This msg is stored on {eris}<lispcore>internal>library>do-test.tedit
The tester is on {eris}<lispcore>internal>library>do-test.dcom.

The main entries are the following:

(DO-TEST *name forms*)
A test succeeds if the final *form* returns a non-nil result. *Name* is just the name which can be an atom or string; strings are preferred. Forms are presumed to be read with the Common Lisp reader in package XCL-TEST, which uses LISP and XCL. If a test fails or an error occurs during evaluation, a message is printed to *ERROR-OUTPUT*.

(DO-TEST-GROUP *name&options forms*)
For associating a group of tests. For instance, a group of tests may all require the same setup and cleanup. If there are any options (see below) then the CAR of *name&options* is the name and the CDR is a keyword/value list. All *forms* must be DO-TEST forms.

(EXPECT-ERRORS *error-types forms*)
Error-types is a list of errors that may occur while executing the *forms*. If one of the listed errors occurs, EXPECT-ERRORS returns (values t error-that-occurred), otherwise NIL. Normal use of this form is:

```
(DO-TEST "a test"
      (EXPECT-ERRORS (type-of-error)
            (THIS-FORM 'SHOULD 'ERROR)))
```

(DO-TEST-FILE *filename*)
Reads and executes a file of tests. All forms in the file are read before any are executed. The file should be clear text (clearput in TEdit) and terminate with a STOP. The format for test names is
Chap#[-sec#[-subsec#]]-comment.TEST

(CL-READFILE *filename*)
Reads all forms in *filename* and returns a list of them. This function is used by DO-TEST-FILE to read test files; test writers who want to see if their files are syntactically valid should first see if CL-READFILE will read them, then see if DO-TEST-FILE will execute them.

```
(DO-ALL-TESTS &key (results *test-batch-results*)
                   (patterns *test-file-pattern*)
                   (sysout-type nil)
                   (resume nil))
```
Calls DO-TEST-FILE on each file that matches *patterns*, which is a list of directory patterns, and prints the results to a new version of a file named *results*. If *results* is T, results are printed to the window where DO-ALL-TESTS is running. The header of the results file is a message of the date and time the tests are being run and the MAKESYSDATE of the sysout; if *sysout-type* is supplied, a line for it goes out too. If *resume* is non-NIL, DO-ALL-TESTS attempts to resume an interrupted test sequence, appending the results onto the latest version of *results*.

TEST-SETQ, TEST-DEFUN, TEST-DEFMACRO
These work like SETQ, DEFUN, and DEFMACRO, except that if they are executed within a DO-TEST-GROUP, their effects are manually undone (old values are saved and then restored) upon leaving the DO-TEST-GROUP. Use these in :BEFORE forms that a whole group of DO-TESTs want to see. DON'T use TEST-SETQ on locally-bound variables or in loops.

Relevant variables:

TEST-MODE
Default is :batch, which means to report test failures and errors on *ERROR-OUTPUT* (which is usually a file), and continue. Other values possible are:
:interactive which means to print a message before running each test, print another message for test failures, and produce a break window on errors.
:batch-verbose which means to generate all the messages of :interactive and do not break on errors.

TEST-BATCH-RESULTS
Defaults to "{eris}<lispcore>cml>test>test-results"

TEST-FILE-PATTERN

Defaults to ("{eris}<lispcore>cml>test>*.test;" "{eris}<lispcore>cml>test>*.x") which runs all the internal tests.

`*TEST-COMPILE*`
If this switch is non-nil, `DO-TEST` compiles its forms before testing them. `DO-ALL-TESTS` will print a message in its header if this switch is on.

`*ALL-FILES-REMAINING*`
While `DO-ALL-TESTS` is running, this variable contains a list of all the files remaining to be processed; files are removed from it AFTER they are read and executed. To restart a test run that somehow crashes the test driver, first clean up whatever blew up the run (if necessary, dump `*ALL-FILES-REMAINING*` to a file and get a new sysout), then do
`(DO-ALL-TESTS :RESUME T` [:RESULTS "wherever"]).

Options to do-test-group.

`:before`	allows for a setup form for a group of tests.
`:after`	allows a form to be run after the tests without affecting results.

The normal form of a `DO-TEST-GROUP` using all its features is:

```
(DO-TEST-GROUP
       ("a test group"
       :BEFORE (progn (before-form-1) (before-form-2)...)
       :AFTER (progn (after-form-1) (after-form-2))
       )
       (DO-TEST "first test" ....)
       (DO-TEST "second test" ....)
)
```