

Ruby - Bug #20319

Singleton class is being frozen lazily in some cases

03/01/2024 11:52 AM - andrykonchin (Andrew Konchin)

Status:	Open	Backport: 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN
Priority:	Normal	
Assignee:		
Target version:		
ruby -v:	3.2.2	
Description		
I've noticed suspicious behaviour (it doesn't affect anything in practice for me though) when an object becomes frozen only its own singleton class becomes frozen immediately. A singleton class of the object immediate singleton class becomes frozen lazily after #singleton_class method call: <pre>o = Object.new klass = o.singleton_class.singleton_class o.freeze puts klass.frozen? # false <== here we expect true puts o.singleton_class.singleton_class.frozen? # true puts klass.frozen? # true</pre> I would expect all created (and visible to user) singleton classes in an object singleton classes chain to become frozen immediately when the object gets frozen.		

History

#1 - 03/01/2024 11:52 AM - andrykonchin (Andrew Konchin)

- Description updated

#2 - 03/01/2024 11:53 AM - andrykonchin (Andrew Konchin)

- Description updated

#3 - 03/01/2024 12:04 PM - Eregon (Benoit Daloze)

Alternatively, I think we could simplify Kernel#freeze to always only freeze the direct object and no other object, i.e. no singleton class would be frozen when freezing a singleton object.
State and methods are separate things, so IMO it doesn't gain anything to freeze singleton classes of a singleton object on singleton_object.freeze.
The methods can still be modified e.g. using singleton_object.class.class_exec { def foo = 42 }.

I think Kernel#freeze should only freeze the object itself, which means instance variables (and internal state if core/TypedStruct object) cannot be changed.
That would be both more consistent semantically and more efficient, while avoiding weird cases like the above.

#4 - 03/04/2024 05:46 PM - jeremyevans0 (Jeremy Evans)

Eregon (Benoit Daloze) wrote in [#note-3](#):

I think Kernel#freeze should only freeze the object itself, which means instance variables (and internal state if core/TypedStruct object) cannot be changed.
That would be both more consistent semantically and more efficient, while avoiding weird cases like the above.

I disagree. If you do not freeze the object's singleton class, then you can define or undefine any method in the singleton class, which is almost the same as being able to modify the object (from a Ruby perspective, not a C perspective).
However, redefining methods in the singleton class of the singleton class of the object does not allow you to modify the object, it only allows you to modify the object's singleton class. However, as the object's singleton class is frozen, we still should prevent it.

This bug should be fairly simple to fix by having Kernel#freeze go up the singleton class chain and freeze all currently instantiated singleton classes.

#5 - 03/11/2024 01:35 PM - Eregon (Benoit Daloze)

jeremyevans0 (Jeremy Evans) wrote in [#note-4](#):

I disagree. If you do not freeze the object's singleton class, then you can define or undefine any method in the singleton class, which is almost the same as being able to modify the object (from a Ruby perspective, not a C perspective).

One can still define or undefine any method in one of the ancestors (except the singleton class), or use prepend/include on any of these or even refine to change the behavior of that object.

Freezing the singleton class achieves very little IMO and is inconsistent with Kernel#freeze being a shallow freeze (that is, only freeze the immediate object, not other objects, which AFAIK holds for all other cases).

#6 - 03/11/2024 01:37 PM - Eregon (Benoit Daloze)

This bug should be fairly simple to fix by having Kernel#freeze go up the singleton class chain and freeze all currently instantiated singleton classes.

Yeah, that seems the smaller fix for now and at least it fixes a "should-be-idempotent method like singleton_class" dynamically freezing an existing singleton class.

Would you be able to make a PR for that?

#7 - 03/14/2024 01:37 AM - jeremyevans0 (Jeremy Evans)

I looked into this and the current behavior is deliberate, as there is a comment stating should not propagate to meta-meta-class (see rb_freeze_singleton_class definition in class.c). See related commit [d9a597408f0f192ff25ab51e1e6733fe6f6c01b](#).

Freezing the singleton class of the singleton class breaks TestModule#test_frozen_singleton_class. I came up with a solution that fixes this by walking down the attached object chain to make sure it ends in the object being frozen. Not sure that is correct, but it does appear to pass CI: <https://github.com/ruby/ruby/pull/10245>

#8 - 04/18/2024 11:44 AM - Eregon (Benoit Daloze)

[@nobu \(Nobuyoshi Nakada\)](#) WDYT? Could you review that PR?

#9 - 06/06/2024 09:41 AM - mame (Yusuke Endoh)

Discussed at the dev meeting, but no one had a deep understanding of Jeremy's patch, and matz was unable to make any decision.