Audacious - Bug #97

ABI breakage in audacious 3.2.2

April 06, 2012 08:58 - Alessio Treglia

Status:	Rejected	**Start date:**	April 06, 2012
Priority:	Major	**Due date:**	
Assignee:		**% Done:**	0%
Category:	libaudcore	**Estimated time:**	0.00 hour
Target version:			
Affects version:	3.2.2		

Description

Hi,

Audacious 3.2.2 introduces a ABI breakage in libaudcore1, some symbols has been removed from the public interface:

- convert_table@Base 2.3
- tuple_evalctx_free@Base 2.3
- tuple_evalctx_new@Base 2.3
- tuple_evalctx_reset@Base 2.3
- tuple_evalnode_free@Base 2.3
- tuple_formatter_compile@Base 2.3
- tuple_formatter_eval@Base 2.3
- tuple_formatter_process_string@Base 2.3
- vfs_async_file_get_contents_trampoline@Base 2.4~beta1
- vfs_async_file_get_contents_worker@Base 2.4~beta1
- _aud_api_table@Base 2.4~beta1

[many others follow]

Please update the SONAME of libaudcore accordingly.
Thanks for considering.

History

#1 - April 07, 2012 16:41 - Michael Schwendt

Notice last comment in bug #86. Prior to 3.2.2, many symbols have been made public by accident. They are non-public (and not part of the /usr/include/audacious/* API either), however, so nothing should depend on them.

#2 - April 08, 2012 16:17 - John Lindgren

- Status changed from New to Rejected

Exactly as Michael says, these symbols were not intended to be part of the API, so there should be no breakage. If there is something linking to these symbols, it is a bug in that program or plugin.

#3 - April 12, 2012 01:55 - Alessio Treglia

Sorry guys, but I don't agree.
Although those symbols were not part of any public API, they were publicly exported by the library and removing them definitely means breaking ABI: no one can guarantee that applications previously linked against libaudcore still work fine without recompiling.

So, please, update the SONAME: that would be useful to push your downstreams to recompile libaudcore-dependent stuff, which is the right way in these cases.

Thanks in advance for any reply.

#4 - April 14, 2012 00:21 - Michael Schwendt

FWIW, I'm downstream for the Fedora packages of Audacious. We rebuild only if the API version of Audacious requires it, or if the ABI is affected by a compiler upgrade, for example.

> no one can guarantee that applications previously linked against libaudcore still work fine without recompiling.

The API decides what those applications may use and what they will depend on. They cannot (and must not) access library internals by bypassing the API definition. Only if they did that, they would fail to link at run-time due to the changed symbol visibility.

#5 - April 16, 2012 22:07 - John Lindgren

Alessio Treglia wrote:

> Although those symbols were not part of any public API, they were publicly exported by the library and removing them definitely means breaking ABI:

Um, no. And no.

> no one can guarantee that applications previously linked against libaudcore still work fine without recompiling.

If there are applications out there using symbols from libaudcore that are not part of the public API, then yes, they will break, and it will be their own damn fault.

#6 - April 17, 2012 00:26 - Alessio Treglia

John Lindgren wrote:

> Alessio Treglia wrote:
>
> > Although those symbols were not part of any public API, they were publicly exported by the library and removing them definitely means breaking ABI:
>
> Um, no. And no.

Oh come on, the meaning of "ABI" does not depend on what ABI means for you: removing symbols breaks ABI and changing the SONAME implies that programs using libaudcore need to be recompiled.

> If there are applications out there using symbols from libaudcore that are not part of the public API, then yes, they will break, and it will be their own damn fault.

Surely you can neglect that applications using those symbols won't work anymore: as Debian Developer I can't.
I must take care of Debian's archive integrity as well as the interests of Debian users, hence I won't introduce anything which could potentially break local user applications.

I'll sit and wait to see what you decide to do on this matter.
Thanks for your time and work.

#7 - April 17, 2012 01:22 - Michael Schwendt

It would be unnecessarily pedantic in this case to break the ABI by changing the SONAME, because the relevant combination of API+ABI hasn't changed. On the contrary, symbol collision in Audacious lower than 3.2.2 leads to real problems.

#8 - April 17, 2012 15:09 - John Lindgren

Alessio Treglia wrote:

> Surely you can neglect that applications using those symbols won't work anymore: as Debian Developer I can't.
> I must take care of Debian's archive integrity as well as the interests of Debian users, hence I won't introduce anything which could potentially break local user applications.

I see your point about not wanting to break anything, but I think you are being overcautious. Touching the private symbols in libaudcore is almost guaranteed to lead to undefined behavior, so any application using them is already broken. In fact, it would be better for those applications (supposing that they exist -- which, by the way, I doubt) to fail to load because of undefined symbols than to load and then crash.

#9 - April 18, 2012 10:24 - Alessio Treglia

Hi again John et al,

I apologize to have been rude, the great work done on Audacious is very appreciated, many users of Debian and Ubuntu install and use audacious everyday and they are happy for that.

I agree this is not a big concern since those symbols were not part of any public API, but they were removed from the application binary interface and that still makes this a bug and should be resolved.

If you do not intend to change the SONAME, I'll patch Audacious in Debian to manage the transition to the new soname'd library.

Thanks for your collaboration, keep up the good work!