Ruby - Bug #20250

Crash with "Object ID seen, but not in mapping table: proc" error

02/09/2024 10:11 AM - zetter (Chris Zetter)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.3.0 (2023-12-25 revision 5124f9ac75) [arm64-darwin23]	**Backport:**	3.0: WONTFIX, 3.1: REQUIRED, 3.2: DONE, 3.3: DONE

Description

Hello, I experienced a crash which I was able to reliably reproduce with the following:

```
require 'bundler/inline'

gemfile(true) do
  source 'https://rubygems.org'
  gem 'activesupport', '7.1.3'
end

require 'active_support'

logger = ActiveSupport::Logger.new('log/log.log', 1, 100 * 1024 * 1024)

logger.formatter = proc {|_, _, _, message| "#{message}\n" }

logger = ActiveSupport::TaggedLogging.new(logger)

logger.tagged("TAG").info "hello"
logger.tagged("TAG").info "hello" # usually crashes here
GC.start # sometimes need to trigger crash with GC
```

It looks like this is caused by the interaction with the formatter proc and the tagged logging features of activesupport. Let me know if a more minimal example would be useful.

I can reproduce this with:

- ruby 3.3.0 on arm64-darwin23
- ruby 3.3.0 on x86_64-linux
- latest ruby HEAD (5e12b75716) on arm64-darwin23

I cannot reproduce this on ruby 3.2.2.

Thanks for any help

Related issues:

Related to Ruby - Bug #20253: `Proc.dup` and `Proc#clone` don't preserve fina...	**Closed**

Associated revisions

Revision d19d683a354530a27b4cbb049223f8dc70c75849 - 02/09/2024 04:38 PM - byroot (Jean Boussier)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

[Bug #20250]

We're seting up a new instance, so it never had an associated
object_id.

Revision d19d683a354530a27b4cbb049223f8dc70c75849 - 02/09/2024 04:38 PM - byroot (Jean Boussier)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

[Bug #20250]

We're seting up a new instance, so it never had an associated
object_id.

Revision d19d683a - 02/09/2024 04:38 PM - byroot (Jean Boussier)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

[Bug #20250]

We're seting up a new instance, so it never had an associated
object_id.

Revision de1a586ecc2ee7f465f0c0a69291054136a3a819 - 02/12/2024 05:31 PM - byroot (Jean Boussier)

proc.c: get rid of CLONESETUP

[Bug #20253]

All the way down to Ruby 1.9, Proc, Method, UnboundMethod
and Binding always had their own specific clone and dup routine.

This caused various discrepancies with how other objects behave
on dup and `clone. [Bug #20250], [Bug #20253].

This commit get rid of CLONESETUP and use the the same codepath
as all other types, so ensure consistency.

NB: It's still not accepting the freeze keyword argument on clone.

Co-Authored-By: Étienne Barrié etienne.barrie@gmail.com

Revision de1a586ecc2ee7f465f0c0a69291054136a3a819 - 02/12/2024 05:31 PM - byroot (Jean Boussier)

proc.c: get rid of CLONESETUP

[Bug #20253]

All the way down to Ruby 1.9, Proc, Method, UnboundMethod
and Binding always had their own specific clone and dup routine.

This caused various discrepancies with how other objects behave
on dup and `clone. [Bug #20250], [Bug #20253].

This commit get rid of CLONESETUP and use the the same codepath
as all other types, so ensure consistency.

NB: It's still not accepting the freeze keyword argument on clone.

Co-Authored-By: Étienne Barrié etienne.barrie@gmail.com

Revision de1a586e - 02/12/2024 05:31 PM - byroot (Jean Boussier)

proc.c: get rid of CLONESETUP

[Bug #20253]

All the way down to Ruby 1.9, Proc, Method, UnboundMethod
and Binding always had their own specific clone and dup routine.

This caused various discrepancies with how other objects behave
on dup and `clone. [Bug #20250], [Bug #20253].

This commit get rid of CLONESETUP and use the the same codepath
as all other types, so ensure consistency.

NB: It's still not accepting the freeze keyword argument on clone.

Co-Authored-By: Étienne Barrié etienne.barrie@gmail.com

Revision a63e979853783601a60228b45741f8b3776e5507 - 03/21/2024 01:45 AM - NARUSE, Yui

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849,de1a586ecc2ee7f465f0c0a69291054136a3a819: [Backport #20250] (#10308)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

```
    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.

    proc.c: get rid of `CLONESETUP`
    MIME-Version: 1.0
    Content-Type: text/plain; charset=UTF-8
    Content-Transfer-Encoding: 8bit

    [Bug #20253]

    All the way down to Ruby 1.9, `Proc`, `Method`, `UnboundMethod`
    and `Binding` always had their own specific clone and dup routine.

    This caused various discrepancies with how other objects behave
    on `dup` and `clone. [Bug #20250], [Bug #20253].

    This commit get rid of `CLONESETUP` and use the the same codepath
    as all other types, so ensure consistency.

    NB: It's still not accepting the `freeze` keyword argument on `clone`.

    Co-Authored-By: Étienne Barrié <etienne.barrie@gmail.com>
```

Revision a63e979853783601a60228b45741f8b3776e5507 - 03/21/2024 01:45 AM - NARUSE, Yui

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849,de1a586ecc2ee7f465f0c0a69291054136a3a819: [Backport #20250] (#10308)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

```
    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.

    proc.c: get rid of `CLONESETUP`
    MIME-Version: 1.0
    Content-Type: text/plain; charset=UTF-8
    Content-Transfer-Encoding: 8bit

    [Bug #20253]

    All the way down to Ruby 1.9, `Proc`, `Method`, `UnboundMethod`
    and `Binding` always had their own specific clone and dup routine.

    This caused various discrepancies with how other objects behave
    on `dup` and `clone. [Bug #20250], [Bug #20253].

    This commit get rid of `CLONESETUP` and use the the same codepath
    as all other types, so ensure consistency.

    NB: It's still not accepting the `freeze` keyword argument on `clone`.

    Co-Authored-By: Étienne Barrié <etienne.barrie@gmail.com>
```

Revision a63e9798 - 03/21/2024 01:45 AM - NARUSE, Yui

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849,de1a586ecc2ee7f465f0c0a69291054136a3a819: [Backport #20250] (#10308)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

```
    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.

    proc.c: get rid of `CLONESETUP`
    MIME-Version: 1.0
    Content-Type: text/plain; charset=UTF-8
    Content-Transfer-Encoding: 8bit

    [Bug #20253]

    All the way down to Ruby 1.9, `Proc`, `Method`, `UnboundMethod`
    and `Binding` always had their own specific clone and dup routine.

    This caused various discrepancies with how other objects behave
    on `dup` and `clone. [Bug #20250], [Bug #20253].

    This commit get rid of `CLONESETUP` and use the the same codepath
    as all other types, so ensure consistency.

    NB: It's still not accepting the `freeze` keyword argument on `clone`.

    Co-Authored-By: Étienne Barrié <etienne.barrie@gmail.com>
```

Revision 584a02aaafda74c21d24dc4c5e223a2482c7fde3 - 07/13/2024 06:17 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849: [Backport #20250]

```
    rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.
```

Revision 584a02aaafda74c21d24dc4c5e223a2482c7fde3 - 07/13/2024 06:17 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849: [Backport #20250]

```
    rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.
```

Revision 584a02aa - 07/13/2024 06:17 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849: [Backport #20250]

```
    rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.
```

History

#1 - 02/09/2024 12:30 PM - byroot (Jean Boussier)

I had a quick look and this is very interesting. As far as I can tell cached_object_id for this Proc, so the most likely explanation is that some other part of the codebase is messing with the object flags.

I'll try to dig deeper.

#2 - 02/09/2024 12:42 PM - byroot (Jean Boussier)

Alright, it's a bug in #clone, I managed to reduce it to:

```
proc = Proc.new { }
proc.object_id
proc.clone
GC.start
```

Shouldn't be too hard to figure out. IIRC Proc#clone was modified recently, the flag that indicate the instance has an object_id need to be cleared on the cloned object.

#3 - 02/09/2024 01:31 PM - byroot (Jean Boussier)

- Backport changed from 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN to 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: REQUIRED

Patch: https://github.com/ruby/ruby/pull/9903

Also this bug affect all Ruby versions since the new object_id implementation back in 2.7, I got all of them to crash on it.

#4 - 02/09/2024 04:39 PM - byroot (Jean Boussier)

- Status changed from Open to Closed

Applied in changeset git|d19d683a354530a27b4cbb049223f8dc70c75849.

---

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

[Bug #20250]

We're seting up a new instance, so it never had an associated
object_id.

#5 - 02/09/2024 04:48 PM - byroot (Jean Boussier)

- Related to Bug #20253: `Proc.dup` and `Proc#clone` don't preserve finalizers added

#6 - 03/21/2024 06:59 AM - naruse (Yui NARUSE)

- Backport changed from 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: REQUIRED to 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: DONE

ruby_3_3 a63e979853783601a60228b45741f8b3776e5507 merged revision(s)
d19d683a354530a27b4cbb049223f8dc70c75849,de1a586ecc2ee7f465f0c0a69291054136a3a819.

#7 - 07/13/2024 06:18 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: DONE to 3.0: WONTFIX, 3.1: REQUIRED, 3.2: DONE, 3.3: DONE

ruby_3_2 584a02aaafda74c21d24dc4c5e223a2482c7fde3 merged revision(s) d19d683a354530a27b4cbb049223f8dc70c75849.

Files

ruby-2024-02-09-092829.ips	13.2 KB	02/09/2024	zetter (Chris Zetter)
output.log	142 KB	02/09/2024	zetter (Chris Zetter)
crash_test_2.rb	465 Bytes	02/09/2024	zetter (Chris Zetter)