Audacious - Feature #109

Scrobbling support for http streams

May 08, 2012 20:25 - Michael Jacobs

Status:	Closed	**Start date:**	May 08, 2012
Priority:	Minor	**Due date:**	
Assignee:		**% Done:**	100%
Category:	plugins/scrobbler	**Estimated time:**	0.00 hour
Target version:	3.3		
Affects version:			

Description

I did a small hack to also provide scrobbling for http streams. Check the log of my conversation with John Lindgren for further details.

Hi John,

I recently started to listen to an online stream using audacious, which has been my favorite music player for years now.

I am not even sure, if you are the right person to ask concerning the scrobbler plugin, but I just saw on git hub, that you were the last one who worked on the corresponding files.

A thing that annoyed me a bit, was that none of the tracks I listened to in the stream was scrobbeled to last.fm, although all the tracks where properly tagged. After taking a look into the scroobler plugin's source, I discovered, that this is no accident and was just intended by design.

So my question is: Is it planned to implement scrobbling also for internet streams? Or was it lewft out because many streams do not provide meaningful tags or additionally add their own stuff to them?

Last weekend, I spent a few hours on the plugin and added a straight forward hack that enabled the scrobbling for streams and also detected when the tag changed for a new title. I am not the best C programmer (probably I violated many coding conventions or added some vulnerabilities) and I only tested the build and running on my machine (Arch Linux). But maybe my piece of code could be usefull to safe some work if somebody at the plugin team decides to implement scrobbling for streams one day. I would like to provide the code to the team, if it could be usefull in any way. I am happy with audacious for so many years now, maybe I can give at least a little thing back.

Would it be usefull to provide a git patch and if yes, who would like to receive the patch?

Regards
Michael

Hi Michael,

The author of the scrobbler plugin hasn't been doing any work on
Audacious recently, so it's more or less orphaned at the moment. The
best way to submit a patch is to create a feature request on the bug
tracker and attach it there. I can't promise I will get to it
immediately, but I hope to have more time for such things this summer.

-- John

History

#1 - May 19, 2012 19:30 - John Lindgren

This line seems wrong:
tuple_set_int(submit_tuple, FIELD_LENGTH, NULL, 240000);

Why should you artificially set the length to 4m00s?

#2 - May 19, 2012 20:25 - Michael Jacobs

For the different titles in the same stream, we do initially not know how long they are. So initially, the length is set to an undefined value:

tuple_get_int(submit_tuple, FIELD_LENGTH, NULL) <= 0

In general, we will not be able to determine the length of the title until the next title starts (which is noticed by a change in the title or artist tag field). So my first attempt was to keep the length at the initial value or assign it a very small value before we know its real length.

But then I observed that the length value that is submitted as now playing to last.fm seems to have an influence on how long the track is displayed as "now playing" on last.fm. This means that every track was only shown as "now playing" at last.fm for a few seconds and then disappeared again, which is clearly not as intended.

So I decided for the following work-around:

- Initially assume a length of four minutes.
- If the title is shorter than four minutes, then this is no problem. The title's real length is obtained from the amount of time that the title was playing at the moment when the next title starts. The old title information together with its length found out is then put into the list of tracks that were listened to completely. () The now playing information for the new title is then immediately submitted.
- The problematic case is when the title played is longer than four minutes. So after the currently approximated length is over (np_item->timeplayed > np_item->len), another now playing request (with the same length) is sent again and the length is set to the double of the value. This means after for 4 minutes another now playing of four minutes is sent. After 8 minutes another now playing of 8 minutes and so on. This was just an easy possiblility to not have to send a now playing request too often.

The crucial part is that the now playing request has no influence on the list of songs really listened to but only controls the now playing information displayed in the last.fm profile. So initially overapproximating is uncritical as the next now playing request for another title will overwrite the old one is needed. Initially underapproximating is also uncritical as we may simply approximate the doubled value as soon as the currently approximated value is too short.

In my tests, this leaded to an appropriate treatment of the now playing information in my last.fm profile.

Another possibility would be to first send a request to the last.fm title db to query the real lenth of the title when it starts already. But there are a few problems:

- Titles so far not known to last.fm would be treated wrong
- If the title has a different length then the same title in the last.fm db (an alternate version or something), then it could possibly be shown as now playing for a too short period of time. But the intention of the now playing field is too show the title exactly as long it is really played.

#3 - May 24, 2012 01:08 - John Lindgren

I see now why you would submit a bogus length to last.fm, but changing the length in the tuple will change it in the Audacious playlist also. That's not

what you intended, is it?

#4 - May 24, 2012 13:44 - Michael Jacobs

You are right, that is not what I intended. I just had a short look at the sources again and it seems I did not quite understand the meaning of aud_playlist_entry_get_tuple. I thought it would create an independent copy of the information in the playlist.

I did not realize that this could be the case as, at least with my version here, in the user interface there is no length displayed for streams at all.

I agree with you, that this is surely not the best way. Maybe one should use a plugin-local variable to do the accumulated changing of the title length in and replace the use of the tuple in all places by this?

On the other hand, if it is clear, that streams will most likely not starting to provide some meaningful length tag information, then user interfaces will also most likely not start to display that value and there will be nobody noticing the bogus value inside the length field. And before the hack, there was also a bogus value in that field, namely something <= 0.

It's clearly a matter of style, how to solve this. Both ways are not optimal and I wonder if it is worth rewriting from one to ther other.

I think, on the long run, the information in the playlist tuple of audacious should anyway become more aware of the possibility of stream entries. We should not have to use our own ishttp helper function. Maybe there should one day be an own entry sub type for stream entries and and one for file based entries. The length field would then only exist for file based entries and we could not even try to put a bogus value in there.

#5 - May 25, 2012 01:57 - John Lindgren

If you want to make a copy of the tuple data locally, there is tuple_copy(). As for expecting the user interface to not display a bogus value, I think that is a bad idea. We exclude 0 and -1 from being displayed because some plugins used to treat those as reserved values meaning "unknown length", but even that behavior should not be relied on.

If you want to make a second version of the patch, I will be happy to look it over.

#6 - June 02, 2012 19:05 - Michael Jacobs

- File 0002-Created-a-local-copy-of-playlist-entry-tuples-before.patch added

Yes, you are right. tuple_copy seems to be the right thing in that situation.

The enclosed incremental patch should fix that.

#7 - June 16, 2012 23:30 - John Lindgren

- File scrobble-http.diff added

Attaching a combined patch against current Git. Still needs review.

#8 - June 16, 2012 23:36 - John Lindgren

This block of code:

```
    if (submit_tuple)
        tuple_unref (submit_tuple);
    {
        Tuple *temp_tuple = aud_playlist_entry_get_tuple (playlist, pos, FALSE);
        if (! temp_tuple)
            return;
        submit_tuple = tuple_copy (temp_tuple);
        tuple_unref (temp_tuple);
    }
```

has multiple problems. First, the block enclosed by braces looks like it is attached to the "if" condition, but it is not, which is highly confusing. Second, if you are going to unref() submit_tuple whenever it is not null, you must be sure to set it to null after you unref() it; otherwise, you may try to unref() it again.

#9 - June 16, 2012 23:46 - John Lindgren

Also, you set item->is_http_source to FALSE in create_item(), and then set it to its proper value after create_item() returns. It would be clearer to pass the proper value to create_item().

#10 - June 17, 2012 18:31 - Michael Jacobs

- File scrobble-http-2.diff added

Thanks for the feedback.

@ #8:
You are right, that the braces directly after the if-statement are a bit confusing. I'd personally just add a blank line between the if statement and the opening brace.

In fact, the second point you mention is also related to this. You are right about the thing that temp_tuple should in general be set to null after unreferencing it. However, in this special case, I omitted it, as the scope of temp_tuple ends anyway after the unref and as the only purpose of the braces was to wrap the temporal variable and prohibit its usage at other positions in the method.

If you, however, do not like this style of wrapping temporal variables in own braces, I can as well drop the extra scope braces and assign null to temp_tuple after the unref. I prepared an alternative version in the patch file.

@ #9:
I agree, seems like a good idea. I added this.

#11 - June 18, 2012 01:09 - Ariadne Conill

hi,

last.fm does not allow scrobbling http streams like shoutcast, they will revoke audacious's privileges to scrobble if we implement that.

how do you prevent scrobbling shoutcast streams?

#12 - June 18, 2012 22:30 - Michael Jacobs

Hi William,

in fact, I was not aware that some streams are not allowed to be scrobbled.

There is currently no solution to this problem.

I will first try to do a little web research on this topic. Could you please provide me your source of information?

The thing that surprises me a bit is that other players seem to also scrobble from streams. Maybe I will have to take a look if and how they treat this problem.

#13 - June 19, 2012 00:32 - John Lindgren

Michael Jacobs wrote:

> In fact, the second point you mention is also related to this. You are right about the thing that temp_tuple should in general be set to null after unreferencing it. However, in this special case, I omitted it, as the scope of temp_tuple ends anyway after the unref and as the only purpose of the braces was to wrap the temporal variable and prohibit its usage at other positions in the method.

Read what I said again. temp_tuple is not the problem, it's submit_tuple that you need to set to null.

#14 - June 19, 2012 00:39 - John Lindgren

William Pitcock wrote:

> last.fm does not allow scrobbling http streams like shoutcast, they will revoke audacious's privileges to scrobble if we implement that.

Are you sure this is still the case? I skimmed Last.fm's API documentation (http://www.last.fm/api/scrobbling) and their terms of service (http://www.last.fm/api/tos) and didn't see anything mentioned.

#15 - June 19, 2012 23:57 - Michael Jacobs

- File scrobble-http-3.diff added

John Lindgren wrote:

> Read what I said again. temp_tuple is not the problem, it's submit_tuple that you need to set to null.

You are right. I changed this. I also decided to reintroduce the extra braces again, as the additional braces needed now at the if statement anyway should prevent further confusion.

#16 - June 20, 2012 02:02 - John Lindgren

- File scrobbler-http-revised.diff added

This is looking good. Here is a revised version of the patch that gets around the whole messy changing-the-song-length-in-the-tuple by using a separate length variable. Would you look it over and make sure it still works?

#17 - June 21, 2012 21:03 - Michael Jacobs

- File scrobbler-http-revised-2.diff added

Thanks for your work. It is a real simplification and almost worked out of the box.

The only thing missing was a call to "sc_idle((GMutex*)data)" just after adding the previous now playing track to the submit queue. Without that call, the track is only added to the local file based queue. That queue is not flushed to the last.fm profile before sc_idle is called. Therefore I also had to add the parameter (gpointer)m_scrobbler again to the initialization of the timeout for the callback function.

A few hints about the behavior w.r.t. queue dumping and qeue submission:

For non-stream-tracks, that call to sc_idle is done in aud_hook_playback_end, which is effectively never reached as long as playing a single stream track. This would imply that for stream tracks, we would only see the current now playing and none of the previously played tracks of the stream would be listed as definitely listened to before we stop the stream. I guess this is not what one would expect or wish.

The point that normal tracks are already put to the scrobble queue after they have played at least half or 4 minutes is just to prevent tracks that have already played long enough to be scrobbled from not being scrobbled by a seq-fault or something happening later but before the end of the track. For stream tracks, however, we cannot do this for the half of the song, as we do not know the real length. I also decided to not do it after 4 minutes, as this would be too early to have found the "real" track length.

also note that the first track of a stream may auto-detect a wrong length, as the stream was started too early to get the whole track, but still early enough to get more than 30 second and therefore the track is scrobbled anyway.

#18 - June 22, 2012 03:00 - John Lindgren

- Status changed from New to Closed

- Target version set to 3.3

- % Done changed from 0 to 100

- Affects version deleted (3.3)

Okay, I've applied the updated patch. Thanks for your work!
https://github.com/audacious-media-player/audacious-plugins/commit/181422f3f5701165fc522f787ec75f47ce79e12c

Files

0001-Added-very-basic-support-for-scrobbling-of-data-obta.patch	7.59 KB	May 08, 2012	Michael Jacobs
0002-Created-a-local-copy-of-playlist-entry-tuples-before.patch	2.23 KB	June 02, 2012	Michael Jacobs
scrobble-http.diff	7.5 KB	June 16, 2012	John Lindgren
scrobble-http-2.diff	7.99 KB	June 17, 2012	Michael Jacobs
scrobble-http-3.diff	8.08 KB	June 19, 2012	Michael Jacobs
scrobbler-http-revised.diff	8.75 KB	June 20, 2012	John Lindgren
scrobbler-http-revised-2.diff	8.81 KB	June 21, 2012	Michael Jacobs