Ruby master - Feature #16175

Object#clone(freeze: true) is inconsistent with Object#clone(freeze: false)

09/21/2019 05:33 PM - zverok (Victor Shepelev)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

In #12300, the new keyword freeze: was introduced, allowing this:

```
h = {}.freeze
h.clone.frozen?
# => true
h.clone(freeze: false).frozen?
# => false
```

Though, it turns to me that behavior is not symmetric:

```
h = {}
h.frozen?
# => false
h.clone.frozen?
# => false
h.clone(freeze: true).frozen?
# => false -- I expected true here!
```

I wonder, if it is "by design" and should be addressed in docs, or just an implementation inconsistency that can be fixed?

History

#1 - 09/21/2019 08:26 PM - jeremyevans0 (Jeremy Evans)

- Backport deleted (2.5: UNKNOWN, 2.6: UNKNOWN)

- Subject changed from Object#clone(freeze: true) to Make Object#clone(freeze: true) return frozen clone even if receiver is not frozen

- Tracker changed from Bug to Feature

The freeze: false option was intended to be: "do not freeze clone if receiver is already frozen". The Object#clone documentation states: "#clone copies the frozen (unless :freeze keyword argument is given with a false value) and tainted state of obj". The behavior when freeze: true is provided is therefore unspecified, and therefore, I do not think this should be considered a bug.

The reason freeze: false was introduced is because you previously could not use clone with frozen objects with singleton classes/extended modules and keep the ability to modify copies of the frozen objects. Adding freeze: false made something previously impossible in Ruby possible. Adding freeze: true for consistency doesn't enable new behavior, as you can use clone.freeze instead of clone(freeze: true). That being said, I'm not opposed to freeze: true being supported.

#2 - 09/22/2019 09:30 AM - shevegen (Robert A. Heiler)

I sort of agree with zverok at the least for the expectation of freeze: true working. IMO for boolean toggle-values I would think it is simpler to have both variants work; then again I don't think I need either of the two variants myself. It's interesting to point out that freeze adds a bit of complexity though.

#3 - 09/22/2019 09:59 AM - zverok (Victor Shepelev)

- Backport set to 2.5: UNKNOWN, 2.6: UNKNOWN

- Subject changed from Make Object#clone(freeze: true) return frozen clone even if receiver is not frozen to Object#clone(freeze: true) is inconsistent with Object#clone(freeze: false)

- Tracker changed from Feature to Bug

- File freeze-true.patch added

jeremyevans0 (Jeremy Evans) thanks for your answer.

Let me explain my point a bit.

I come upon this inconsistency (or what **I** see as an inconsistency) when working on 2.4 version of my RubyChanges project. Typically, when going through the language version's changelog and trying to rationalize it, I use to find some less-documented changes which makes me providing documentation patches and clarification tickets.

So, from where I am standing, the situation looks as follows:

1. The behavior is definitely an *inconsistency* (either behavioral or at least in documentation), I do believe PoLS should lead to having the symmetry or at least documenting of the lack of it.
2. As for three years since Ruby 2.4 I am the first to notice it (and even this is going through the changelog, not writing code), the problem is of pretty low priority.
3. Though, I don't really believe that having such an inconsistency is really justifiable, especially considering recent uprising of interest to immutability (and therefore freezing).
4. I don't think that "freeze cloned can be achieved with clone.freeze, therefore clone(freeze: true) is redundant" is a strong argument. For me, it looks a bit like saying "we can achieve subtraction with x.+(-y), therefore x.-(y) is redundant": clone(freeze: true) looks atomic, logical and readable, I don't see why it shouldn't work.
5. OK, "We don't need it because there are other ways to achieve it" could be justifiable **if** fixing the inconsistency requires significant effort (in planning and/or implementation). For what I can see, the change is pretty trivial. Attached is the patch implementing it. It is not the prettiest code possible, but at least it allows to estimate an amount of effort.

PS: I hope you will not find completely inappropriate that I've changed back the type of the issue to "Bug" and rephrased the title to show WHY I consider it so. From my perspective, it is not me requesting some new weird feature, but trying to straighten up some obvious inconsistency (even if a low-priority one).

#4 - 09/22/2019 10:52 AM - mame (Yusuke Endoh)

- Backport deleted (2.5: UNKNOWN, 2.6: UNKNOWN)

- Tracker changed from Bug to Feature

I agree with Jeremy. It is not a bug.

I think that the wording of freeze: false is a bit confusing, but the document explains clearly. Do you have a real-world use case for clone(freeze: true)?

Anyway, please do not modify the tracker. Bug/Feature is not significant for you. It is mainly used as an advice for branch maintainers: the change should be backported or not.

#5 - 09/23/2019 11:23 PM - jeremyevans0 (Jeremy Evans)

- File clone-freeze-true-16175.patch added

Attached is an alternative approach for implementing this. It uses VALUE for the kwfreeze variable, so we can use Qundef instead of the magic value of -1 for the default behavior. It also updates the documentation to reflect you can use true or false.

Technically, this feature breaks backwards compatibility, because freeze: true did not freeze clones of unfrozen receivers previously. However, it seems unlikely that someone would rely on that behavior.

Files

freeze-true.patch	1.47 KB	09/22/2019	zverok (Victor Shepelev)
clone-freeze-true-16175.patch	4.42 KB	09/23/2019	jeremyevans0 (Jeremy Evans)