

## .dyalogtest:

```
DyalogTest: 1.84
[SuccessValue: ...]
[Setup: ...]
Test: test_1
Test: test_foo
...
[Teardown: ...]
```

## Test function

```
▽ r←mytest sink
A test stuff
▽
    OR
test_f←{
    y←testSubj arg
    [MsgVar]←expc Assert y:
    ...
}
```

Test functions need to return an empty string to indicate success. If you want to use 0 or other values, have a look at the "SuccessValue" modifier or add it to the .dyalogtest suite.

## Test DSL

**[docvar]←x Assert y OR x Check y**

Returns 1 if the assertion that  $x \neq y$  is wrong, 0 otherwise.

If `—halt` modifier is set, halts execution if check fails.

Additional comments on line or immediately before or after. If comments are computed, use `docvar←x Assert y`

**x IsNotElement y**

test  $\sim x \in y$  and halts execution if it isn't.

**x Because y**

concatenates y to global "r" and returns x.

=> "Syntax sugar" to enable statements like:

```
:if 1 Check 2 ⋄ →0 Because '1≠2!' ⋄ :endif
```

**n←[id] ##.RandomVal x [y]**

generates y (default=1) random values identified by „id“ (like [y]?x).

**('Type' 'I|W|E')Log txt**

Adds txt to specified log (Info / Warning / Error)

## ]DTest {.dyalogtest | .apl | .dyalog | path} -modifiers

Modifiers:

- `—halt`: halts execution when Check or Assert fails (so that you can examine the ws)
- `—trace`: trace into setup(s) and tests()
- `—verbose`: show text logged with Log. (test fns should access `##.verbose` if they want to support this for `□←..output!`)
- `—quiet [=0 | 1]`: only shows error messages (1) or all messages (0)
- `—filter=aaa`: select tests to execute (supports \* and ?)
- `—loglvl=n`: controls the log files DTest creates. Value is a sum of the values...
  - 1={base.log} - Errors
  - 2={base}.warn.log - Warnings
  - 4={base}.info.log - Informations
  - 8={base}.session.log - Session log
  - 16={base}.session.log - Session log ONLY for failing tests
  - 32={base}.log.json - machine-readable results ("rc"=20: Success, 21=Failure)
- `—off [=0 | 1]`: do (1) or do not (0) exit APL after running tests (also writes logfiles if required)
- `—order [=0 | 1 | "numvec"]`: order of tests. (0=random, 1=alphabetical, numvec specifies alternate order)
- `—SuccessValue=nnn`: the value that successful tests need to return

## Useful variables for tests

`##.TESTSOURCE`: the path of the current test

`##.DYALOG`: interpreter's home-directory

`##.DyaVersion`: interpreter's version number

## Info about the platform (boolean flags)

`##._isClassic / ##._is32bit / ##._is64bit`

`##._isWin / ##._isLinux / ##._isMacOS`

## Code Coverage

In the .dyalogtest file: `CodeCoverage_Subject: #.ns[,#.ns2]`

use `—coverage` modifier when running ]DTest