜)

Cheers

Jay

Mon Jul 11, 2005 2:01 pm

Ultravox servers require that a User-Agent be passed. You should make it a habit to utilize this header in all of your HTTP transactions anyhow.

- Jay

donotdespisethesnake

Mon Jul 11, 2005 4:28 pm

Cool! That works much better. Thanks.

All I need to do now is write the code!

skoehler

Fri May 16, 2008 10:55 pm

Who invented this protocol? Because it's so simple to messup something so pretty like HTTP.

So what do you think will a HTTP-client or a HTTP-proxy do with an reply like "ICY 200 OK" ?
Right - he will say "oh, i don't know what ICY is, but it's surely not HTTP".
And then, in the good old days, there was HTTP 0.9. So he may actually interpretate things in HTTP 0.9 fashion. That is, that the "ICY 200 OK" and all the headers following it are interpretated as the data of the response. That's right, HTTP 0.9 didn't know any response-headers.

As an example take "http://gffstream.ic.llnwd.net/stream/gf ... einslive_a".

There's a limecast 2.0 server working - replying "ICY 200 OK" to an HTTP-request. (HTTP-request deserve nothing else than HTTP-responses). What the hell did the investors think when designing this!?

So mp3-players like WinAMP or Audacious may work, when directly connected.
Now imagine, the connection is handled by a proxy - maybe some famous software like squid.
What does squid do? Well, it's an HTTP proxy, and it's handling HTTP like it should be handled. That is: treating some "ICY 200 OK" like HTTP 0.9. mp3playback is horribly broken.

ICY should die. It can be replaced by a simpler properly HTTP-based mechanisms.

Jay

Sat May 17, 2008 3:13 pm

You can think the Nullsoft folks for this one. When it was written back in 1998 it wasn't really intended to be under it's own standard. It is simply a hack up of HTTP so they could support both in the player. Ultravox is supposed to fix all this but it hasn't really caught on nor have they really done anything with the protocol in their current server offering.

- Jay

polemon

Sun Aug 03, 2008 11:08 pm

hmm, this article has been lying around for a few years, but I have a related question, so I'll post this here:

in [yp-tch] (track title update)
/tchsrv takes the following additional parameters:

cm=(long unsigned int)
ht=(long unsigned int)

What are those parameters?

Jay

Mon Aug 04, 2008 2:26 pm

Those are stats that are sent back to the yp for listener connect times and stream hits.

- Jay

Post Reply

30 posts 1 2

Return to “Articles”

Jump to

Board index

Delete cookies All times are UTC-05:00