# DrA User Guide (Preliminary)

First Draft, October 12th, 2012

## Introduction

An error logging mechanism is an important part of any modern APL application – especially a server application, which is expected to trap all errors and keep running unless something “fatal” occurs. “DrA” is an error logging tool for Dyalog APL, named after “Dr.Watson”, a debugging tool for the Watcom C compiler that was used to compile Dyalog APL back when DrA first saw the light of day.

This document has been hastily written as a companion to the “Starting a New APL Application” workshop at Dyalog’12. A more complete user guide will hopefully be written soon!

## Functionality

When correctly configured, DrA creates an APL component file when a trapped error is encountered. The file contains information about the program stack when the error occurred, the definition of the failing function, and the values of any variables which were referenced on the line of code which failed. It is also possible for the developer to request logging of a list of fixed variable names,

When errors occur, the log file can automatically be sent as an attachment via SMTP e-mail to a system administrator, and execution directed to a label called RESUME, if one exists.

The following simple application illustrates how to use DrA:

∇ Foo;⎕TRAP;A;B;C

[1] DrA.SetDefaults

[2] ⎕TRAP←DrA.TrapGUI

[3] A←1 ⋄ B←0

[4] C←A÷B

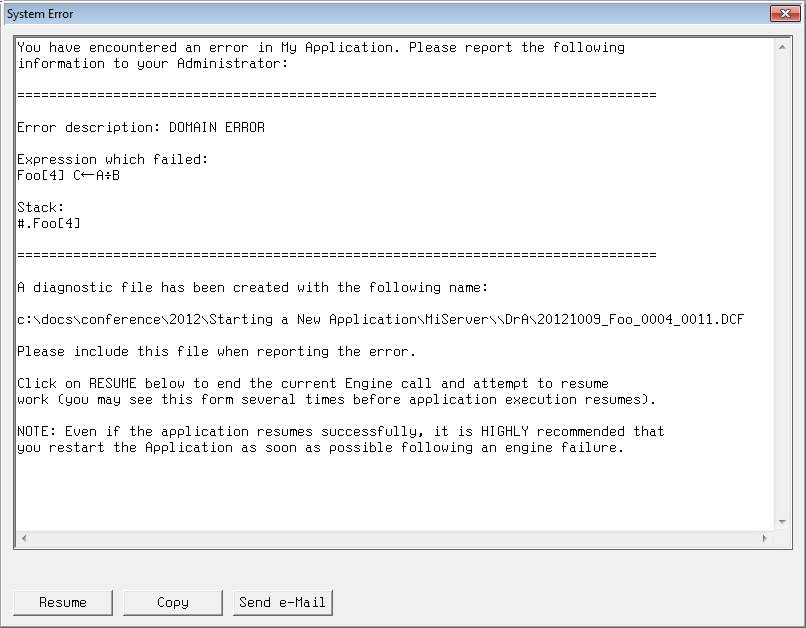
[5] →0

[6]

[7] RESUME: ⎕←'Application Resumed'

∇

When this function runs, a DOMAIN ERROR will occur on line [4] and the user interface shown at the top of the next page will be displayed. The user will be presented with the option of resuming execution, copying the text to the clipboard, or sending an e-mail containing diagnostic information to the system administrator.



The function DrA.Diagnose allows inspection of the logged errors:

NS←DrA.Diagnose ''

1. c:\myapp\DrA\20121009\_Foo\_0004\_0005

2. c:\myapp\DrA\20121009\_Foo\_0004\_0011

⎕:

2

⎕DM DOMAIN ERROR

Foo[4] C←A÷B

∧

⎕THIS #

⎕XSI #.Foo[4]

⎕TS 2012 10 9 23 32 25 25

⎕WA 521314276

⎕IO 1

⎕EN 11

TimeStamps 2012-10-09 23:20:28

2012-10-09 23:32:25

(2 occurrences)

⎕WSID c:\myapp\myws

APLVersion Windows 13.1.14610.0 W Development

Variables Name Shape Depth Bytes Data

A 0 16 1

B 0 16 0

Function ∇ Foo;⎕TRAP;A;B;C

[1] DrA.SetDefaults

[2] ⎕TRAP←DrA.TrapGUI

[3] A←1 ⋄ B←0

[4] C←A÷B

[5] →0

[6]

[7] RESUME:⎕←'Application Resumed'

∇

Function Fixed: Foo

Variables defined: A B

The result of Diagnose is a namespace containing the relevant information, for closer inspection:

NS.⎕nl 2 3

A

B

DrAInfo

Foo

NS.(A B)

1 0

## Configuring DrA

The behavior of DrA is controlled by a number of variables contained within the DrA namespace. The function DrA.SetDefaults should be run to set these variables to suitable defaults:

∇ r←SetDefaults

[1] ⍝ Set default values for DrA globals

[2]

[3] ⍝ For interactive session: ⎕TRAP←DrA.TrapGUI

[4] ⍝ For unattended operation: ⎕TRAP←DrA.TrapServer ⋄ DrA.NoUser←1

[5] ⍝ and have a branch destination called RESUME

[6] TrapGUI←((901 902)'E' '⎕SIGNAL 10+⎕EN')  
 (0 'E' '⍎#.DrA.Start ⍎⎕FX #.DrA.(⎕NR ''\_DrA\_GetErrorInfo'')')

[7] TrapServer←(⊂912 'C' '→RESUME'),TrapGUI

[8]

[9] GUI\_Title←AppName←'My Application'

[10] Path←{(1-⌊/(⌽⍵)⍳'/\')↓⍵}⎕WSID ⍝ Application root  
 ('DrA\' will be added)

[11] Mode←1 ⍝ 1=Runtime, 2=Developer Mode, see Start

[12] NoUser←0 ⍝ 1: Running unattended: Log Error, "Resume" (signals 912)?

[13] ⍝ 2: Means attempt to send e-mail

[14] LogVars←0⍴⊂'' ⍝ List of variables to be included in all logs

[15]

[16] SMTP\_Gateway←''

[17] MailMethod←'NONE'

[18] MailRecipient←''

[19]

[20] LogFn←'' ⍝ Set to name of external logging function if relevant

[21]

[22] r←'DrA defaults applied'

∇

We will discuss the two “Trap” variables in detail in the next section. The rest of the “switches” have the following effect:

|  |  |
| --- | --- |
| Variable | Effect / Meaning |
| GUI\_Title | A caption which will appears in the dialog boxes that DrA displays. |
| Path | When error log files are created, they will be put in a folder called DrA below this path. |
| Mode | Controls whether a user-interface will be displayed.  Mode=0: No user interface  Mode=1: User mode. A simple form is displayed which allows the user to decide whether to resume or abandon the application  Mode=2: Developer mode. An advanced user interface allows the developer to pop the stack one level at a time, and exit from DrA to debug the problem |
| NoUser | NoUser=0: means use the Mode setting to decide behavior  NoUser=1: Always attempt to resume execution  NoUser=2: Send an e-mail, then resume |
| LogVars | DrA analyzes the failing line of code to determine which variable values |
| MailMethod | ? |
| MailRecipient | ? |
| SMTP\_Gateway | ? |
| LogFn | ? |

Upon inspection of the code, the last four switches seem to be disabled in this version of DrA. If you have need of this functionality, contact the APL Tools group, a new version of DrA will be available sometime after the conference.

## Trap Statements

The first thing to do if you want to use an error trapping system is obviously to set ⎕TRAP! DrA comes with two suggestions for trap statements, depending on your application – DrA.TrapGUI and DrA.TrapServer. TrapGUI is a rather convoluted trap statement; it extracts the definition for a function to collect environmental information from the guts of DrA and fixes it in the current environment, and passes the result of this function to DrA.Start. When DrA.Start is done, the result of the function is executed; the expression which is returned will either signal the error one level up the stack, or branch to RESUME.

TrapServer adds a trap for 912, if this is combined with setting DrA.NoUser←1, then a 912 will always be signaled, the intention is that the application will have a label RESUME at the level where ⎕TRAP is localized.