Ruby - Bug #21330

Namespace: Class and Module frozen status is not namespaced

05/13/2025 09:56 AM - byroot (Jean Boussier)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: UNKNOWN

Description

```
File.write("/tmp/test.rb", <<~'RUBY')
  Hash.freeze
RUBY

ns = Namespace.new
ns.require("/tmp/test.rb")

class Hash
  def monkey_patch
  end
end
```

Expected behavior:

Since the monkey patch is in a different namespace, I'd expect it to not impact code that is running in another namespace.

Actual behavior:

```
test.rb:9:in '<class:Hash>': can't modify frozen class: Hash (FrozenError)
```

The class is frozen globally, breaking code in other namespaces.

Is this by design, or does that mean the frozen status need to be moved in the classext_t as well?

cc @tagomoris (Satoshi Tagomori)

Associated revisions

Revision 45a2c95d - 06/24/2025 10:29 AM - byroot (Jean Boussier)

Reduce exposure of FL_FREEZE

The FL_FREEZE flag is redundant with SHAPE_ID_FL_FROZEN, so
ideally it should be eliminated in favor of the later.

Doing so would eliminate the risk of desync between the two, but
also solve the problem of the frozen status being global in namespace
context (See Bug #21330).

History

#1 - 05/13/2025 10:08 AM - byroot (Jean Boussier)

- Subject changed from Namespace: Class and Module frozen status is not namespace to Namespace: Class and Module frozen status is not namespaced

#2 - 05/13/2025 12:00 PM - tagomoris (Satoshi Tagomori)

The frozen flag is what I missed. I think we should move the flag to rb_classext_t.

#3 - 06/03/2025 05:46 AM - hsbt (Hiroshi SHIBATA)

- Tags set to namespace

#4 - 06/03/2025 06:01 AM - ko1 (Koichi Sasada)

I'm afraid that moving frozen flag into class_ext (and each built-in class variants can has each frozen state) makese OBJ_FROZEN code:

```
// include/ruby/internal/fl_type.h
static inline VALUE
RB_OBJ_FROZEN_RAW(VALUE obj)
{
    return RB_FL_TEST_RAW(obj, RUBY_FL_FREEZE);
}
```

especially, if nobody freeze builtin classes. Any usecases?

#5 - 06/03/2025 06:08 AM - jeremyevans0 (Jeremy Evans)

ko1 (Koichi Sasada) wrote in #note-4:

> I'm afraid that moving frozen flag into class_ext (and each built-in class variants can has each frozen state) makese OBJ_FROZEN code:
>
> ```
> // include/ruby/internal/fl_type.h
> static inline VALUE
> RB_OBJ_FROZEN_RAW(VALUE obj)
> {
>     return RB_FL_TEST_RAW(obj, RUBY_FL_FREEZE);
> }
> ```
>
> especially, if nobody freeze builtin classes. Any usecases?

roda-sequel-stack (https://github.com/jeremyevans/roda-sequel-stack) supports and recommends freezing all core classes at runtime (after the application is loaded). This is how production Roda applications are commonly run. Freezing core classes is recommended to ensure that no libraries the application is using are modifying the core classes at runtime (they probably aren't, but how can you be sure without freezing them)?

#6 - 06/03/2025 06:30 AM - byroot (Jean Boussier)

> I'm afraid that moving frozen flag into class_ext (and each built-in class variants can has each frozen state) makese OBJ_FROZEN code:

It seems like your sentence is missing a word. Did you mean to say that it will make OBJ_FROZEN slower? If so yes, absolutely, but I thought it was a given. Namespaces are introducing a level of indirection it need to be checked and isn't cheap.

But on this note I'm currently working on two patches that may help with that:

- Moving the frozen bit inside the shape_id: https://github.com/ruby/ruby/pull/13289
- Delegating classes variables handling: https://github.com/ruby/ruby/pull/13411

With those two changes, once finished, I believe I can come up with a RB_OBJ_FROZEN_RAW that has almost the exact same performance.

> Any usecases?

What Jeremy said, but also just consistency? It's very hard to reason about a feature that isn't behaving consistently.

#7 - 06/24/2025 10:29 AM - byroot (Jean Boussier)

- Status changed from Open to Closed

Applied in changeset git|45a2c95d0f7184c9cd64ddd26699af31bea8675d.

Reduce exposure of FL_FREEZE

The FL_FREEZE flag is redundant with SHAPE_ID_FL_FROZEN, so
ideally it should be eliminated in favor of the later.

Doing so would eliminate the risk of desync between the two, but
also solve the problem of the frozen status being global in namespace
context (See Bug #21330).