Error Handling

Adám Brudzewsky

Error Handling

Adám Brudzewsky

Error Handling – Part 1

Tradfn structure `:Trap errno` (0 means all)

Dfn error guard `errno::result_expression`

Error Number of last error `⎕EN`

Error Message for number `⎕EM`

Error Handling – Part 1

dyalog.tv

Tradfn structure `:Trap errno` (0 means all)

Dfn error guard `errno::result_expression`

Error Number of last error `⎕EN`

Error Message for number `⎕EM`

Overview

Overview

- Trap Event:

Overview

- Trap Error: ⎕TRAP

Overview

- Trap Error: ⎕TRAP
- Extended Diagnostic Message: ⎕DMX

Preventing the build-up of a stack

Preventing the build-up of a stack

```
∇ foo  ⍝ bad function
  1÷0
∇
```

Preventing the build-up of a stack

```
      ∇ foo  ⍝ bad function
        1÷0
      ∇
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
```

Preventing the build-up of a stack

```
      ∇ foo  ⍝ bad function
        1÷0
      ∇
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
      )si
#.foo[1]*
```

Preventing the build-up of a stack

```
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
```

Preventing the build-up of a stack

```
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
      )si
#.foo[1]*
#.foo[1]*
```

Including scope info in an error report

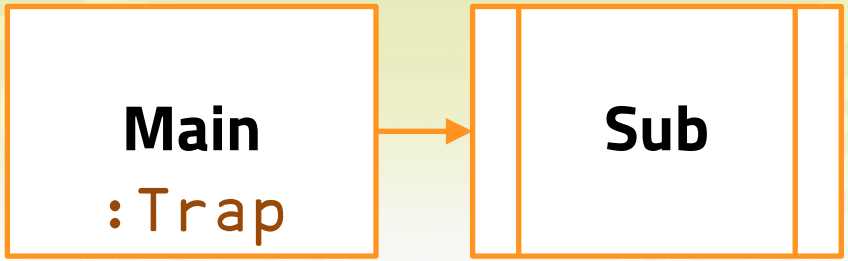
Including scope info in an error report

Including scope info in an error report

Main

```
:Trap
```

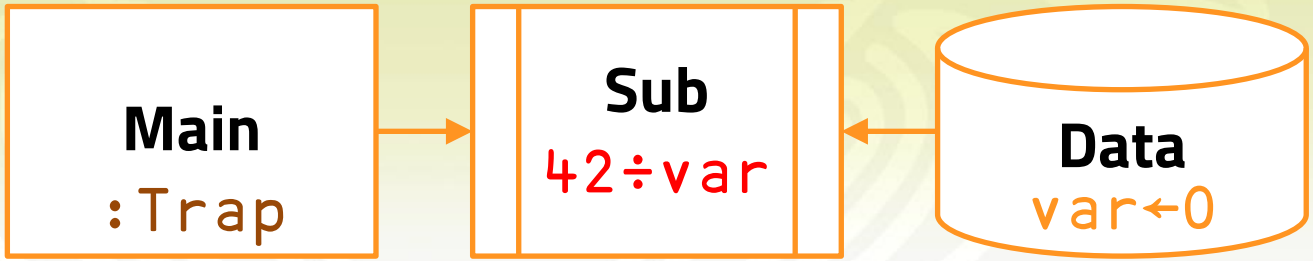
Including scope info in an error report



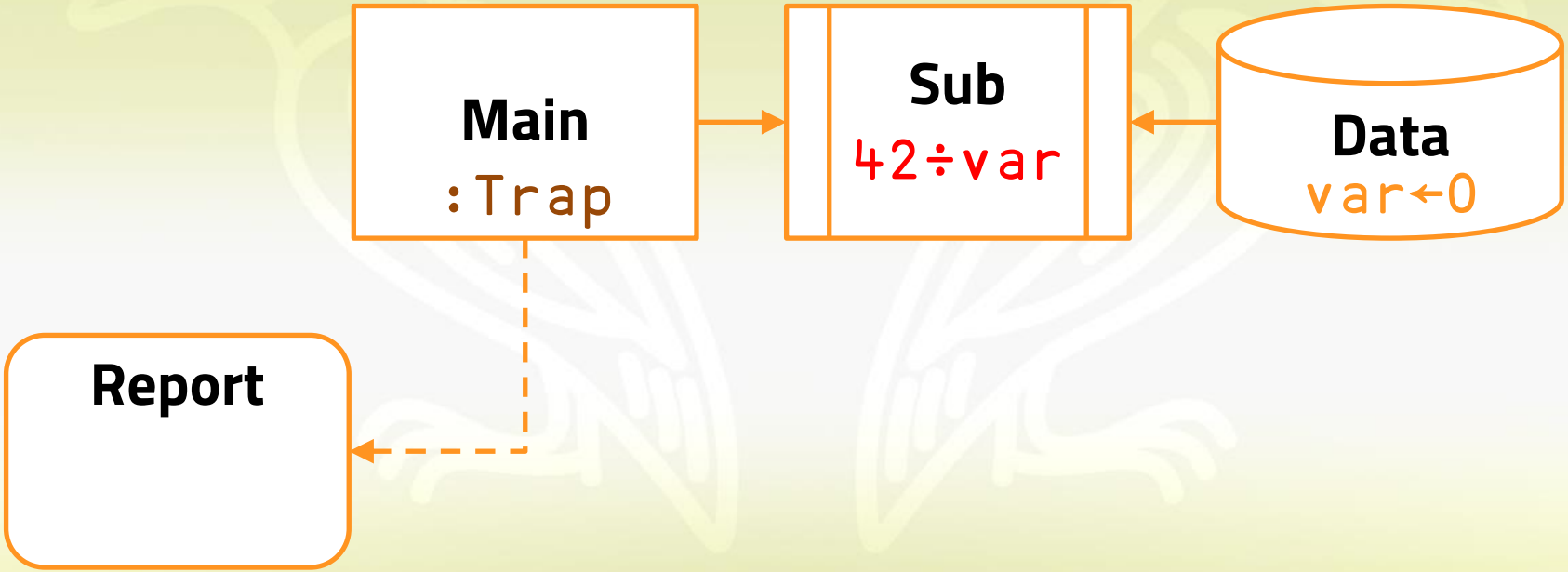
Including scope info in an error report

Including scope info in an error report

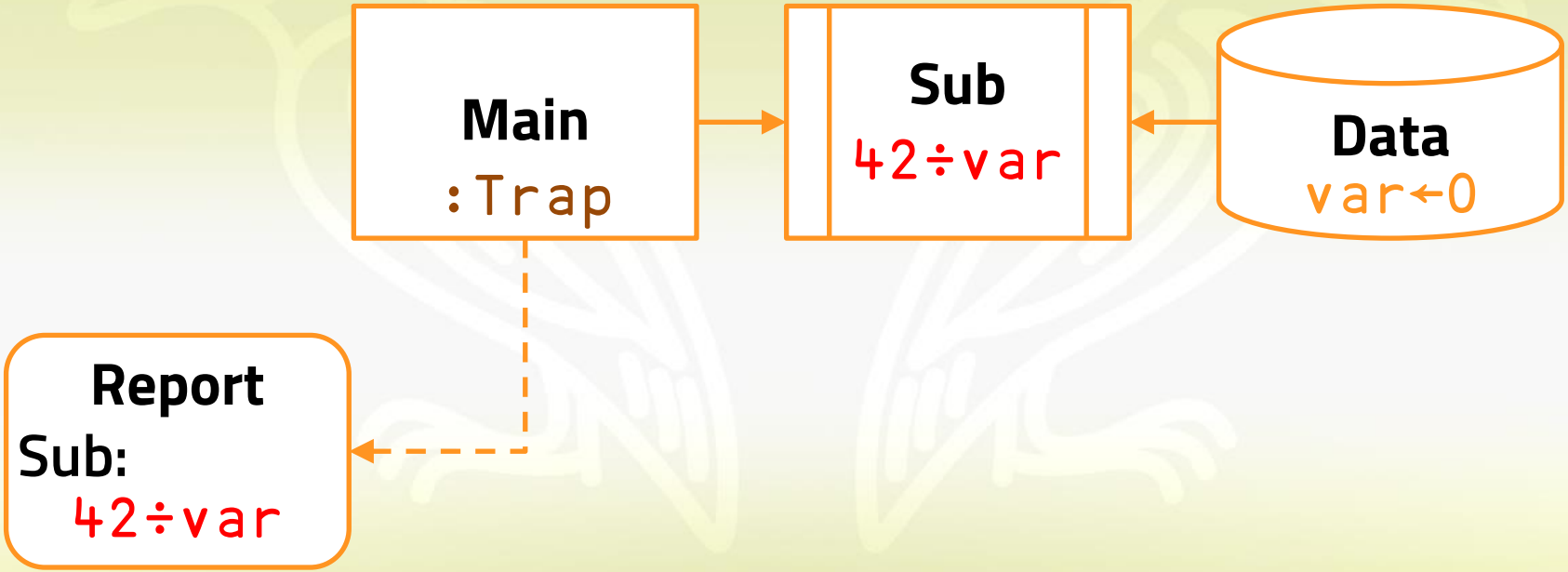
Including scope info in an error report



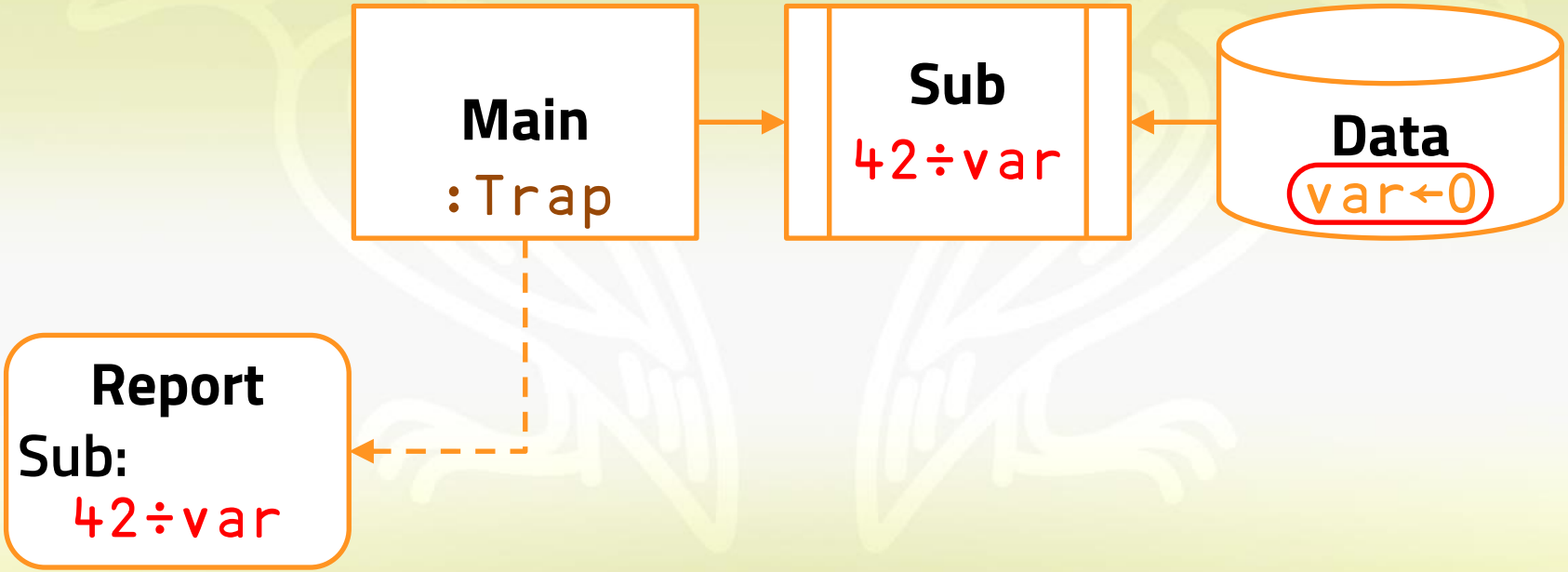
Including scope info in an error report



Including scope info in an error report



Including scope info in an error report



Trap Error

system variable

⎕TRAP

Trap Error

```
⍞TRAP ←
```

handler

Trap Error

```
⎕TRAP ←                    handler
                  handler  handler  handler
```

Trap Error

```
⎕TRAP ←                handler
          handler  handler  handler
                  ⊂handler
```

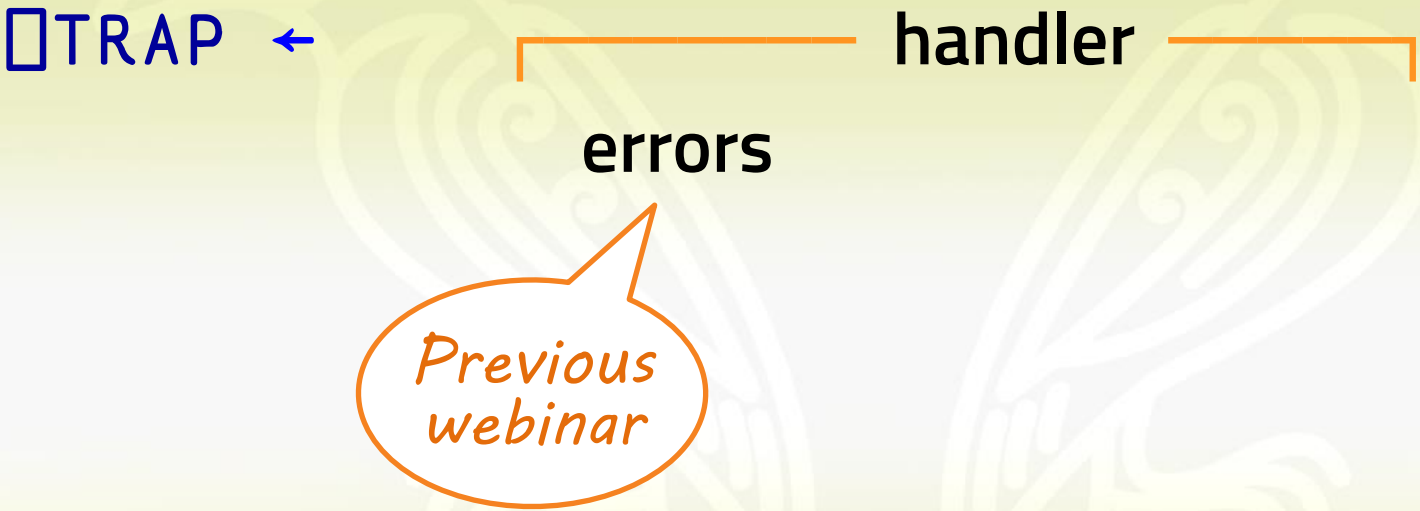
Trap Error

```
⎕TRAP ← handler
```

Trap Error

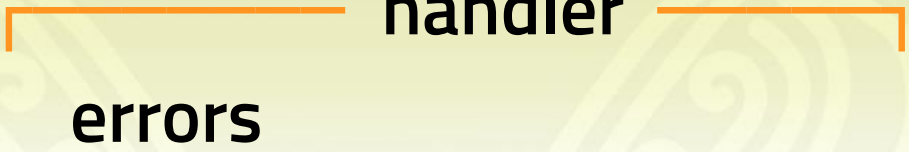
Trap Error

Trap Error



Trap Error

```
⍞TRAP ←
```



Trap Error

```
⍞TRAP ←
```

handler

errors action

Trap Error

```
⍞TRAP ←
```

handler

errors action code

Trap Error

```
⍞TRAP ←
```

handler

errors action code

errors action

Trap Error

```
⍞TRAP ←
```

handler

Execute there: **errors** `'E'` **code**

Trap Error

```
⍞TRAP ←
```

handler

Execute there:	**errors**	`'E'`	**code**
Cut stack first:	**errors**	`'C'`	**code**

Trap Error

```
⎕TRAP ← ┌──────────── handler ────────────┐
```

Execute there: **errors** `'E'` **code**

Cut stack first: **errors** `'C'` **code**

Next handler: **errors** `'N'`

Trap Error

```
⎕TRAP ←
```

handler

Execute there:	**errors**	`'E'`	**code**
Cut stack first:	**errors**	`'C'`	**code**
Next handler:	**errors**	`'N'`	
Stop looking:	**errors**	`'S'`	

Trap Error

```
⎕TRAP ←
```

handler

Execute there: **errors** `'E'` **code**

Trap Error: Execute there

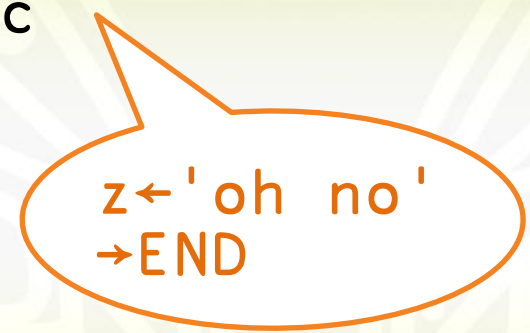
```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```



Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP                    Foo 1
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP                    Foo 1
  c←'z←''oh no''◇→END'               4 3 5 0
  ⎕TRAP←11 'E' c                     d o n e

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

```
      Foo 1
4 3 5 0
d o n e
      Foo 0
```

Trap Error: Execute there

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

```
      Foo 1
4 3 5 0
d o n e
      Foo 0
oh no
done
```

Trap Error: Execute there

Trap Error: Execute there

```
⎕TRAP←5 'E' '''Silly!'''
```

Trap Error: Execute there

```
⎕TRAP←5 'E' '''Silly!'''
```

Trap Error: Execute there

```
      ⎕TRAP←5 'E' '''Silly!'''
      'a'+'b'
DOMAIN ERROR
      'a'+'b'
         ^
```

Trap Error: Execute there

```
      ⎕TRAP←5 'E' '''Silly!'''
      'a'+'b'
DOMAIN ERROR
      'a'+'b'
         ^
      1 2+3 4 5
Silly!
```

Trap Error: ⎕TRAP value normalisation

```
⎕TRAP←5 'E' '''Silly!'''
```

Trap Error: ⎕TRAP value normalisation

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⍴⎕TRAP
1
```

Trap Error: ⎕TRAP value normalisation

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⍴⎕TRAP
1

      ⎕SE.Dyalog.Utils.repObj ⎕TRAP
```

Trap Error: ⎕TRAP value normalisation

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⍴⎕TRAP
1

      ⎕SE.Dyalog.Utils.repObj ⎕TRAP
```

Trap Error: ⎕TRAP value normalisation

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⍴⎕TRAP
1
      ⎕SE.Dyalog.Utils.repObj ⎕TRAP
,⊂(,5) 'E' '''Silly!'''
```

18·1:
]Repr

Trap Error: ⎕TRAP value normalisation

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⍴⎕TRAP
1
      ⎕SE.Dyalog.Utils.repObj ⎕TRAP
,⊂(,5) 'E' '''Silly!'''
```

Trap Error: ⎕TRAP value normalisation

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⍴⎕TRAP
1
      ⎕SE.Dyalog.Utils.repObj ⎕TRAP
,⊂(,5) 'E' '''Silly!'''
```

Trap Error: Resetting ⎕TRAP

```
⎕TRAP←5 'E' '''Silly!'''
```

Trap Error: Resetting ⎕TRAP

```
⎕TRAP←5 'E' '''Silly!'''
⎕TRAP←0
```

Trap Error: Resetting ⎕TRAP

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⎕TRAP←0
Silly!
```

Trap Error: Resetting ⎕TRAP

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⎕TRAP←0
Silly!
```

Trap Error: Resetting ⎕TRAP

```
⎕TRAP←5 'E' '''Silly!'''
```

Trap Error: Resetting ⎕TRAP

```
⎕TRAP←5 'E' '''Silly!'''
⎕TRAP←0⍴⎕TRAP
```

Trap Error: Resetting ⎕TRAP

```
⎕TRAP←5 'E' '''Silly!'''
⎕TRAP←0⍴⎕TRAP
⎕SE.Dyalog.Utils.repObj ⎕TRAP
```

Trap Error: Resetting ⎕TRAP

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⎕TRAP←0⍴⎕TRAP
      ⎕SE.Dyalog.Utils.repObj ⎕TRAP
0⍴⊂(,0) ' ' (8⍴' ')
```

Trap Error: Resetting ⎕TRAP

```
      ⎕TRAP←5 'E' '''Silly!'''
      ⎕TRAP←0⍴⎕TRAP
      ⎕SE.Dyalog.Utils.repObj ⎕TRAP
0⍴⊂(,0) ' ' (8⍴' ')
```

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP                    Foo 1 2
  c←'z←''oh no''⋄→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←11 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

```
      Foo 1 2
LENGTH ERROR: Mismatche
Foo[2] z←4 3 5÷y
           ^
```

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←(11 5) 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←(11 5) 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

11 5::
DOMAIN
& LENGTH
ERROR

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP                    Foo 1 2
  c←'z←''oh no''◇→END'
  ⎕TRAP←(11 5) 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

11 5::
DOMAIN
& LENGTH
ERROR

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←(11 5) 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

```
      Foo 1 2
oh no
done
```

11 5::
DOMAIN
& LENGTH
ERROR

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←(11 5) 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

```
      Foo 1 2
oh no
done

      Foo 0
```

11 5::
DOMAIN
& LENGTH
ERROR

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''◇→END'
  ⎕TRAP←(11 5) 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

```
      Foo 1 2
oh no
done

      Foo 0
oh no
done
```

11 5::
DOMAIN
& LENGTH
ERROR

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←0 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Multiple handlers in one

```
∇ z←Foo y;c;⎕TRAP
  c←'z←''oh no''⋄→END'
  ⎕TRAP←0 'E' c

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

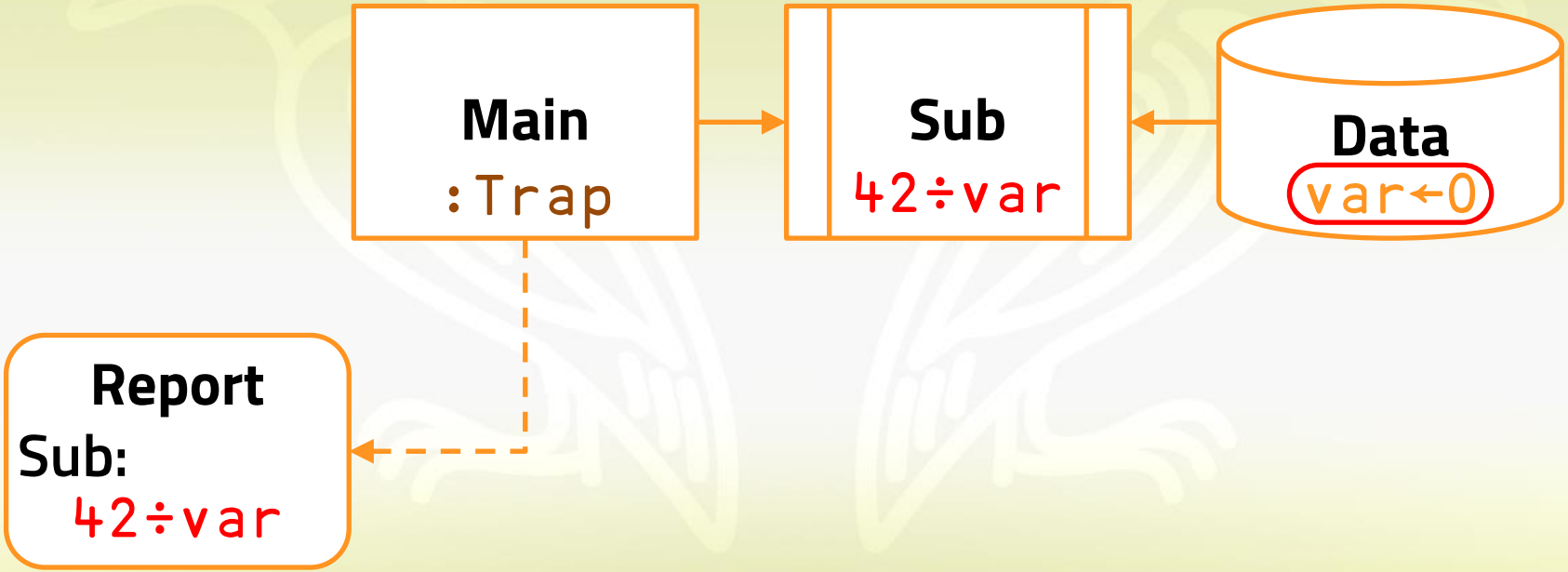
Trap Error: Multiple handlers in one

```
∇ z←Foo y;b;c;⎕TRAP
  b←'z←''bad''⋄→END' ⋄ c←'z←''count!''⋄→END'
  ⎕TRAP←(11 'E' b) (5 'E' c)

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

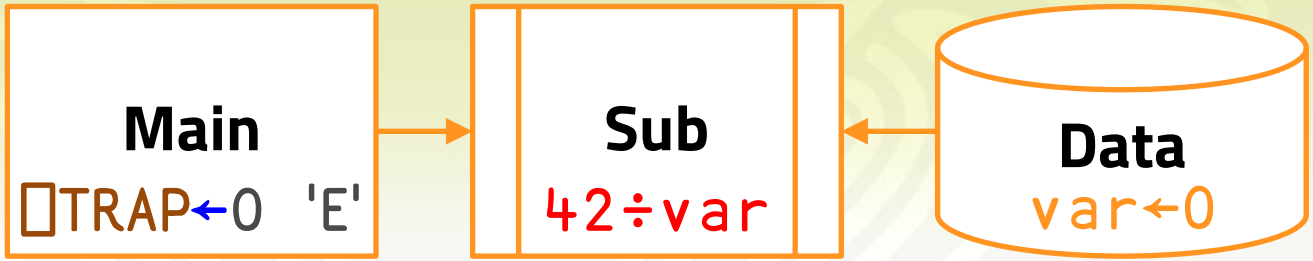
Trap Error: Multiple handlers in one

```
∇ z←Foo y;b;c;d;⎕TRAP
  b←'z←''bad''⋄→END' ⋄ c←'z←''count!''⋄→END'
  d←'z←''oh no''⋄→END'
  ⎕TRAP←(11 'E' b) (5 'E' c) (0 'E' d)

  z←4 3 5÷y
END:
  z←↑z'done'
∇
```

Trap Error: Use-case

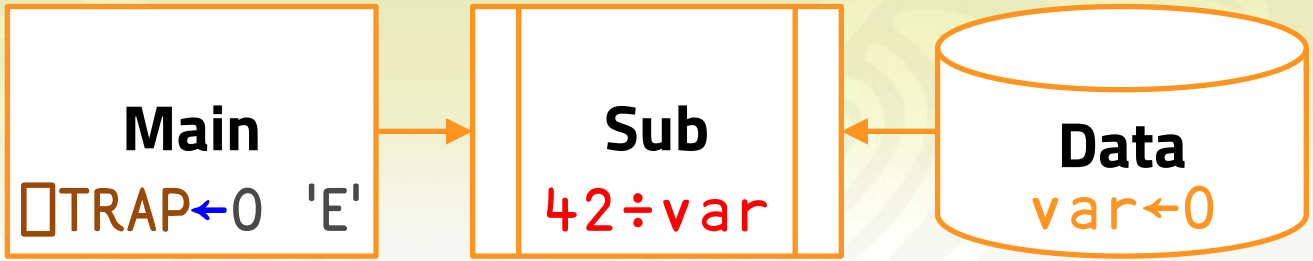


Trap Error: Use-case



Report
Sub:
42÷var

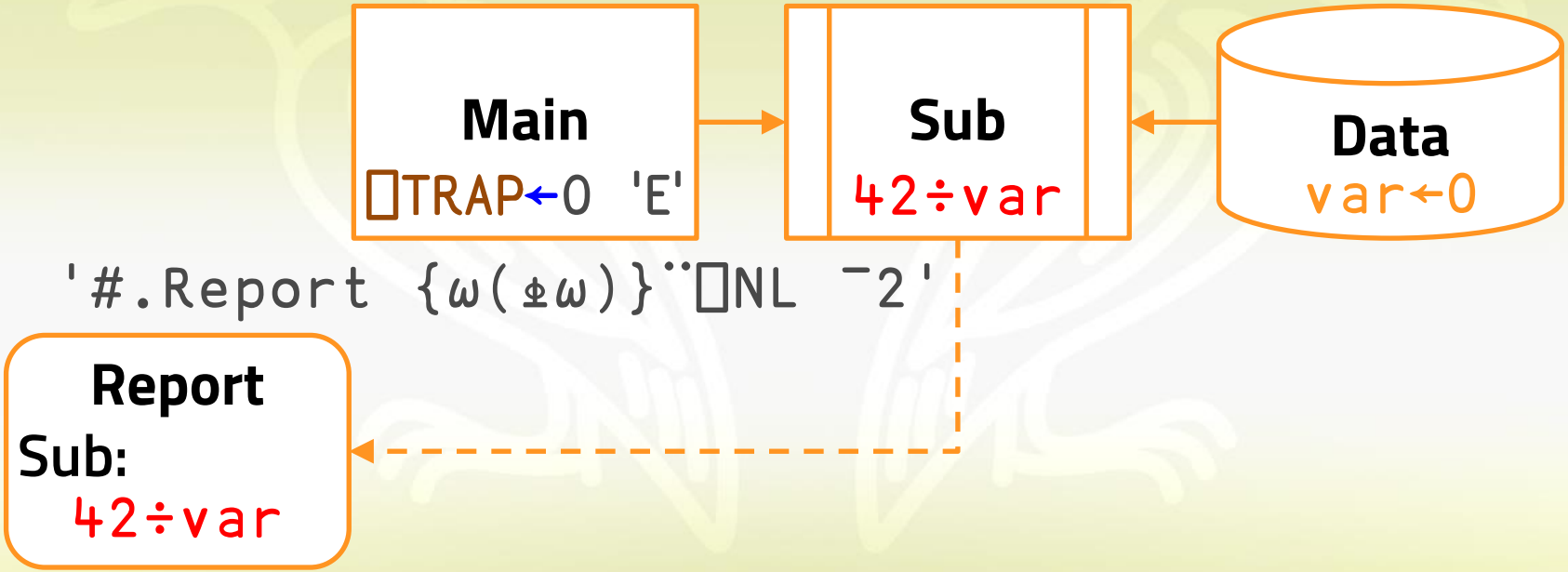
Trap Error: Use-case



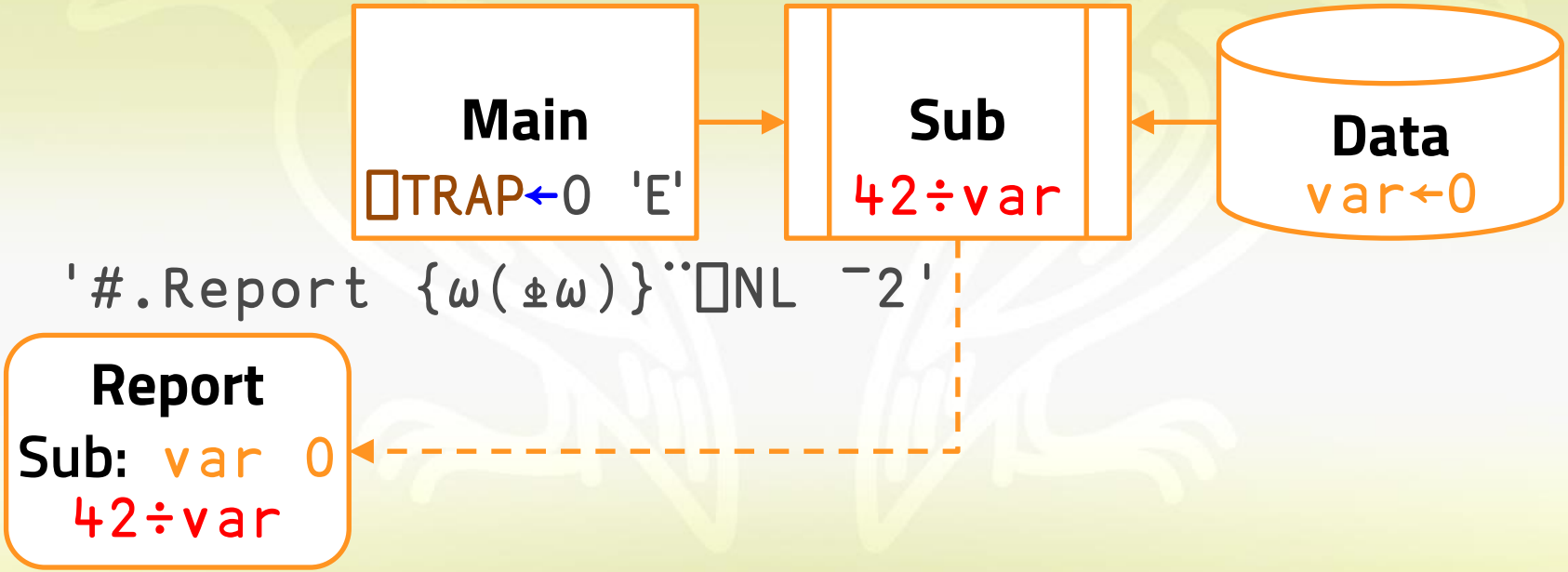

```
'#.Report {ω(⍎ω)}¨⎕NL ¯2'
```

Report
Sub:
42÷var

Trap Error: Use-case



Trap Error: Use-case



Trap Error: Use-case

```
⎕TRAP←0 'E' '#.Report {⍵(≢⍵)}¨⎕NL ¯2'
```

Trap Error: Use-case

```
⎕TRAP←0 'E' '#.Report {⍵(≢⍵)}¨⎕NL ¯2'

⎕TRAP←0 'E' 'err #.Report {⍵(≢⍵)}¨⎕NL ¯2'
```

Trap Error: Use-case

```
⎕TRAP←0 'E' '#.Report {⍵(⍎⍵)}¨⎕NL ¯2'

⎕TRAP←0 'E' '⎕EN #.Report {⍵(⍎⍵)}¨⎕NL ¯2'
```

Trap Error: Use-case

```
⎕TRAP←0 'E' '#.Report {⍵(↓⍵)}¨⎕NL ¯2'

⎕TRAP←0 'E' '⎕EN #.Report {⍵(↓⍵)}¨⎕NL ¯2'
```

Extended Diagnostic Message

niladic namespace-returning system function

⎕DMX

Extended Diagnostic Message

Extended Diagnostic Message

```
      )clear
clear ws
```

Extended Diagnostic Message

```
      )clear
clear ws
      p⎕DMX
```

Extended Diagnostic Message

```
      )clear
clear ws
      ⍴⎕DMX

      ≡⎕DMX
0
```

Extended Diagnostic Message

```
      )clear
clear ws
      ⍴⎕DMX

      ≡⎕DMX
0
      ⎕DMX
```

Extended Diagnostic Message

```
      )clear
clear ws
      ⍴⎕DMX

      ≡⎕DMX
0
      ⎕DMX
      {⎕NC'⍵'}⎕DMX
9
```

Extended Diagnostic Message

```
      )clear
clear ws
      ⍴⎕DMX
0
      ⎕DMX
      {⎕NC'⍵'}⎕DMX
9
```

Extended Diagnostic Message

```
      1÷0
DOMAIN ERROR: Divide by zero
      1÷0
       ^
```

Extended Diagnostic Message

```
      1÷0
DOMAIN ERROR: Divide by zero
      1÷0
        ^
      ⎕DMX
 EM          DOMAIN ERROR
 Message     Divide by zero
```

Extended Diagnostic Message

```
      ⎕DMX
EM        DOMAIN ERROR
Message   Divide by zero
```

Extended Diagnostic Message

```
      ⎕DMX.(2 2⍴'EM' EM 'Message' Message)
EM          DOMAIN ERROR
Message     Divide by zero

      ⎕DMX
EM          DOMAIN ERROR
Message     Divide by zero
```

Extended Diagnostic Message

```
⎕JSON⍠'Compact'0⊢⎕DMX
```

Extended Diagnostic Message

```
      ⎕JSON⍠'Compact'0⊢⎕DMX
{
  "Category": "General",
  "DM": [
    "DOMAIN ERROR",
    "      1÷0",
    "        ^"
  ],
  "EM": "DOMAIN ERROR",
  "EN": 11,
  "ENX": 1,
  "HelpURL": "https://help.dyalog.com/dmx/18.0/General/1",
```

Extended Diagnostic Message: use case

Extended Diagnostic Message: use case

```
      ⎕DMX.(EM,' caused by ',{⍵/⍨∨\⍵≠' '}2⊃DM)
DOMAIN ERROR caused by 1÷0
```

Extended Diagnostic Message: use case

```
      ⎕DMX.(EM,' caused by ',{⍵/⍨~⍵∨.\⍵≠' '}2⊃DM)
DOMAIN ERROR caused by 1÷0
      ⎕TRAP←0 'E' ⋄
⎕DMX.(EM,' caused by ',{⍵/⍨~⍵∨.\⍵≠' '}2⊃DM)
```

Extended Diagnostic Message: use case

```
      ⎕DMX.(EM,' caused by ',{⍵/⍨~⍵∨.\⍵≠' '}2⊃DM)
DOMAIN ERROR caused by 1÷0
      ⎕TRAP←0 'E' ⋄
⎕DMX.(EM,' caused by ',{⍵/⍨~⍵∨.\⍵≠' '}2⊃DM)

      ⍬+1 2
LENGTH ERROR caused by ⍬+1 2
```

Extended Diagnostic Message: use case

```
⎕TRAP⍴⍨←0
```

Extended Diagnostic Message: use case

```
      ⎕TRAP←0
      1÷0
DOMAIN ERROR: Divide by zero
      1÷0
        ^
      2×'A'
DOMAIN ERROR
      2×'A'
        ^
```

Extended Diagnostic Message: use case

```
⎕TRAP⍴⍨←0
```

Extended Diagnostic Message: use case

```
⎕TRAP←⍬
⎕NGET 'nope'
```

Extended Diagnostic Message: use case

```
      ⎕TRAP⍴⍨←0
      ⎕NGET'nope'
FILE NAME ERROR: nope: Unable to open file
("The system cannot find the file specified.")
      ⎕NGET'nope'
      ^
```

Extended Diagnostic Message: use case

aplcart.info

```
      ⎕TRAP←0
      ⎕NGET'nope'
FILE NAME ERROR: nope: Unable to open file
("The system cannot find the file specified.")
      ⎕NGET'nope'
      ^
```

Extended Diagnostic Message: use case

```
      ⎕TRAP⍴⍨←0
      ⎕NGET'nope'
FILE NAME ERROR: nope: Unable to open file
("The system cannot find the file specified.")
      ⎕NGET'nope'
      ^
```

aplcart.info

18·1:
]APLcart

Tell me about:

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

⍬	Empty Numeric Vector
⊢Y	Same: Y
X dop Y⊢Z	Separate dyadic operator's right operand from its right argument (same as (X dop Y)Z)
X⊢Y	Right: Y
X⊢Y	Church Boolean false (X if false, else Y)
⊣Y	Same: Y
X⊣Y	Left: X
X⊣Y	Church Boolean true (X if true, else Y)
+Y	Conjugate ('Identity' if Y not complex)
+N	Mirror complex N across x-axis
M+N	Adding N to M

⍁ ? Tell me about: dmx ⊗ #

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

▶ ⎕DMX

? Namespace containing details of most recent error in current thread

```
{r}←{msg} ⎕SIGNAL errno/{nvpairs}…
```

? Signal an error; nvpairs set ⎕DMX properties

```
{0::⎕DMX⍪⎕DMX.(⎕EX⎕NL)2 ⋄ .}Y
```

New root namespace (Y is ignored)

▶
```
⎕SIGNAL⊂⎕DMX.(('EN'EN)
('Message'(OSError{⍵,2⍕(×≢⍵)/''")
("',⍵←⊃⍬⍴2⍕⍺}Message)))
```

Re-signal last caught error to caller (works with any ⎕IO and ⎕ML)

▶
```
⎕DMX.(OSError{⍵,2⍕(×≢⊃⍬⍴2⍕⍺)/''")
("',⊃⍬⍴2⍕⍺}Message{⍵,⍺,⍨': '/⍨×≢⍺}⊃⍬⍴⎕DM,⊂'')
```

Construct first line of printed error message (works with any ⎕IO and ⎕ML)

≤⎡ ? Tell me about: dmx ⊗ #

X,Y,Z:any M,N:num I,J:int A,B:Bool C,D:char f,g,h:fn ax:axis s:scal v:vec m:mat

⎕DMX	Namespace containing details of most recent error in current thread
`{r}←{msg} ⎕SIGNAL errno/{nvpairs}…`	Signal an error; nvpairs set ⎕DMX properties
`{0::⎕DMX⊣⎕DMX.(⎕EX⎕NL)2 ⋄ .}Y`	New root namespace (Y is ignored)
`⎕SIGNAL⊂⎕DMX.(('EN'EN)('Message'(OSError{⍵,2⌽(×≢⍺)/'"')('"',⍺←⊃⍬⍴2⌽⍺}Message)))`	Re-signal last caught error to caller (works with any ⎕IO and ⎕ML)
`⎕DMX.(OSError{⍵,2⌽(×≢⊃⍬⍴2⌽⍺)/'"')('"',⊃⍬⍴2⌽⍺}Message{⍵,⍺,⍨': '/⍨×≢⍺}⊃⍬⍴DM,⊂'')`	Construct first line of printed error message (works with any ⎕IO and ⎕ML)

Showing 5 of 2557 Quiz

Extended Diagnostic Message: use case

Extended Diagnostic Message: use case

```
∇ err  ⍝ error handler


∇
```

Extended Diagnostic Message: use case

```
∇ err  ⍝ error handler
  ⎕DMX.(OSError{⍵,2⌽(×≢⊃⍬⍴2⌽⍺)/'"') ("',⊃⍬⍴
∇
```

Extended Diagnostic Message: use case

```
∇ err  ⍝ error handler
  ⎕DMX.(OSError{⍵,2⌽(×≢⊃⍬⍴2⌽⍺)/'"') ("',⊃⍬⍴
  ↑1↓⎕DMX.DM
∇
```

Extended Diagnostic Message: use case

```
∇ err  ⍝ error handler
  ⎕DMX.(OSError{ω,2⍕(×≢⊃⍵⍴2⍕⍺)/'"') ('"',⊃⍵⍴
  ↑1↓⎕DMX.DM
∇
⎕TRAP←0 'E' '#.err'
```

Extended Diagnostic Message: use case

```
      ∇ err  ⍝ error handler
        ⎕DMX.(OSError{⍵,2⍕(×≢⊃⍬⍴2⍕⍺)/'"') ("',⊃⍬⍴
        ↑1↓⎕DMX.DM
      ∇
      ⎕TRAP←0 'E' '#.err'
      1÷0
DOMAIN ERROR: Divide by zero
      1÷0
       ^
```

Extended Diagnostic Message: use case

```
      ∇ err  ⍝ error handler
        ⎕DMX.(OSError{ω,2⍕(×≢⊃⍵⍴2⍕⍺)/'"') ('",⊃⍵⍴
        ↑1↓⎕DMX.DM
      ∇
      ⎕TRAP←0 'E' '#.err'
      1÷0
DOMAIN ERROR: Divide by zero
      1÷0
       ^
```

???

Extended Diagnostic Message: use case

Extended Diagnostic Message: use case

```
∇ foo  ⍝ bad function
  1÷0
∇
```

Extended Diagnostic Message: use case

```
      ∇ foo  ⍝ bad function
        1÷0
      ∇
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
```

Extended Diagnostic Message: use case

```
      ∇ foo  ⍝ bad function
        1÷0
      ∇
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
      )si
#.foo[1]*
```

Extended Diagnostic Message: use case

```
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
```

Extended Diagnostic Message: use case

```
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
      )si
#.foo[1]*
#.foo[1]*
```

Trap Error

```
⍞TRAP ←
```

handler

Execute there: **errors** `'E'` **code**

Cut stack first: **errors** `'C'` **code**

Next handler: **errors** `'N'`

Stop looking: **errors** `'S'`

Trap Error

```
⍞TRAP ←
```

handler

Cut stack first: **errors** `'C'` **code**

Extended Diagnostic Message: use case

```
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
      )si
#.foo[1]*
#.foo[1]*
```

Extended Diagnostic Message: use case

```
⎕TRAP←0 'E' '#.err'
```

Extended Diagnostic Message: use case

```
⎕TRAP←0 'C' '#.err'
```

Extended Diagnostic Message: use case

```
⎕TRAP←0 'C' '#.err'
)sic
```

Extended Diagnostic Message: use case

```
      ⎕TRAP←0 'C' '#.err'
      )sic
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
```

Extended Diagnostic Message: use case

```
      ⎕TRAP←0 'C' '#.err'
      )sic
      foo
DOMAIN ERROR: Divide by zero
foo[1] 1÷0
         ^
      )si
```

Error Handling – Part 2

Execute there on error	`⎕TRAP←errors 'E' code`
Cut back and execute	`⎕TRAP←errors 'C' code`
Ext'd Diagnostic Message	`⎕DMX`

Error Handling – Part 2

Construct error message
aplcart.info?q=dmx 1

Execute there on error `⎕TRAP←errors 'E' code`

Cut back and execute `⎕TRAP←errors 'C' code`

Ext'd Diagnostic Message `⎕DMX`

Error Handling – Part 2

Construct error message
aplcart.info?q=dmx 1

Execute there on error `⎕TRAP←errors 'E' code`

Cut back and execute `⎕TRAP←errors 'C' code`

Ext'd Di…ge `⎕DMX`

Press F1 *for full docs!*

Upcoming webinars

Apr	22	BAA	open session
Apr	29	- - -	
May	6	BAA	open session
May	13	Dyalog	Drawing the Mandelbrot Set
May	20	BAA	open session
May	27	- - -	
Jun	5	BAA	open session
Jun	10	Dyalog	Error Handling – Part 3

Upcoming webinars

Apr	22	BAA	open session
Apr	29	- - -	
May	6	BAA	open session
May	13	Dyalog	Drawing the Mandelbrot Set
May	20	BAA	open session
May	27	- - -	
Jun	5	BAA	open session
Jun	10	Dyalog	Error Handling – Part 3

britishaplassociation.org