

# Re: apparently undefined function called by macro

---

*Source:* <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2006-06/msg01400.html>

---

- *From:* Pascal Costanza <[pc@xxxxxxxxx](mailto:pc@xxxxxxxxx)>
  - *Date:* Tue, 27 Jun 2006 23:22:32 +0200
- 

Greg Bacon wrote:

```
(eval-when (:load-toplevel :compile-toplevel :execute)
  (defun id (x) x)
  )

(defmacro define-foo-class (name slots)
  `(defclass ,name ()
    ,(mapcar #'id slots)))

(define-foo-class bar
  ((baz :initarg :baz :accessor baz)
   (quux :initarg :quux :accessor quux)))
```

If I try to load it in a fresh image, clisp complains about undefined id:

```
[1]> (asdf:oos 'asdf:load-op :foo)
; loading system definition from foo.asd into #<PACKAGE ASDF0>
;; Loading file foo.asd ...
; registering #<SYSTEM FOO #x102A2839> as FOO
;; Loaded file foo.asd
;; Compiling file /tmp/asdf/packages.lisp ...
;; Wrote file /tmp/asdf/packages.fas
;; Loading file /tmp/asdf/packages.fas ...
;; Loaded file /tmp/asdf/packages.fas
;; Compiling file /tmp/asdf/foo.lisp ...
*** - FUNCTION: undefined function ID
[...]
```

Loading by hand, i.e., (progn (load "packages.lisp") (load "foo.lisp")), is no problem, and as you can guess from the commented lines, wrapping ID in EVAL-WHEN is also a workaround.

Well, hmm, my misunderstanding is more basic than ASDF or even the reader. After removing the initial IN-PACKAGE form from foo.lisp, COMPILE-FILE still complains about undefined ID.

## Re: apparently undefined function called by macro

This isn't related to packages or systems (only in the sense that systems can help you to solve this problem – see below).

Why is ID undefined? Please help me correct the error in my mental model.

Common Lisp distinguishes between interpretation and compilation. When code is interpreted, each top-level form is interpreted after another, which means that each form can rely on all effects being "correctly" produced by the previous forms. That is, a defun defines a function, defmacro defines a macro, a defvar defines a variable, and so on.

When code is compiled, this is not necessarily the case. It is the case that each form is processed by the compiler, and for example it is guaranteed that all macros are completely expanded and thus "removed" from the code, but it is not necessarily the case that the effects are produced. Typically, the effects will only be produced at runtime. So, for example, a defun only announces the presence of a function at compile-time (so that other parts of the code can be c