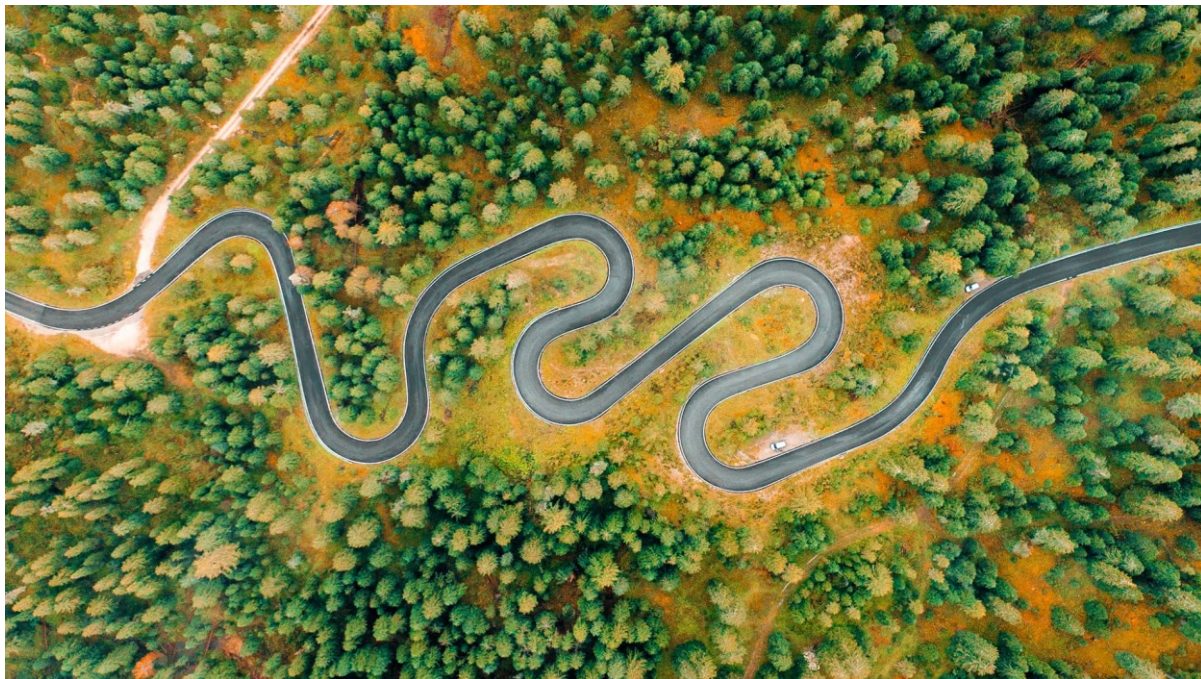




# The Dyalog Road Map

*Morten Kromberg*  
*CTO*





# The Dyalog Road Map

*Morten Kromberg*  
*CTO*



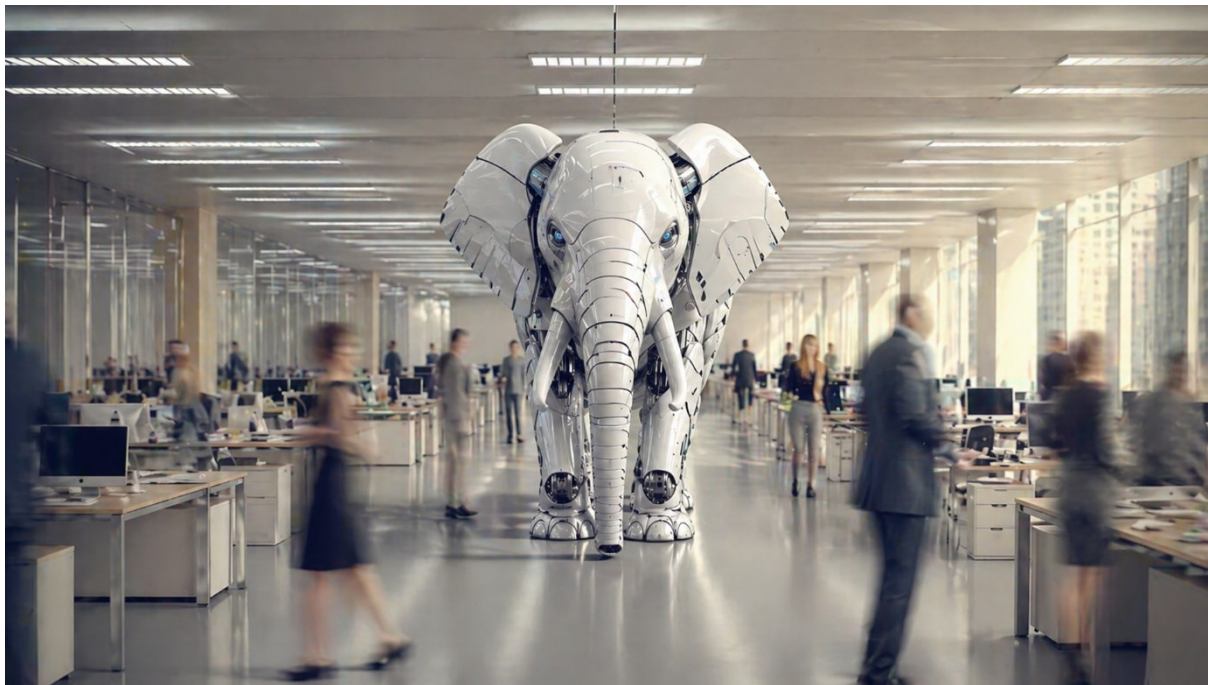


# Large Language Models / AI



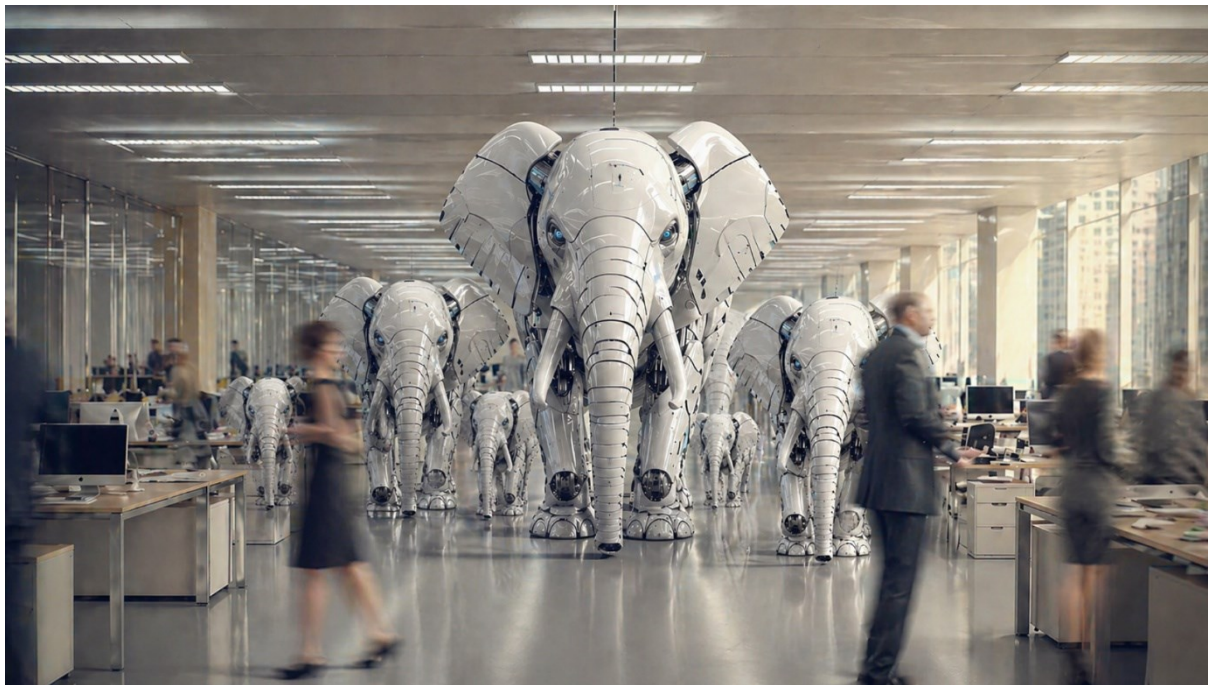
No presentation can be complete without mention of Artificial Intelligence

# Large Language Models / AI



Nobody can afford to ignore LLMs.

# Large Language Models / AI



Nobody can afford to ignore LLMs.



Mike Mingard

# Model improvements

Smarter, not bigger

- ◆ **Significant** improvement in coding performance
- ◆ Console agents are capable of hours' of unsupervised work
- ◆ In your terminal, in the cloud, in your GitHub, on your phone
- ◆ In skilled hands, capable of producing production-grade code in sweet spot languages

# APL Improvements

- Last year, models could kinda explain moderately complex APL
- ...but not reliably write it
- A year later, they're assisting in writing more complex APL applications
- ...in the hands of a capable practitioner!
- ...and with one or two tooling injections





# "Claude is the new Quad"

(Stephen Taylor)

- Turn your application into a service with good documentation
  - (Holden will show you one way to do that later)
- Let Claude write the GUI
  - Claude is really good at JavaScript
  - (Brian will show an example of this later)
- But guess what...



Holden Hoover



Brian Becker



# Claude can do WC too

A — Build GUI

```
'RTForm'WC'Form' 'DataSet Browser'('Coord' 'Pixel')('Size' 640 900)

'RTForm.fnt'WC'Font' 'Consoles' 19
'RTForm.fntSm'WC'Font' 'Consoles' 17
'RTForm.fntGrid'WC'Font' 'Consoles' 18

'RTForm.lblStatus'WC'Label' 'Select a dataset, or import from Excel'('Posn' 10 10)('Size' 20 880)('FontObj' 'RTForm.fnt')

'RTForm.grid'WC'Grid'(1 1p-')('Posn' 40 10)('Size' 480 880)('FontObj' 'RTForm.fntGrid')
RTForm.grid.TitleWidth←0
RTForm.grid.TitleHeight←0
'RTForm.grid.edL'WC'Edit'
'RTForm.grid.edD'WC'Edit'
'RTForm.grid.edRO'WC'Edit' ''('ReadOnly' 1)
RTForm.grid.Input←'RTForm.grid.edL' 'RTForm.grid.edD' 'RTForm.grid.edRO'
RTForm.grid.BCol←(255 255 255)(178 201 255)(220 220 220) A BCol: 1=label, 2=data, 3=read-only (gray)
RTForm.grid.CellTypes←1 1p2 A Initial 1x1 grid, all cells type 2 (data/white); overwritten on each selection
RTForm.grid.CellWidths←120
'RTForm.grid'WS'Event' 'CellChanged' 'GUI.RTF_onCellChanged'
```

# Building GUI in APL with Claude

```
hasExcel←0<≠path
```

```
⌈NA'I2 user32|GetAsyncKeyState I4'
```

```
⌈NA'U4 user32|GetForegroundWindow'
```

```
RTF_NSAV←0           A Running count of confirmed saves (displayed in lblCount)
```

```
RTF_SHEET←''         A Name of the Excel sheet that owns the current selection
```

```
RTF_SHEETS←0ρ←''     A Accumulates names of top-level sheet namespaces created this session
```

```
RTF_ND←0           A Auto-naming counter for datasets
```

Less impressive is...

```
noLab←{(ω≡'NoLabel')∨(ω≡'default')∨(ω≡'data')}
```

```
ci←(1 ⌈C''cl)⌈1 ⌈C name
```

```
ri←(1 ⌈C''rl)⌈1 ⌈C name
```

# Building GUI in APL with Claude

hasExcel←0

ONA'I2 user

ONA'U4 user

RTF\_NSAV←0

RTF\_SHEET←

RTF\_SHEETS←

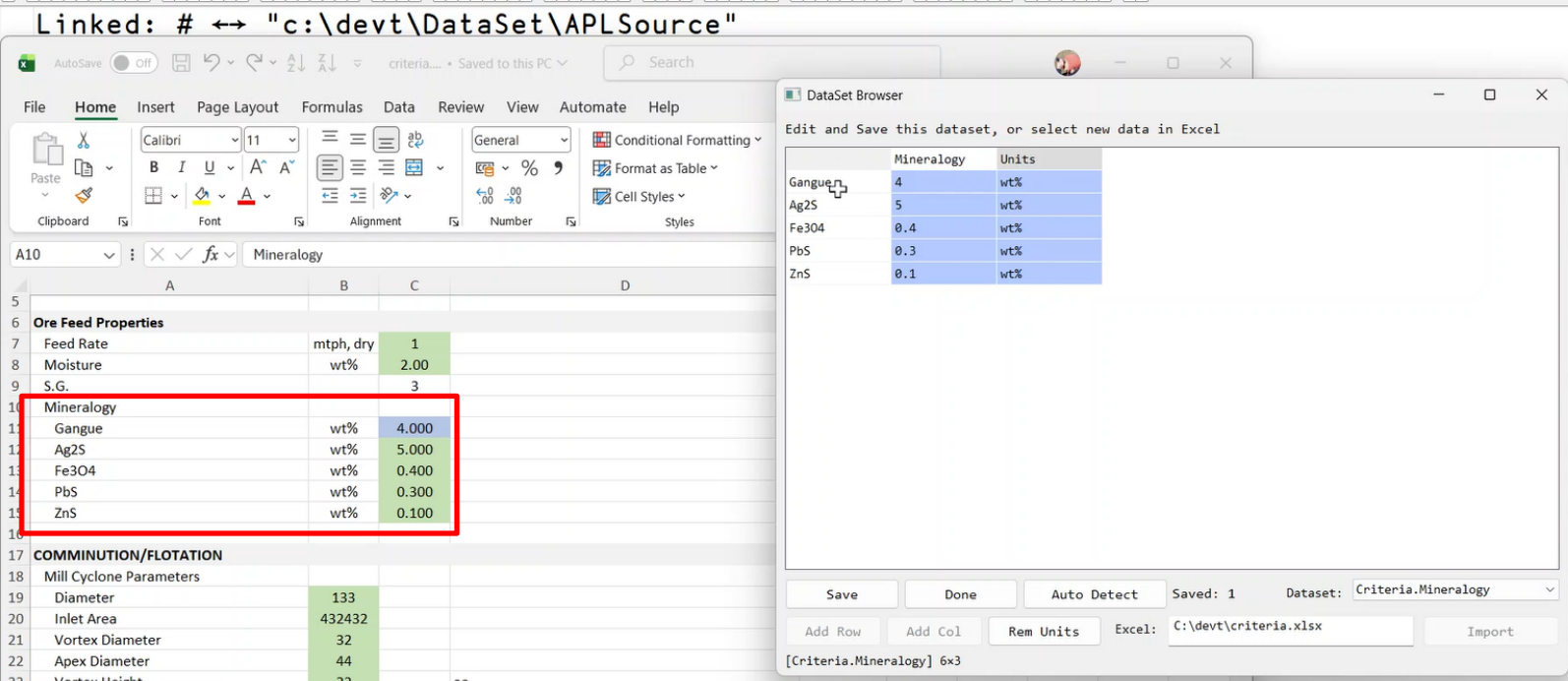
RTF\_ND←0

Less impressive

noLab←{(ω≡'NoLab

ci←(1 ⌊C"cl)⌊c1

ri←(1 ⌊C"rl)⌊c1 ⌊C name



The screenshot displays a Microsoft Excel spreadsheet and a DataSet Browser window. The Excel spreadsheet is titled "c:\devt\DataSet\APLSource" and shows a table with columns A, B, and C. The table is divided into sections: "Ore Feed Properties", "Mineralogy", and "COMMINUTION/FLOTATION". The "Mineralogy" section is highlighted with a red box, showing a table with columns "Mineralogy", "Units", and "Value". The "DataSet Browser" window is open, showing a table with columns "Mineralogy" and "Units". The table contains the following data:

Mineralogy	Units
Gangue	4 wt%
Ag2S	5 wt%
Fe3O4	0.4 wt%
PbS	0.3 wt%
ZnS	0.1 wt%

The DataSet Browser window also shows a "Save" button, a "Done" button, an "Auto Detect" button, and a "Saved: 1" status. The "Dataset:" dropdown is set to "Criteria.Mineralogy". The "Excel:" field shows the file path "C:\devt\criteria.xlsx". The "Import" button is also visible.

# OK, so Claude can do $\square$ WC

- What about cross-platform GUI in APL?



# EWC (Everywhere WC)

- ◆ Cross-platform emulation of WC
  - ◆ Reproduce legacy GUI in the cloud
  - ◆ Also support CSS and Flex to allow modernization
- ◆ Prototype implementation by external consultant was extremely poor
- ◆ Being rebuilt by internal resource [will be re-launched later this year]
- ◆ With some help from Claude ...

# EWC Tests and Tutorial

- Created by Claude based on documentation and a few code samples
- ... guided by ...



Neil Kirsopp



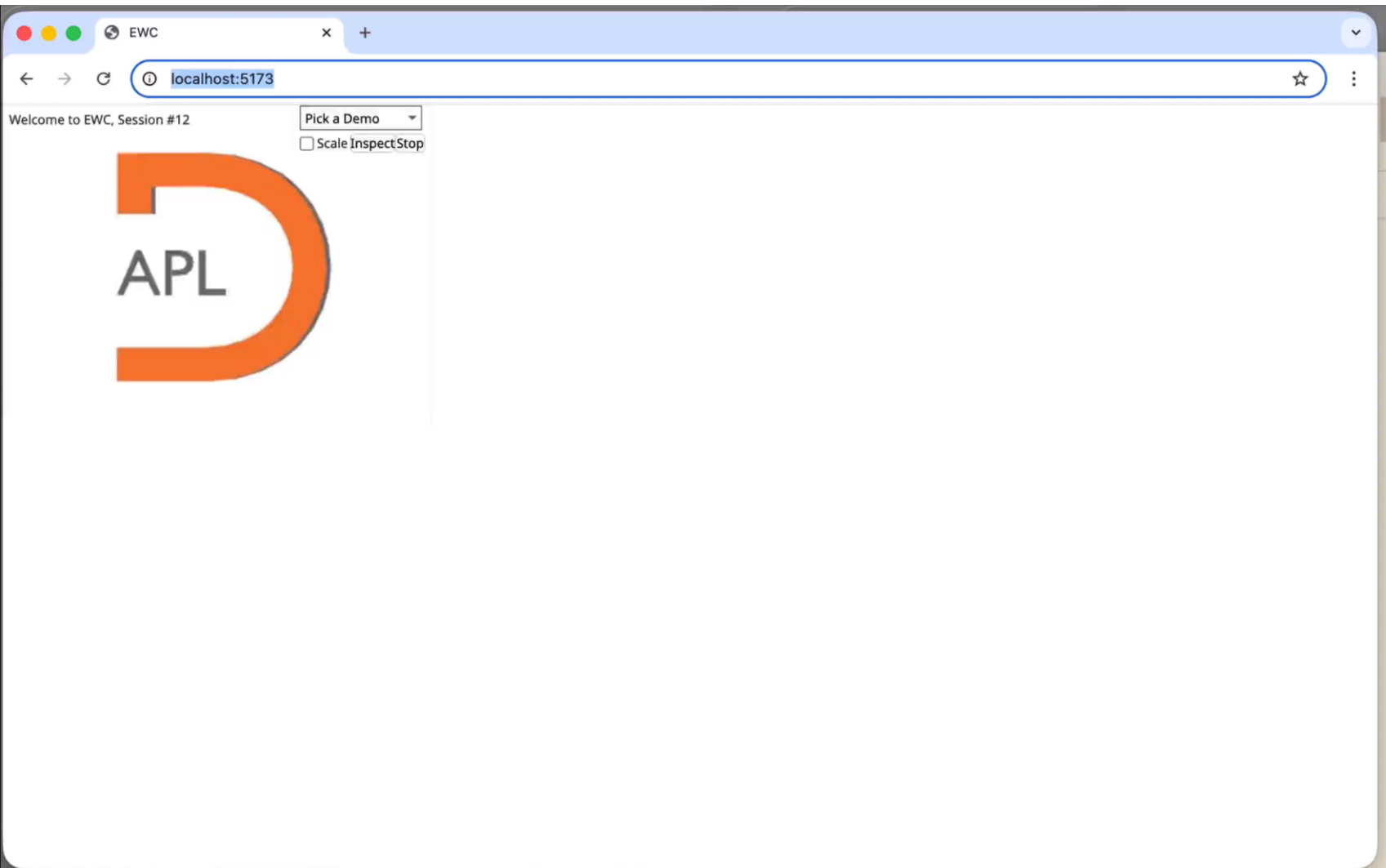
Aarush Bhat



Code

Blame

```
51     test('keyboard navigation through cells', async () => {
52         const grid = page.locator('.nugrid');
53         await grid.click();
54         await page.waitForTimeout(SHORT);
55
56         // Walk down through products
57         for (let i = 0; i < 4; i++) {
58             await page.keyboard.press('ArrowDown');
59             await page.waitForTimeout(BEAT);
60         }
61
62         // Walk right through columns
63         for (let i = 0; i < 3; i++) {
64             await page.keyboard.press('ArrowRight');
65             await page.waitForTimeout(BEAT);
66         }
67         await page.waitForTimeout(SHORT);
68
69         // Home / End / Ctrl+Home
70         await page.keyboard.press('Home');
71         await page.waitForTimeout(SHORT);
72
73         await page.keyboard.press('End');
74         await page.waitForTimeout(SHORT);
75
76         await page.keyboard.press('Control+Home');
77         await page.waitForTimeout(SHORT);
```





## EWC Tutorial

Learn to build web UIs with Dyalog APL, one lesson at a time.

**1 Hello World**  
Your first EWC app: a Form with a Label.

**2 Buttons & Events**  
Add buttons and handle click events with callbacks.

**3 Layout with Flex**  
Use SubForms and Flex for row/column layouts.

**4 Text Input**  
Collect user input with Edit fields.

**5 Lists & Combos**  
Dropdown selection with Combo boxes.

**6 Grids**  
Display tabular data with the Grid component.

**7 Tabs**  
Organise content with TabControl.

**8 Styling**  
Colours, fonts, and CSS customisation.

**9 Charts**  
Data visualisation with ApexCharts.

**10 Putting It Together**  
A complete mini-app combining multiple components.

● lesson.apl

Ctrl+Enter

**A Lesson 3: Layout with Flex****A Row and column layouts using SubForm + Flex**`'F1'eWC'Form' 'Flex Layout' ('Coord' 'Pixel') ('Flex' 'fill') ('BCol' (242 241 240))`**A Header row**`'F1.HDR'eWC'SubForm' ('Flex' 'row') ('BCol' (0 59 92)) ('CSS' 'padding:20px 30px, align-items:ce``'F1.HDR.TITLE'eWC'Label' 'Dashboard' ('FCol' (255 255 255)) ('CSS' 'font-size:20px, font-weight:``'F1.HDR.USER'eWC'Label' 'Welcome, APL Developer' ('FCol' (255 255 255)) ('CSS' 'font-size:13px, i`**A Cards row**`'F1.CARDS'eWC'SubForm' ('Flex' 'row') ('BCol' (242 241 240)) ('CSS' 'padding:24px, gap:16px, fle`**A Card 1**`'F1.CARDS.C1'eWC'SubForm' ('Flex' 'column') ('BCol' (255 255 255)) ('CSS' 'padding:24px, border-``'F1.CARDS.C1.NUM'eWC'Label' '1,247' ('FCol' (255 106 19)) ('CSS' 'font-size:28px, font-weight:70``'F1.CARDS.C1.LBL'eWC'Label' 'Active Users' ('FCol' (102 102 102)) ('CSS' 'font-size:13px, margin`**A Card 2**`'F1.CARDS.C2'eWC'SubForm' ('Flex' 'column') ('BCol' (255 255 255)) ('CSS' 'padding:24px, border-``'F1.CARDS.C2.NUM'eWC'Label' '89%' ('FCol' (137 134 202)) ('CSS' 'font-size:28px, font-weight:700``'F1.CARDS.C2.LBL'eWC'Label' 'Uptime' ('FCol' (102 102 102)) ('CSS' 'font-size:13px, margin-top:4`**A Card 3**`'F1.CARDS.C3'eWC'SubForm' ('Flex' 'column') ('BCol' (255 255 255)) ('CSS' 'padding:24px, border-``'F1.CARDS.C3.NUM'eWC'Label' '42ms' ('FCol' (0 59 92)) ('CSS' 'font-size:28px, font-weight:700')``'F1.CARDS.C3.LBL'eWC'Label' 'Avg Response' ('FCol' (102 102 102)) ('CSS' 'font-size:13px, margin`

Preview

ok

## Dashboard

Welcome, APL Developer

**1,247**  
Active Users**89%**  
Uptime**42ms**  
Avg Response

# LLMs write documentation

Can you read this workspace to get an idea of what it does?

## Workspace Overview

- End-to-end implementation of the Dyalog File Server (DFS): the shared core class defines service states, error codes, configuration flags, and Conga networking hooks that every DFS component inherits ( `DFSCore.dyalog:1` ).
- Service layer modelling: `DFSService` wraps both DRC server/client behaviour (connection tables, heartbeat/timeout management) for any DFS daemon ( `DFSService.dyalog:1` ), while `DFSMaster` extends it to orchestrate configuration, service discovery, user management, and cluster-wide commands ( `DFSMaster.dyalog:1` ).
- Bootstrapping + web plumbing: the MildServer-based loader in `Core/Boot.dyalog` pulls SALT components, extensions, and utility libraries into the workspace, wiring authentication/logging helpers and virtual resources for the monitor UI ( `Core/Boot.dyalog:1` ).
- Configuration + security: XML files in `DFSConfig/` describe system-wide tuning, ports, file mappings, and user/role access matrices that DFS loads at start-up ( `DFSConfig/config.xml:1` , `DFSConfig/servicemap.xml:1` , `DFSConfig/filemap.xml:1` , `DFSConfig/users.xml:1` , `DFSConfig/roles.xml:1` ).
- Monitoring UI: the `DFSMonitor` web app is an APL-driven front end using jQuery/DataTables to show service health, locks, queues, and to trigger start/stop/admin actions ( `DFSMonitor/index.dyalog:1` plus `DFSMonitor/Styles/` & `PlugIns/` ).
- Auth/session plug-ins and helpers: pluggable authentication/session handlers live under `Extensions/` (for example, Basic auth via `Extensions/SimpleAuth.dyalog:1` ), backed by shared utilities for

# LLMs write documentation

Given the content of the repo and the map you just wrote, can you draw (or line out) a logical map of all the most relevant files/classes/functions which implements the DFS?



## DFS Implementation Map

### • Bootstrap & Runtime

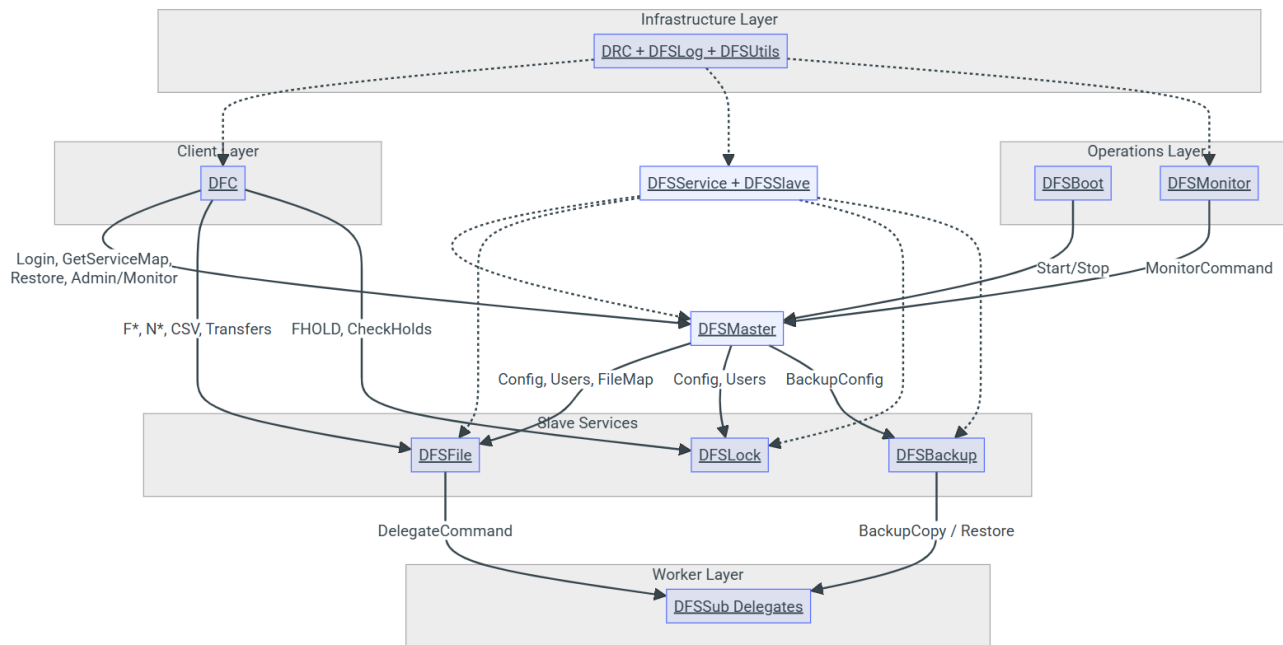
- `Core/Boot.dyalog`
  - Initializes MildServer, loads SALT classes, utilities, and extensions.
  - Resolves environment paths ( `MSRoot` , `AppRoot` ) and loads Monitor/Extensions code.
- `DFSBoot.dyalog`
  - Reads `DFSConfig/*.xml` , builds service map, launches master + auxiliary services via `DFSCore` .
  - Manages start/stop lifecycle and RIDE port registry ( `DFSConfig/RidePort.csv` ).

### • Foundation Layer

- `DFSCore.dyalog`
  - Global enums (service states, error codes), configuration fields, path utilities, casing/validation helpers, random ID generation.
- `DFSConnection.dyalog`
  - Thin wrapper around Conga DRC for client/service communication (sockets, heartbeat, login).
- `DFSLog.dyalog`
  - Channel-based logging, used by all services for `ServiceInfo` , `ServiceError` , etc.

# LLMs write documentation

## DFS Components

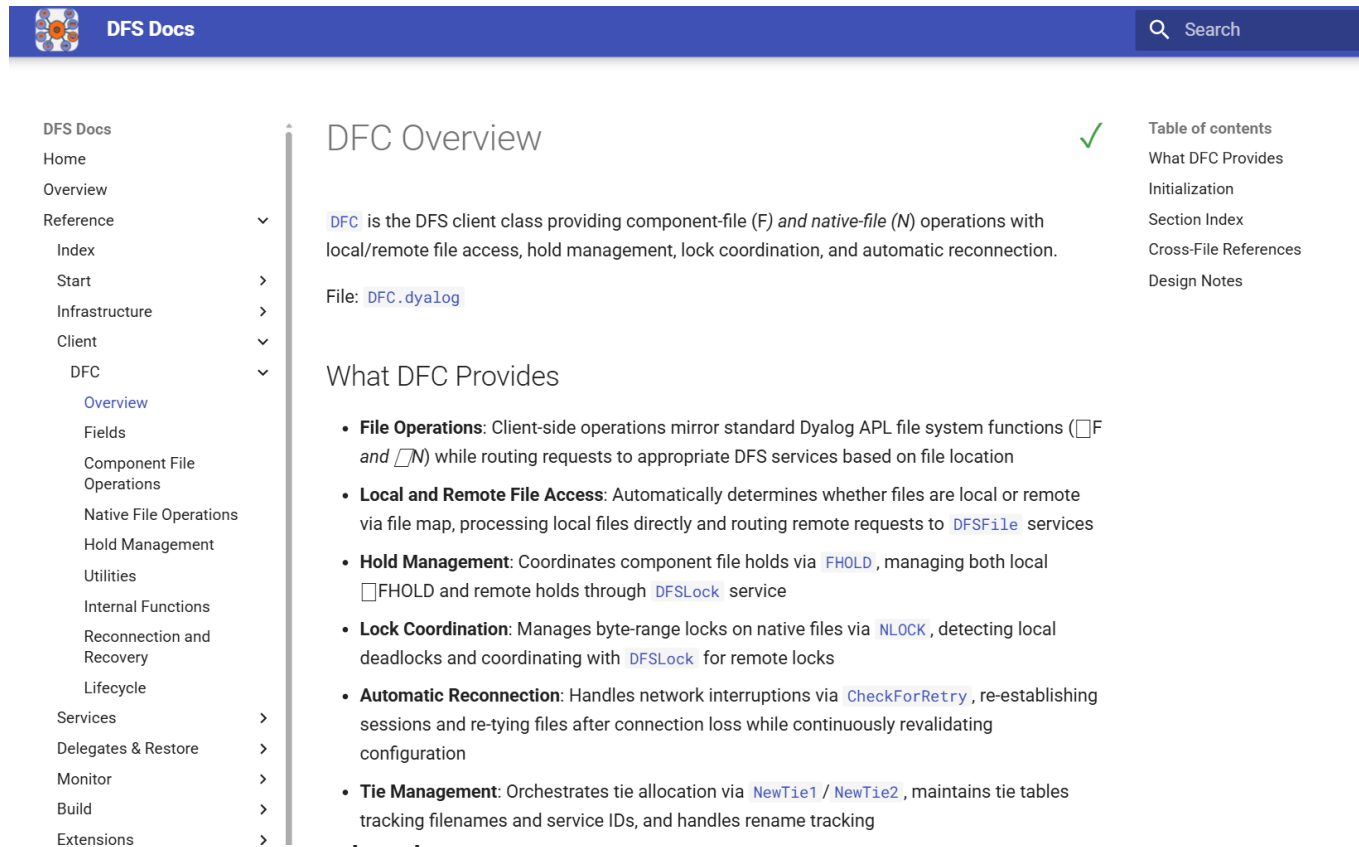




# LLMs write documentation

- Scan the codebase
- Obtain a map
- Write the documentation for each file in a .md file
- Build mkdocs and populate it

- NB A developer MUST review ALL the generated text!!!



The screenshot displays the DFS Docs website. The top navigation bar is dark blue with the DFS Docs logo and a search bar. The left sidebar contains a table of contents with links to various sections. The main content area is titled 'DFC Overview' and includes a green checkmark icon. The text describes the DFC client class and its operations. A section titled 'What DFC Provides' lists several key features: File Operations, Local and Remote File Access, Hold Management, Lock Coordination, Automatic Reconnection, and Tie Management. The right sidebar contains a table of contents for the DFC Overview page.

DFS Docs

Home

Overview

Reference

Index

Start

Infrastructure

Client

DFC

Overview

Fields

Component File Operations

Native File Operations

Hold Management

Utilities

Internal Functions

Reconnection and Recovery

Lifecycle

Services

Delegates & Restore

Monitor

Build

Extensions

DFC Overview

DFC is the DFS client class providing component-file (F) and native-file (N) operations with local/remote file access, hold management, lock coordination, and automatic reconnection.

File: [DFC.dyalog](#)

What DFC Provides

- **File Operations:** Client-side operations mirror standard Dyalog APL file system functions (`□F` and `□N`) while routing requests to appropriate DFS services based on file location
- **Local and Remote File Access:** Automatically determines whether files are local or remote via file map, processing local files directly and routing remote requests to [DFSFile](#) services
- **Hold Management:** Coordinates component file holds via [FHOLD](#), managing both local `□FHOLD` and remote holds through [DFSLock](#) service
- **Lock Coordination:** Manages byte-range locks on native files via [NLOCK](#), detecting local deadlocks and coordinating with [DFSLock](#) for remote locks
- **Automatic Reconnection:** Handles network interruptions via [CheckForRetry](#), re-establishing sessions and re-tying files after connection loss while continuously revalidating configuration
- **Tie Management:** Orchestrates tie allocation via [NewTie1](#) / [NewTie2](#), maintains tie tables tracking filenames and service IDs, and handles rename tracking

Table of contents

What DFC Provides

Initialization

Section Index

Cross-File References

Design Notes

# Building Prototypes

- ◆ LLMs are extremely good at building UI on top of well-documented APIs
- ◆ Dyalog's RIDE protocol is open source
- ◆ Gilgamesh Athoraya once created a VS Code debug adapter communicating with the interpreter via the RIDE protocol
- ◆ Why not ask Claude to create a VS Code plugin that "mashes that up"?



Gilgamesh Athoraya

# VSC Plugin – Interactive APL Session

The screenshot displays the Visual Studio Code editor with the `RealTimeGUI.apl` file open. The code defines various variables for dataset management and GUI interaction. A `DataSet Browser` dialog is overlaid on the editor, showing a table of data with columns `lab`, `c1`, `c2`, and `Units`. The dialog also includes buttons for `Save`, `Done`, `Auto Detect`, and `Import`.

**RealTimeGUI.apl Code:**

```
10 A RealTimeGUI 'file.xlsx' A browse + import from Excel
11 RTF_SHEET=' ' A Name of the Excel sheet that owns the current selection
12 RTF_SHEET=' ' A Accumulates names of top-level sheet namespaces created this session
20 RTF_ND=0 A Auto-naming counter for datasets
21 RTF_KNOWN_UNITS='m' 'km' 'mm' 'ft' 'in' 'kg' 'g' 'mg' 'lb' 'oz' 't' 'ton' 'tons' 's' 'ms' 'min' 'h' 'hr'
22 RTF_KNOWN_UNITS,+ 'm/s' 'km/h' 'mph' 'm' 'm2' 'm3' 'l' 'L' 'ml' 'gal' 'ha' 'ac' 'oF' 'K' 'Pa' 'kPa' 'bar' 'psi'
23 RTF_KNOWN_UNITS,+ 'N' 'kN' 'J' 'kJ' 'HJ' 'kWh' 'BTU' 'W' 'kW' 'MW' 'hp' 'watt' 'V' 'kV' 'A' 'mA' 'Hz' 'kHz' 'MHz' 'X' 'ppm'
24 RTF_KNOWN_UNITS,+ 'wtZ' 'mtpH' 'micron' 'mtpH'
25 RTF_DS_NAMES='RTF_KNOWN_UNITS'
26 RTF_DS_NAMES=0 A Parallel vectors: fully-qualified dataset names ("Sheet.Var")
27 RTF_DS_CUR=' ' A - and matching grid-display namespaces
28 RTF_DS_CUR=' ' A Name of the dataset currently loaded in the grid ('' = none)
29 RTF_RANGE=' ' A Excel range address of the current selection (e.g. '$A$1:$D$5$')
```

**DataSet Browser Dialog:**

lab	c1	c2	Units
a	1	4	cm
b	2	5	mm
c	3	6	[ ]

Buttons: Save, Done, Auto Detect, Saved: 1, Dataset: Sheet1.lab, Add Row, Add Col, Rem Units, Excel: c:/devt/Book1.xlsx, Import

23



# VSC Plugin – Documentation

**RealTimeGUI.aplf**

```
7 A
8 A Usage:
9 A RealTimeGUI ''           A browse existing datasets
10 A RealTimeGUI 'file.xlsx'  A browse + import from Excel
11
12 hasExcel=0<#path
13
14 DNA'I2 user32\GetAsyncKeyState I4'
15 DNA'U4 user32\GetForegroundWindow'
16
17 RTF_NSAY=0           A Running count of confirmed saves (displayed in toolbar)
18 RTF_SHEET=0         A Name of the Excel sheet that owns the current selection
19 RTF_SHEETS=0<#c     A Accumulates names of top-level sheet namespaces created
20 RTF_ND=0             A Auto-naming counter for datasets
21 RTF_KNOWN_UNITS=0<# A Known units: 'm' 'km' 'cm' 'mm' 'ft' 'in' 'kg' 'g' 'mg' 'lb' 'oz' 't' 'ton' 'tonne'
22 RTF_KNOWN_UNITS+=0<# A Known units: 'm/s' 'km/h' 'mph' 'm2' 'm3' 'l' 'L' 'ml' 'gal' 'ha' 'hectare'
23 RTF_KNOWN_UNITS+=0<# A Known units: 'k3' 'HJ' 'kWh' 'BTU' 'W' 'kW' 'MW' 'hp' 'watt' 'VA' 'VA'
24 RTF_KNOWN_UNITS+=0<# A Known units: 'mtpH' 'mtpH' 'micron' 'mtpH'
25 RTF_KNOWN_UNITS+=0<# A Known units: 'mtpH' 'mtpH' 'micron' 'mtpH'
26 RTF_DSNAME=0<#c     A Parallel vectors: fully-qualified dataset names ("Sheet.Var"
27 RTF_DSGRIDS=0<#c     A ... and matching grid-display namespaces
28 RTF_DSCLR=0<#c       A Name of the dataset currently loaded in the grid ('' = none)
29 RTF_RANGE=0<#c       A Excel range address of the current selection (e.g. '$A$1:$D$5')
30
31 A Excel-specific state - initialized to defaults, set properly by RTF_onImportExcel
```

**Name Association** {R}+{X}[DNA Y]

DNA provides access from APL to compiled functions within a library. A library is implemented according to the Operating System as follows:

- a Dynamic Link Library (DLL) under Windows
- a Shared Library (.so or .dylib) under Linux or macOS
- a static library (.a) under AIX

A DLL is a collection of functions typically written in C (or C++) each of which may take arguments and return a result.

Instructional examples using DNA can be found in the supplied workspace `quedna`.

The DLL may be part of the standard operating system software, a library purchased from a third party supplier, or one that you have written yourself.

The right argument Y is a character vector that identifies the name and syntax of the function to be associated. The left argument X is a character vector that contains the name to be associated with the external function. If the DNA is successful, a function (name class 3) is established in the active workspace with name X. If X is omitted, the name of the external function itself is used for the association.

The shy result R is a character vector containing the name of the external function that was fixed.

For example, `math.dll` might be a library of mathematical functions containing a function `divide`.

In a compiled language such as C, the types of arguments and results of functions must be declared explicitly. Typically, these types will be published with the documentation that accompanies the DLL. For example, function `divide` might be declared:

```
double divide(int32_t, int32_t);
```

**Session 1**

Connected

Rebuilding user command cache...  
done

```
QSE.Link.Create '# ' 'c:\devt\DataSet\APLSource'
Linked: # -- 'c:\devt\DataSet\APLSource'
RealTimeGUI ''
```

Done - namespaces:  
RealTimeGUI ''

Done - namespaces: Sheet1

APL

Enter APL expression...

main 01:41 0 0 0 Debug current workspace 20.0 [DataSet]

Finish Setup & 1 | DDQ: 0 | S: 0 | IO: 1 | ML: 1



# VSC Plugin – HTML Graphics

The screenshot displays the Visual Studio Code (VSC) interface with the VSC Plugin for HTML Graphics. The interface is divided into several panels:

- File Explorer (Left):** Shows the project structure with folders like `DATASET`, `.claude`, `.vscode`, `launch.json`, `link.json`, `APLSource`, `DS`, `GUI`, `Tests`, and `RealTimeGUI.aplf`.
- Code Editor (Center):** Displays the `RealTimeGUI.aplf` file. The code includes comments and APL expressions for handling datasets, namespaces, and plotting. Key lines include:

```
7 A  
8 A Usage:  
9 A RealTimeGUI '' A browse existing datasets  
10 A RealTimeGUI 'file.xlsx' A browse + import from Excel  
11  
12 hasExcel=0<#path  
13  
14 [DNA'I2 user32|GetAsyncKeyState I4'  
15 [DNA'U4 user32|GetForegroundWindow'  
16  
17 RTF_NSAY=0 A Running count of confirmed saves (displayed in blc  
18 RTF_SHEET='' A Name of the Excel sheet that owns the current selec  
19 RTF_SHEETS=0<#c'' A Accumulates names of top-level sheet namespaces crea  
20 RTF_ND=0 A Auto-naming counter for datasets  
21 RTF_KNOWN_UNITS='m' 'km' 'cm' 'mm' 'ft' 'in' 'kg' 'g' 'mg' 'lb' 'oz' 't' 'ton' 'ton  
22 RTF_KNOWN_UNITS,+m/s' 'km/h' 'mph' 'm2' 'm2' 'm3' 'l' 'L' 'ml' 'gal' 'ba' 'ac  
23 RTF_KNOWN_UNITS,+N' 'kn' 'J' 'kJ' 'HJ' 'kWh' 'BTU' 'W' 'kW' 'MW' 'hp' 'watt' 'V'  
24 RTF_KNOWN_UNITS,+v'tz' 'mph, dry' 'micron' 'mtp' 'mtp'  
25 RTF_KNOWN_UNITS+,'RTF_KNOWN_UNITS  
26 RTF_DSNAHES=0<#c'' A Parallel vectors: fully-qualified dataset names ('Sheet,Var'  
27 RTF_DSGRIDS=0 A ... and matching grid-display namespaces  
28 RTF_DSCLR='' A Name of the dataset currently loaded in the grid ('' = none)  
29 RTF_RANGE='' A Excel range address of the current selection (e.g., '$A$1:$D$5'  
30  
31 A Excel-specific state – initialized to defaults, set properly by RTF_onI
```
- Plot Window (Right):** Displays a line graph with the following data points:

X	Y
1	1
2	4
3	2
4	5
5	3
6	6
- Session 1 (Bottom):** Shows the APL session output, including the linked path `c:\dev\DataSet\APLSource` and the execution of the `RealTimeGUI` command. The output shows the namespaces `RealTimeGUI` and `LAB` being created, and the plot being displayed.

# VSC Plugin – Claude Code

The screenshot displays the Visual Studio Code (VS Code) interface with the `RealTimeGUI.aplf` file open. The file contains APL code for a real-time GUI. The code includes comments and function definitions for handling datasets, Excel files, and GUI elements.

The Claude Code plugin is visible on the right side of the screen, showing the prompt: "Can you explain RealTimeGUI[14]?" and the response: "Line 14 is a `NA` (Name Association) declaration that binds a Windows DLL function into the APL workspace so it can be called like a normal APL function. Decoded left-to-right: ..."

The response includes a table with the following content:

Piece	Meaning
<code>NA</code>	Declare an external (native) function
<code>'I2 '</code>	Return type — 16-bit signed integer (Win32 <code>SHORT</code> )
<code>user32</code>	DLL name — <code>USER32.DLL</code>
<code>GetAsyncKeyState</code>	Name of the exported Win32 function (also the APL name it becomes bound to)
<code>'I4 '</code>	Argument type — 32-bit signed integer (the virtual-key code)

Below the table, the response explains the function's behavior and its use in the project. It mentions that `GetAsyncKeyState` is a callable APL function and that it is used to detect when the user releases the mouse after dragging a selection in Excel.

The bottom of the screen shows the VS Code status bar with the text "Enter APL expression..." and the "APL" tab selected.

# VS Code Plugin - Summary

- About one week of work by a gifted developer working with Claude
- Code has been reviewed
- Not ready to be supported by Dyalog
- Rumor has it at least half a dozen similar projects exist at client sites



Martina Crippa

# Summary – AI Productivity Gains

- Biggest gains still in the "APL adjacent" areas -- curiously good at .NET, `□NA`, `□WC` and EWC (!)
- Great at writing C# and APL interfaces to this
- Great at writing Web UIs for a Jarvis service
- Good at following the style of existing code, including trad, OO or dfns
- Good at generating -- and running -- unit tests and writing documentation
- Good at finding bugs and flaws in designs and code

# Dyalog Internal AI Policy

- ◆ Anyone wishing to publish or make long-term use of LLM generated materials must take FULL responsibility for the content
  - ◆ The AI might disappear or change its behaviour
  - ◆ The AI might become very much more expensive
- ◆ No personal or customer data may be exposed to an AI without explicit permission
- ◆ Free AI tools without protections are prohibited, except for experimentation with formal approval

# Dyalog's AI Strategy

- Experimenting a lot with LLMs
- Exposing documentation and code to help train LLMs
- Making APL easy for an LLM to call via the command line in order to
  - Pair programme with you
  - Verify generated code
- EVEN more focus on APL source in text files



Stefan Kruger

# Future of APL in an AI World

- ◆ If nobody wants to think any more, does anyone need a "Tool of Thought"?

Can one imagine...

- ◆ APL as an efficient tool of "thought" for LLMs?
  - ◆ The cost of running LLMs will soon matter
- ◆ APL as a better way to write some "prompts"?
  - ◆ English works well if you can refer to pre-existing, well-understood solutions
- ◆ APL as a way to write easily comprehensible, modifiable and verifiable specifications?



Ken Iverson



# Other Dangerous Wildlife



Security requirements potentially make life much harder for the domain expert.



# Software Security at Dyalog

## BSIMM16 Assessment

Independent assessment by  
Black Duck Software  
March 2026

# 30

OF 128 ACTIVITIES OBSERVED

*Top of our peer group  
in Strategy & Metrics*



## What we do well

CEO-led SSG with security at every stage of development • Continuous fuzz testing, SAST on all C/C++ code • Formal open-source approval and CVE monitoring • All staff trained twice per year



## The honest bit

We are doing more than we are getting credit for — but several real gaps remain. Good practices exist across the organization; not all of them are visible, documented, or repeatable enough for an external assessor to verify.



## What we are fixing

- Centralizing security policies and metrics into a single reportable framework
- Commissioning third-party penetration testing of our core product and external services
- Extending static analysis to APL code — C/C++ is covered, APL is next
- Introducing Static Code Analysis tooling and SBOM generation for customers who need full supply chain transparency

# Software Security @ Dyalog



CEO Stine Kromberg  
(with Claude)



## What we are fixing

- Centralizing security policies and metrics into a single reportable framework
- Commissioning third-party penetration testing of our core product and external services
- Extending static analysis to APL code — C/C++ is covered, APL is next
- Introducing Static Code Analysis tooling and SBOM generation for customers who need full supply chain transparency

# Software Bill of Materials - SBOM

- A growing requirement for clients distributing Dyalog APL as a component
- We will include an SBOM with future Dyalog APL distributions



Richard Smith

```

SSSP.txt
]
},
{
  "name": "CongaSSL Dynamic Link Library",
  "SPDXID": "SPDXRef-Package-binary-CongaSSL-Dynamic-Link-Library-30d3c2ef53904e95",
  "versionInfo": "1, 0, 0, 0",
  "supplier": "Organization: MingW-W64 Project. All rights reserved.",
  "originator": "Organization: MingW-W64 Project. All rights reserved.",
  "downloadLocation": "NOASSERTION",
  "filesAnalyzed": false,
  "sourceInfo": "acquired package info from the following paths: \\obj\\apl\\win\\64\\unicode\\tty\\dev\\dbg\\conga37ssl64.dll",
  "licenseConcluded": "NOASSERTION",
  "licenseDeclared": "NOASSERTION",
  "copyrightText": "NOASSERTION",
  "externalRefs": [
    {
      "referenceCategory": "SECURITY",
      "referenceType": "cpe23Type",
      "referenceLocator": "cpe:2.3:a:CongaSSL_Dynamic_Link_Library:CongaSSL_Dynamic_Link_Library:1\\, 0\\, 0\\, 0:::*:*:*:*:*"
    }
  ]
},
{
  "name": "Dyalog APL",
  "SPDXID": "SPDXRef-Package-binary-Dyalog-APL-7bbccce58bebcdafe",
  "versionInfo": "21.0.53092.0",
  "supplier": "Organization: Dyalog Limited",
  "originator": "Organization: Dyalog Limited",
  "downloadLocation": "NOASSERTION",
  "filesAnalyzed": false,
  "sourceInfo": "acquired package info from the following paths: \\obj\\apl\\win\\32\\unicode\\tty\\dev\\dbg\\dyalog.exe",
  "licenseConcluded": "NOASSERTION",
  "licenseDeclared": "NOASSERTION",
  "copyrightText": "NOASSERTION",
  "externalRefs": [
    {
      "referenceCategory": "SECURITY",
      "referenceType": "cpe23Type",
      "referenceLocator": "cpe:2.3:a:Dyalog_APL:Dyalog_APL:21.0.53092.0:::*:*:*:*:*"
    }
  ]
},
},
}

```

# Static Code Analysis

- Search APL code for known vulnerabilities and "code smells"
- Prototype delivered to launch customer this month
  - Several large clients seem interested
- Initially delivered as a consulting project
  - Considering a free community version
- Output formats:
  - JSON format ingestible by SonarQube
  - CSV
  - In-workspace arrays



Brandon Wilson



Aaron Hsu

# Some Implemented Rules

Id	Name	Description	Quality	Severity
SAST-0001	Syntax Errors	Code that intentionally induces a SYNTAX ERROR should be explicitly marked with an ignore directive.	RELIABILITY	LOW
SAST-0002	Insecure connections	Clear text protocols are insecure. Use TLS versions instead.	SECURITY	MEDIUM
SAST-0003	Fixed Random Seed	Setting <code>RL</code> to a static value may break Roll and Deal.	SECURITY	MEDIUM
SAST-0004	Hard-Coded Secrets	Suspicious constant detected. Ensure this isn't a hard-coded secret.	SECURITY	HIGH
SAST-0005	Code Injection	Detected execution of potentially dangerous user-controlled data.	SECURITY	HIGH
SAST-0006	Leaked Locals	Semi-globals that might be forgotten localizations.	RELIABILITY	LOW
SAST-0007	Duplicate Functions	Candidate duplicate functions may want to be unified.	RELIABILITY	LOW
SAST-0008	Shadow after usage	Shadowing a variable after usage may cause confusion	RELIABILITY	MEDIUM
SAST-0009	Naked strand assignments	Unparenthesized stranded assignments might be confused as modified assignment. Consider enclosing in parentheses.	RELIABILITY	MEDIUM
SAST-0010	Unbound variables	These names might trigger a VALUE ERROR.	RELIABILITY	HIGH



# In-Workspace Analytics

## SUMMARY

10	SAST-0001	Syntax Errors
0	SAST-0002	Insecure connections
3	SAST-0003	Fixed Random Seed
1	SAST-0004	Hard-Coded Secrets
5	SAST-0005	Code Injection
37	SAST-0006	Leaked Locals
2	SAST-0007	Duplicate Functions
0	SAST-0008	Shadow after usage
99	SAST-0009	Naked strand assignments
0	SAST-0010	Unbound variables

## 2↑VIEW 'SAST-0005'

```
#.doThing[7] :If readFlag
#.doThing[8]  r←$someData
#.doThing[9] :EndIf

#.ns.anotherFun[42] args←'--verbose' '--output' out
#.ns.anotherFun[43] exit←4⇒SHELL (<'curl');args
#.ns.anotherFun[44] :If exit≠0
```

```
'#.entryPoint1' '#.entryPoint2' CALLGRAPH '#.ns.anotherFun'
#.entryPoint1[17] #.combobulate[7 11 13] #.ns.fizzle[7] #.ns.greet[1] #.ns.anotherFun
#.entryPoint2[12] #.ns.anotherFun
```

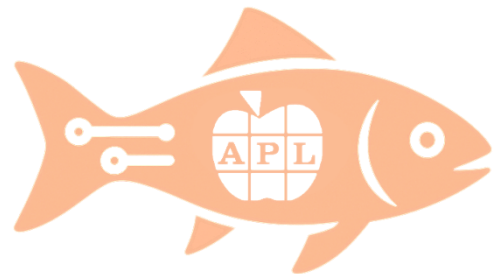
```
'#.entryPoint1' GLOBALS '#.ns.anotherFun'
#.entryPoint1  someVar globalThing setupParam
#.combobulate  anotherVar
#.ns.fizzle
#.ns.greet      greetingMsg
#.ns.anotherFun
```

# Co-dfns update

- Greatly enhanced parser to support Static Analysis project

Coming soon:

- "Lexical" Trad-fns with Branch and Labels
- Performance enhancements and fixes for C backend / Embedded C backend
- Initial Python and JS backends (by End of Year)
- Potential Java backend (Next year, depending on demand)



## Codfns



Aaron Hsu

# Humans



We continue to employ an increasing number of "Ape Coders".

This is the "Consulting, Tools and Data Science" team.

# Consulting, Tools, Data Science

When there is no consulting to do

- Develop productivity tools for APL application development – in APL
- Do research into the Data Science market



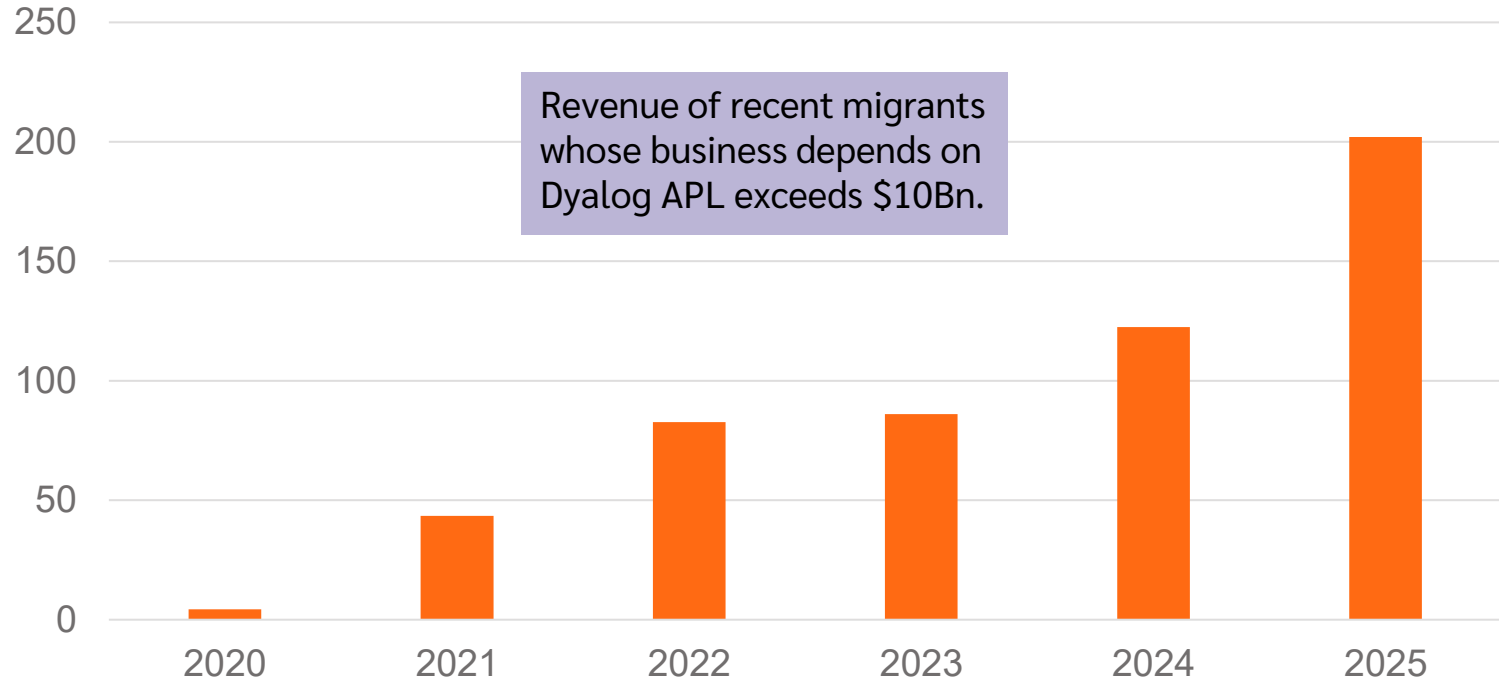
# Consulting

We specialize in:

- Migrations to Dyalog APL
- Migrations to new platforms, e.g. Cloud
  - Dockerization, Continuous Integration
- Using AI with APL
- Architecture reviews and performance tuning



## Revenue from Recent Migrants (kUSD)



# Productivity Tools in APL

**HttpCommand** – utility for making HTTP requests and consuming web services.

Recent enhancements:

- ◆ Server Sent Events (good for talking to LLMs)
- ◆ Convert `www-form-urlencoded` results into namespaces
- ◆ Upload files using `content-type multipart/form-data`
- ◆ Automatic parsing of JSON responses, making API consumption straightforward.





# Productivity Tools in APL

**Jarvis** - web service framework that lets you build JSON or REST APIs.

Recent enhancements:

- JSON mode – endpoints are APL functions.
  - TryAPL and the APL Challenge site use Jarvis in JSON mode
- REST mode – endpoints are resources (real or virtual)
  - DCMS (see Rich's presentation later today) uses Jarvis in REST mode



# Data Science

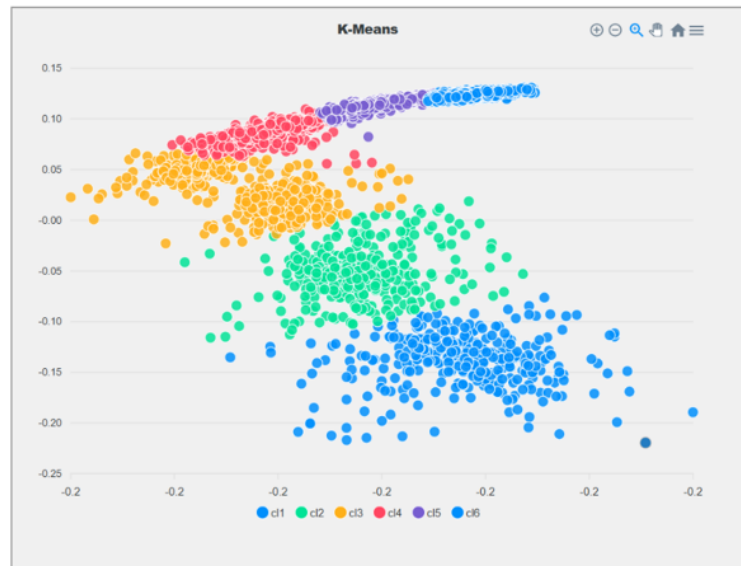
- APL used to be the best tool for DS
- Today, Python has more, better libraries
- We are building a team to understand
  - Where the opportunities are for APL
  - How to make ourselves more visible
  - Where we need to improve to be competitive
  - How to share tools & data with Python



# Data Science Findings

We need to work on:

- Faster Data Loading **Next**
- Faster Matrix Multiplication ✓
- Easy access to LLMs ✓
- New Visualisation Tools
- Arrow & Parquet Data Formats
- Python Integration
  - Use Python packages from APL
  - Use APL as a package from Python



# Core Product Directions

- ◆ Increased focus on speed-ups and other features to improve the "data science experience"
- ◆ Make APL a better command-line tool
  - ◆ Not only to please LLMs
- ◆ Make interfaces Open-Source
  - ◆ HTMLRenderer, Conga (TCP), Crypto etc
- ◆ Take full advantage of Array Notation

# Array Notation

<pre>z ← , c 'Three' z ← , c 'Blind' z ← , c 'Mice'</pre>	=	<pre>('Three' 'Blind' 'Mice')</pre>
---	---	-------------------------------------

<pre>z ← , 0 6 1 8 z ← , 1 4 1 4 z ← , 2 7 1 8 z ← , 3 1 4 2</pre>	=	<pre>[0 6 1 8 1 4 1 4 2 7 1 8 3 1 4 2]</pre>
--	---	--

<pre>z ← , 10 z ← , 20 z ← , 30 z ← , 40</pre>	=	<pre>[10 20 30 40]</pre>
--	---	--------------------------

# Namespace Notation

## □VGET and □VSET

```
people ← (name: 'Jack' ♦ weight:75) (name: 'Jill')
```

```
people.name
```

```
Jack  Jill
```

```
people □VGET 'name' ('weight' 50) ▸ Default weight=50
```

```
Jack  75  Jill  50
```

# Array Notation: To come in v22.0

- APLAN to format and execute array notation
- Enhance system functions to use namespace notation

```
■ SIGNAL (EN:11 ♦ EM:'USER ERROR'  
          ♦ Message:'problem between screen and chair')
```

```
■ NEW 'Timer' (Active:0 ♦ Interval:500)
```

# APL Libraries

- ◆ Add language support for loading modules (`⎕IMPORT`)
- ◆ Simplify the package manager for APL (Tatin)
- ◆ Add a command-line interface to install packages
- ◆ Write more packages
- ◆ Find a way to get YOU to write packages





# Parallel / Async Operator ||

- Implement a primitive operator || which can invoke functions asynchronously
  - A future is immediately returned (as with isolates today)
  - The interpreter will block automatically if the value is used and has not materialised.
- Multiple architectures will be supported simultaneously
  - Isolates (remote processes w/TCP sockets)
  - "Green threads" (as the existing & operator)
  - Multiple interpreters running on separate O/S threads within the same process



Peter Mikkelsen



John Daintree

# Compilation

- The `||` operator will support parallelism using multiple interpreters
- Effective use of GPUs requires a compiler
- Enter `codfns`

