



Dyalog North America Meetup, 11 April 2024

# Migrating to Dyalog

*Morten Kromberg, CTO*

# A New Wave of Migrants

- ◆ Dyalog APL was created by Dyadic Systems Ltd, when the mainframe business faltered (1981)
- ◆ Almost all users of Dyalog APL migrated at some point, from SHARP APL, IBM APL2, APL+Win, or APLX (or DEC APLSF, or ...)
- ◆ Waves of migrants
  - ◆ Death of mainframes and minicomputers (1980's)
  - ◆ Superior support for Windows GUI (1990's)
  - ◆ Now, "the cloud" (& a few more mainframes being shut down)

# Why migrate to Dyalog?

- ◆ Dyalog re-invests 90-95% of revenues in
  - ◆ Enhancing APL core technology
  - ◆ Creating tools for APL developers
  - ◆ Marketing APL outside the current APL community
- ◆ Headcount now 26, up from 4 before the acquisition in 2005
  - ◆ Next generation of developers and toolsmiths has been hired
- ◆ Combined revenues of products and services based on Dyalog APL exceeds \$1Bn per year – and is growing

# Why migrate to Dyalog?

*The largest sustained investment in APL technology  
in the history of the language.*

Morten Kromberg



## The Dyalog Language Engine

At the heart of Dyalog is an [ISO/IEC 13751](#)-compliant APL language engine that has been tuned and optimised for more than 30 years. The current Dyalog language has evolved from a classical APL2-style interpreter into a modern, multi-paradigm programming language. The most important extensions to the original APL language include:

**1983:** Nested arrays: Any element of an array can be another array (APL2)

**1990:** Namespaces

**1995:** Control structures (If/Then/Else, Repeat/Until, exception handling, and so on)

**1996:** Functional programming: dfns provide lexical scope and lambda-style expressions

**2006:** Object orientated programming, allowing integration with OO frameworks and Microsoft .NET

**2014:** Point-free or "tacit" syntax similar to that in the J programming language

**2014:** Futures and isolates for parallel programming

New [versions](#) of Dyalog are released approximately annually.

Dyalog language engines provide the same language features on [all platforms](#) and enable extreme inter-operability; binary workspace images and component files can be shared in real time without conversion between all platforms and TCP sockets can be used to exchange binary data between the platforms.



# Selected Features 2006-2024

- Web Server and Web Service Frameworks
- Run APL as a Windows Service
- Public Docker Containers
- Remote IDE for debugging service processes
- Health Monitor for monitoring collections of processes
- Parallel and Asynchronous Execution
- New Data Types:
  - 128-bit Decimal Floating Point
  - Complex Numbers
- Functional Programming (dfns)
- New primitives: Key, Stencil, Where, ...
- Significant steps towards an APL compiler
- Many speed-ups of interpreter algorithms
- Object Orientation
- Microsoft.Net Integration
- HTMLRenderer object embeds Chromium Web Browser engine
- 64-bit: \*NO\* workspace or component file size limits
- Unicode Support incl APL Source in Text Files
- Secure TCP Sockets w/ IPv6 Support
- Encryption Toolkit
- Regular Expressions (PCRE) built-in to APL
- XML and JSON parsers for fast conversion to (and from) APL structures

Vast majority of features are identical across all platforms

# Selected Features 2006-2024

- Web Server and Web Service Frameworks
  - Run APL as a Windows Service
  - Public Docker Containers
  - Remote IDE for debugging service processes
  - Health Monitor for monitoring of processes
  - Parallel and Asynchronous Execution
  - New Data Types:
    - 128-bit Decimal Floating Point
    - Complex Numbers
  - Functional Programming (dfns)
  - New primitives: Key, Stencil, Where, ...
  - Significant steps towards an APL compiler
  - Many speed-ups of interpreter algorithms
  - Object Orientation
  - Microsoft.Net Integration
  - HTMLRenderer object embeds Chromium Web Browser engine
  - Unicode support incl APL source in Text Files
  - Secure TCP Sockets w/ IPv6 Support
  - Encryption Toolkit
  - Regular Expressions (PCRE) built-in to APL
  - XML and JSON parsers for fast conversion to (and from) APL structures
- 64-bit: \*NO\* workspace or component file size limits ("Legacy" 32-bit system still available)

Vast majority of features are identical across all platforms

# Why Migrate to Dyalog?

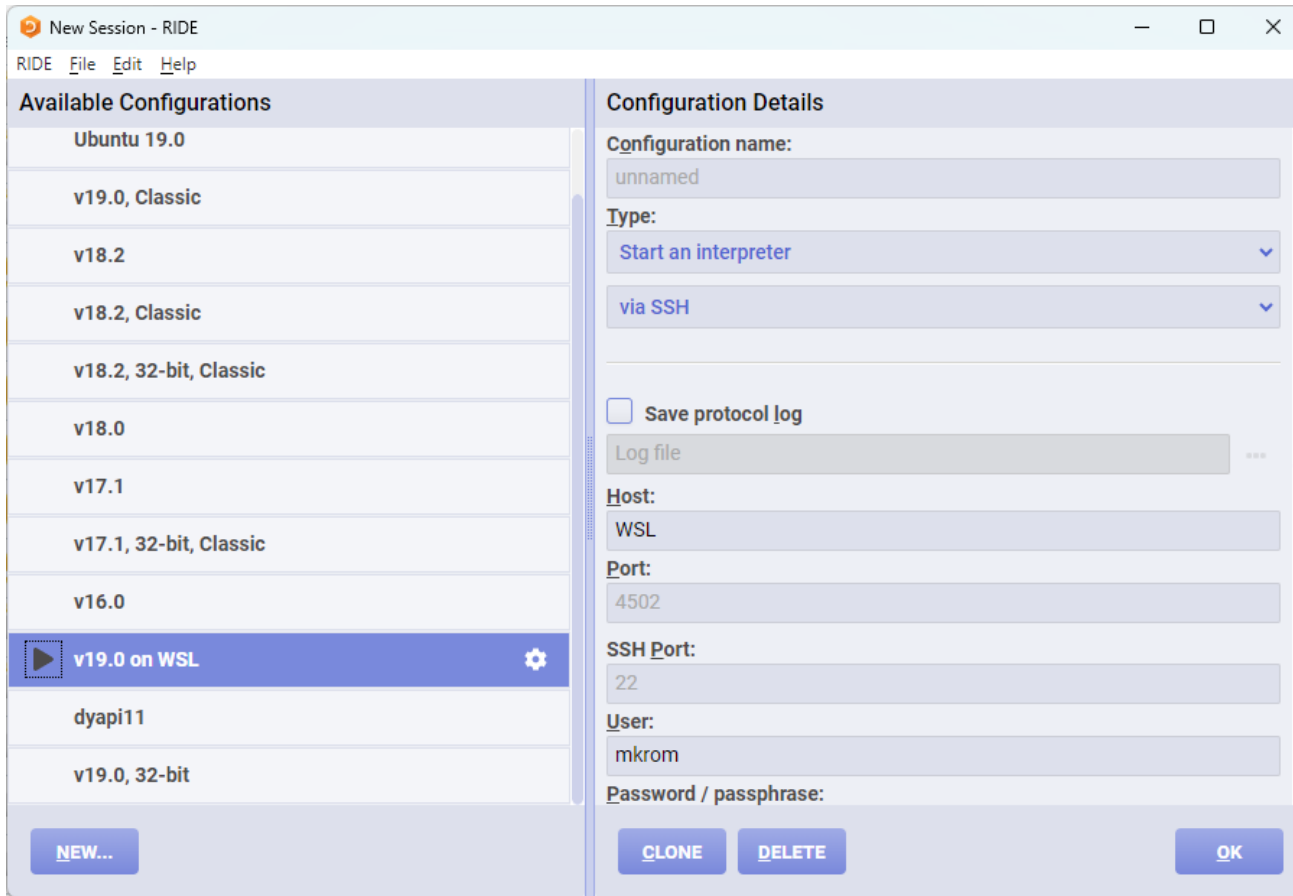
Dyalog is 100% Cross Platform

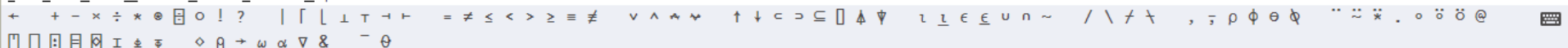
- ◆ Born under UNIX (Solaris, AIX, ...)
  - ◆ Ported to DOS, Windows, Linux (ARM, Intel), MacOS (Intel, Mx)
- ◆ Single source for all platforms
  - ◆ Workspaces and component files compatible across all platforms
- ◆ All tools are tested on all platforms
  - ◆ Exceptions where O/S does not provide a feature
  - ◆ .NET not under AIX, many Windows features like DDE, COM/OLE, GUI



# Remote IDE (RIDE)

- From Windows, Linux or MacOS
- Connect to and debug Dyalog APL running on any platform





Dyalog APL/S-64 Version 19.0.48624

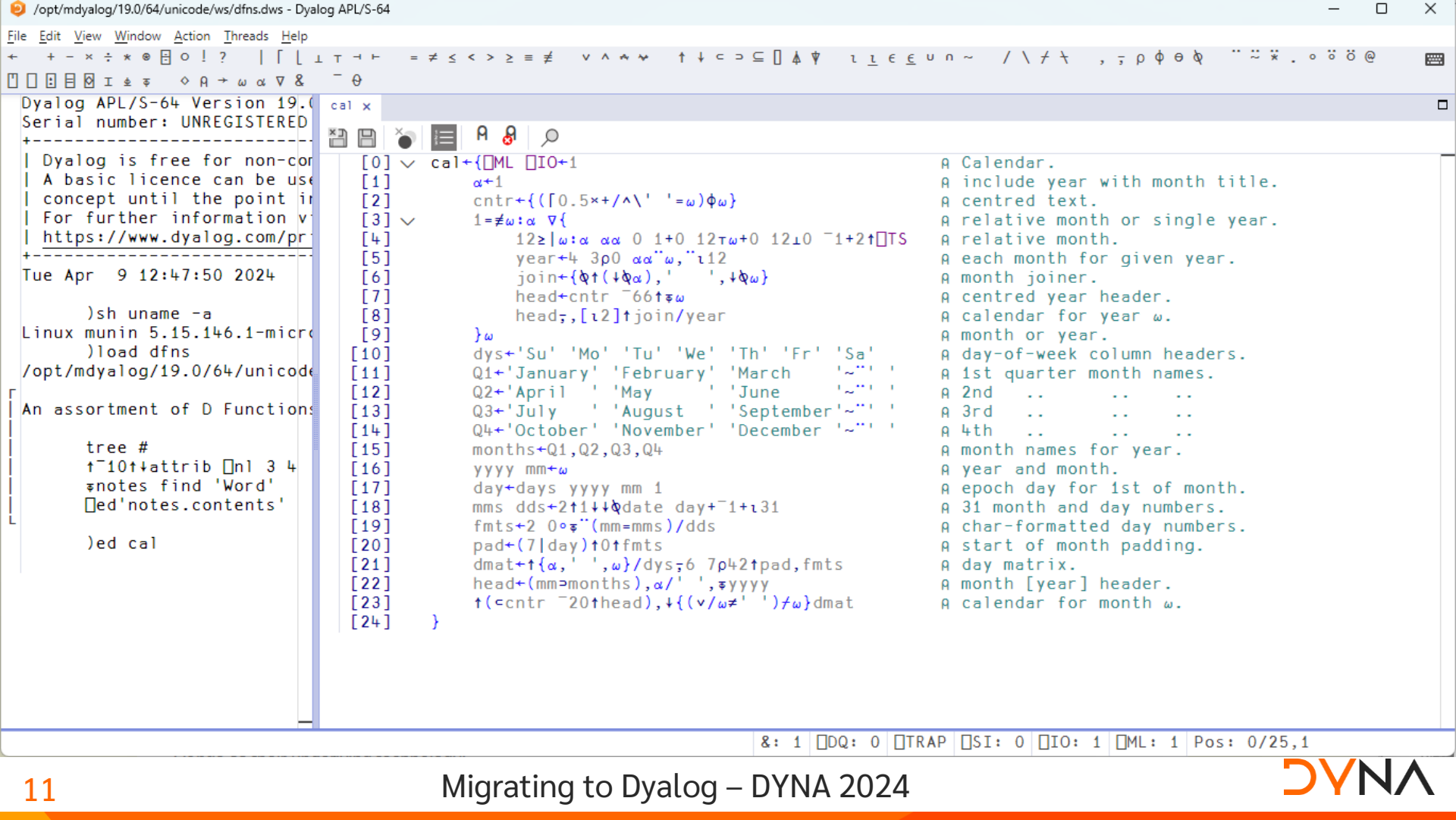
Serial number: UNREGISTERED - not for commercial use

```
+-----+
| Dyalog is free for non-commercial use but is not free software. |
| A basic licence can be used for experiments and proof of       |
| concept until the point in time that it is of value.           |
| For further information visit                                   |
| https://www.dyalog.com/prices-and-licences.htm |
+-----+
```

Tue Apr 9 12:47:50 2024

```
)sh uname -a
```

```
Linux munin 5.15.146.1-microsoft-standard-WSL2 #1 SMP Thu Jan 11 04:09:03 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
```



# Remote IDE (RIDE)

- ◆ Connect to and debug Dyalog APL running on any platform
- ◆ From Windows, Linux or MacOS
- ◆ Or indeed a browser running anywhere...
  - ◆ "Zero Footprint RIDE"

# RIDE running in a browser

The screenshot shows a web browser window with the address bar displaying 'wsl:8888'. The browser's address bar also shows the path '/opt/mdyalog/19.0/64/unicode'. The browser's toolbar includes navigation buttons, a search bar, and various icons. The main content area of the browser displays a Dyalog workspace interface. The workspace has a menu bar with 'Edit', 'View', 'Window', 'Action', 'Threads', and 'Help'. Below the menu bar is a toolbar with various icons for editing and navigation. The workspace content is divided into two panes. The left pane shows a list of functions and their descriptions:

```
)load dfns
/opt/mdyalog/19.0/64/unicode/ws/dfns.dws A saved Mon Jan 29 11:55:11 AM

An assortment of D Functions and Operators.

tree #           A Workspace map.
↑~10↑↑attrib [n] 3 4 A What's new?
↑notes find 'Word' A Apropos "Word".
↑ed'notes.contents' A Workspace overview.

)sh uname -a
Linux munin 5.15.146.1-microsoft-standard-WSL2 #1 SMP Thu Jan 11 22:03:10 UTC 2024 x86_64
)ed cal
```

The right pane shows a script for generating a calendar, with the following code:

```
cal x
[0] |cal+{[ML] [IO]+1
[1]   α+1
[2]   cntn+{([0.5x+/\ ' '=ω)φω}
[3]   1=≠ω:α ∇{
[4]     12z|ω:α αα 0 1+0 12τω+0 12±0 -1+2↑[TS
[5]     year+4 3p0 αα"ω,"i12
[6]     join+{φ↑(↑φα),' ',↑φω}
[7]     head+cntn -66↑≠ω
[8]     head;,[i2]↑join/year
[9]   }ω
[10]  dys+ 'Su' 'Mo' 'Tu' 'We' 'Th' 'Fr' 'Sa'
[11]  Q1+'January' 'February' 'March' '~...'
[12]  Q2+'April' 'May' 'June' '~...'
[13]  Q3+'July' 'August' 'September' '~...'
[14]  Q4+'October' 'November' 'December' '~...'
[15]  months+Q1,Q2,Q3,Q4
[16]  yyyy mm+ω
[17]  day+days yyyy mm 1
[18]  mms dds+2↑1↑↑date day+~1+131
```

The status bar at the bottom of the workspace shows: '&: 1 [Q]Q: 0 [TRAP] [SI: 0 [IO: 1 [ML: 1 Pos: 0/25,1

# Why Migrate to Dyalog?

Dyalog is "Cloud Ready"

- ARM and Intel Linux versions
- Public Docker containers
- Did I mention the "Remote IDE" ?
- Text-based source supports "Continuous Integration"
  - Build & deploy containers on commit or push
- User community starting to gain significant experience
- Working on tools to port Windows GUI to HTML/JS

# Why Migrate to Dyalog?

- ◆ Multiple published licencing models
  - ◆ Custom contracts available
- ◆ Free basic licence for commercial use up to annual revenues of GBP 5,000
- ◆ Free download without providing personal details
- ◆ Unlimited support (within reason 😊) for users with any type of license



# Why Migrate to Dyalog?

Developer tools are free, cross-platform and mostly open source:

Name	Description
SQAPL	ODBC Interface (also ADO and ADO.NET)
Jarvis	HTTP/JSON and REST service framework
HttpCommand	HTTP client
SAWS	SOAP service framework
Conga	TCP and UDP layer
SharpPlot	Business and Technical graphics
<input type="checkbox"/> XML, <input type="checkbox"/> JSON, <input type="checkbox"/> CSV	Built in to interpreter

Name	Description
RConnect	Interface to R
MiServer / DUI	Web Application Framework
Docker Containers	Published examples
Link	Interface to source code management
APLProcess	And isolates

# Why Migrate to Dyalog?

## Emerging Tools

Name	Description
Link	Interface to source code management
Cider	Project Management
Tatin	Package Manager
NuGet	Interface to .NET Packages
Selenium	Automated GUI testing
Jupyter	Jupyter notebooks containing APL
eWC	JavaScript emulation of Win32 GUI
Arrow & Parquet	Data Science data formats

## Separately Licensed Tools

Name	Description
DFS	Dyalog File Server ("SHAREFILE")
Static Analysis	Static Analysis of APL Code (code linting and vulnerability detection) Planned for 2025

<b>bpbecker</b>	50 fix boundary issue in multipartform data (#52)	b8629e8 · last week	🕒 161 Commits
📁 docs	50 fix boundary issue in multipartform data (#52)		last week
📁 source	Fix #50 (#51)		3 weeks ago
📁 tests	remove 819 ibeam from httpserver code (#42)		2 months ago
📄 .gitignore	Implement BaseURL #19		9 months ago
📄 CITA.json5	Docn edits		2 years ago
📄 LICENSE	Initial commit		3 years ago
📄 README.md	Update README.md		2 years ago
📄 apl-package.json	v5.4.6 (#46)		2 months ago

### About

A utility to manage HTTP requests from APL

[dyalog.github.io/HttpCommand/](https://dyalog.github.io/HttpCommand/)

- 📖 README
- 📄 MIT license
- 📈 Activity
- 📁 Custom properties
- ★ 10 stars
- 👁 5 watching
- 🍴 2 forks

Report repository

---

### Releases 26

- HttpCommand
  - Overview
  - Usage
    - Quick Start
    - Usage Guide
    - Examples
    - Troubleshooting
  - Reference
    - Settings >
    - Shared Methods >
    - Instance Methods
    - Result Namespace >
    - Messages and Return Codes
  - Advanced Topics
    - Request Content Types
    - HttpCommand and Conga
    - Secure Communications
    - Using a Proxy Server
    - Integrating HttpCommand
  - About

# Overview

HttpCommand is a utility designed to make it easy for the APL user to send requests to and receive responses from HTTP servers like web servers and web services.

**Note**

While HttpCommand itself is IO and ML insensitive, the examples in this documentation assume an environment of (IO ML)+1.

## Loading HttpCommand

HttpCommand is included with your Dyalog APL installation. To bring it into your workspace:

### Dyalog APL Version 19.0 and later

```
]import {ns} HttpCommand
```

or, under program control, do:

```
SE.Link.Import {ns} 'HttpCommand'
```

### On this page

- Loading HttpCommand
  - Dyalog APL Version 19.0 and later
  - Dyalog APL versions before 19.0
- Upgrading to the Latest HttpCommand
- HttpCommand is Available as a Tatin Package
- Typical Usage Patterns
- Philosophy
- Dyalog Classes
- Use of Conga
- Further Reading

Filters is:issue is:closed Labels 9 Milestones 0 New issue

Clear current search query, filters, and sorts

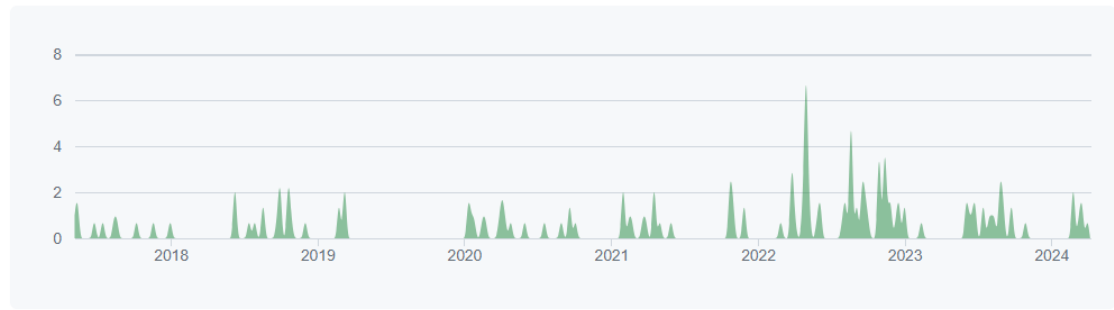
<input type="checkbox"/>	0 Open ✓ 29 Closed	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	✓ <b>Fix boundary issue in multipart/form-data</b> <span>bug</span> #50 by bpbecker was closed 3 weeks ago				1		
<input type="checkbox"/>	✓ <b>Make it easier to use multipart/form-data content</b> <span>enhancement</span> #48 by bpbecker was closed 3 weeks ago						
<input type="checkbox"/>	✓ <b>Year in <code>HttpCommand.Version</code></b> #45 by MaxCan-Code was closed on Feb 28				1		
<input type="checkbox"/>	✓ <b>Don't issue "No URL" error if BaseURL is set but URL is not</b> #43 by bpbecker was closed on Feb 26				1		
<input type="checkbox"/>	✓ <b>Don't attempt to parse payload on a "HEAD" command or empty payload</b> <span>bug</span> #39 by bpbecker was closed on Oct 31, 2023				1		
<input type="checkbox"/>	✓ <b>Classic Only: Disable HttpCommand.Upgrade</b> <span>enhancement</span>				1		

- Pulse
  - Contributors
  - Community
  - Community Standards
  - Traffic
  - Commits
  - Code frequency
  - Dependency graph
  - Network
  - Forks
- People

### May 7, 2017 – Apr 9, 2024

Contributions: Commits

Contributions to master, excluding merge commits



**bpbecker** #1  
 148 commits 9,196 ++ 4,042 --

This chart shows the commit activity for user bpbecker. The y-axis ranges from 0 to 5. The x-axis shows years from 2018 to 2024. The activity is represented by orange vertical bars, showing a steady increase in activity over time, with a notable peak of 5 commits in late 2023.

**abrudz** #2  
 2 commits 6 ++ 5 --

This chart shows the commit activity for user abrudz. The y-axis ranges from 0 to 5. The x-axis shows years from 2018 to 2024. The activity is represented by orange vertical bars, showing a single commit in early 2020 and another single commit in early 2024.



# Why Migrate to Dyalog?

Dyalog APL is carefully designed to last. For example:

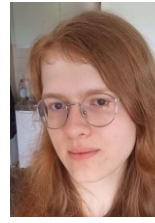
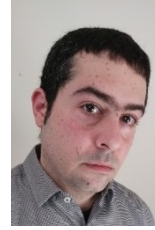
- ◆ Dyalog APL is tightly integrated with .NET
  - ◆ ... and still supports the old .NET Framework
  - ◆ However, Dyalog APL does not and WILL NOT depend on .NET
  - ◆ It also runs under IBM AIX, where .NET does not exist
- ◆ Dyalog *\*will\** remain very portable and independent of "temporary" frameworks

# Why Migrate to Dyalog?

- ◆ Dyalog APL is fast!
- ◆ Core algorithms regularly updated to take advantage of new hardware and new theory
- ◆ Research into a compiler continues



# The Real Reason to Pick Dyalog APL



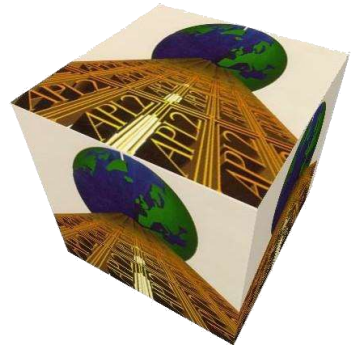
# HOW to Migrate to Dyalog APL

1. From IBM / Logon APL2
2. From APL+Win or MicroAPL APLX

# From APL2

Relatively straightforward

- ◆ A few language differences
- ◆ User Interfaces and file I/O are usually handled by cover-functions and possible to emulate automatically
- ◆ Linux or Windows apps may be making external calls which will require "tweaking"
- ◆ We are considering implementing "format by example" but so far it has not been necessary



# Recent / Active APL2 Migrations

- ◆ Insurance company
  - ◆ No UI, manipulates text and Excel files
  - ◆ Handled by European Consulting Partner
- ◆ Sandvik (Sweden) – in progress: Mainframe APL2 direct to Docker Containers and HTML/svg
  - ◆ Handled by Tiamatica in Malmö (Gilgamesh Athoraya)
- ◆ BIG Jewellers: Windows
  - ◆ Handled by Mark Wolfson himself "with a little help"
- ◆ Two more under discussion
  - ◆ (Germany, Canada)

# Migrated APL2 Mainframe UI

```
Locate Sort
CAPP/COR TEST ----- Routine definition - variables ----- 23-11-10 13:00

Routine.....: X802WM Saved: 23-10-05 12:04 by: STC
Description...: TEST AV SOAP GETLANGS WEBSERVICE
Open for enhanced dialog: Y Yes/No
Prompt variable that contains the information "grade": _____
Var. Cha
Name Num Length Type Send Explanation Line 20 of 99
ART C L X ARTICLE
BART C L ARTICLE
BB C L A DUMMY
CA C L CHARACTER DUMMY
CA1 C L DUMMY
CA2 C L X DUMMY
CA3 C L X DUMMY
CA4 C L A DUMMY
CA5 C L DUMMY
CA6 C L DUMMY
CB C L CHARACTER DUMMY
CC C L CHARACTER DUMMY
CD C L X CHARACTER DUMMY
CE C L A CHARACTER DUMMY
CF C L CHARACTER DUMMY
CG C L CHARACTER DUMMY
CH C L A CHARACTER DUMMY
CHA C L CHARACTER DUMMY
CHA1 C L --
CHA2 C L DUMMY

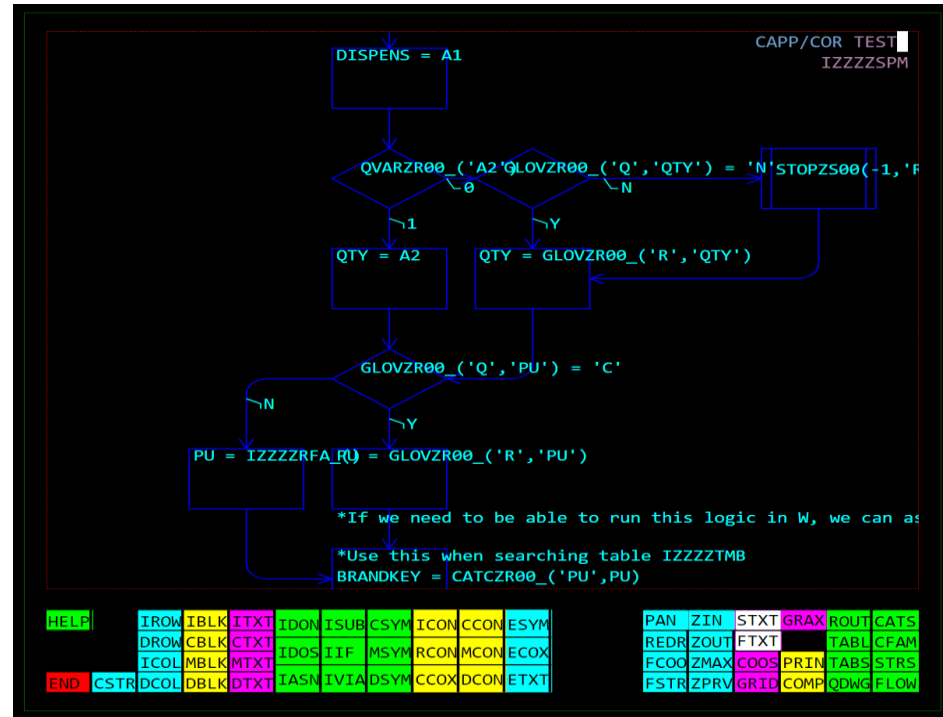
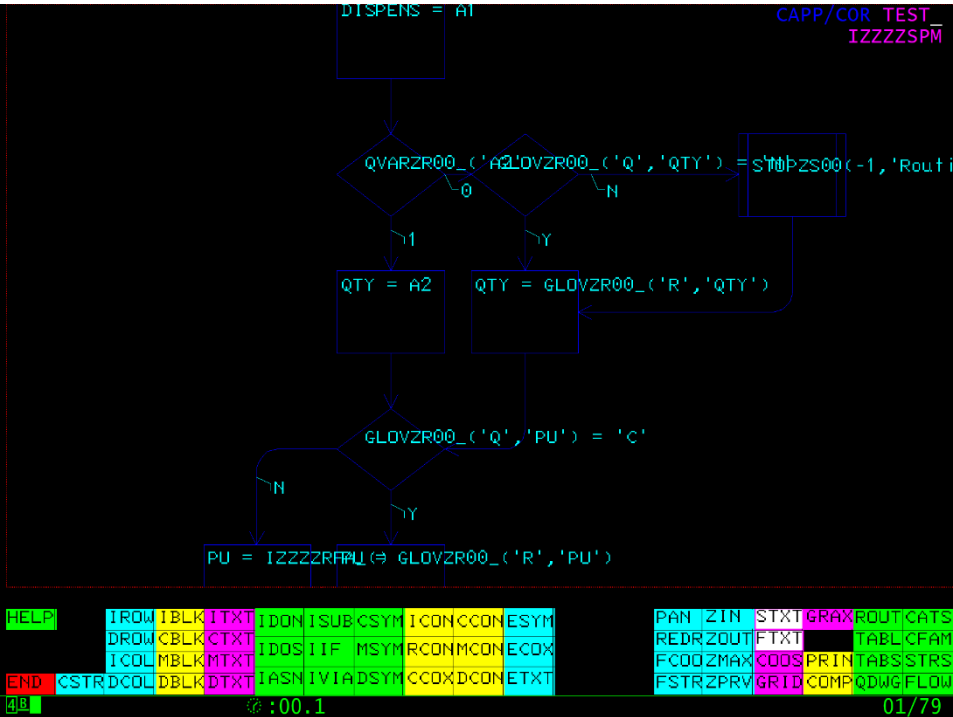
F1=Help F3=End F6=Prompt F7=Up F8=Down
```

```
Locate Sort
CAPP/COR TEST ----- Routine definition - variables ----- 23-11-10 13:04

Routine.....: X802WM Saved: 23-10-05 12:04 by: STC
Description...: TEST AV SOAP GETLANGS WEBSERVICE
Open for enhanced dialog: Y Yes/No
Prompt variable that contains the information "grade": _____
Var. Cha
Name Num Length Type Send Explanation Line 20 of 99
ART C L X ARTICLE
BART C L ARTICLE
BB C L A DUMMY
CA C L CHARACTER DUMMY
CA1 C L DUMMY
CA2 C L X DUMMY
CA3 C L X DUMMY
CA4 C L A DUMMY
CA5 C L DUMMY
CA6 C L DUMMY
CB C L CHARACTER DUMMY
CC C L CHARACTER DUMMY
CD C L X CHARACTER DUMMY
CE C L A CHARACTER DUMMY
CF C L CHARACTER DUMMY
CG C L CHARACTER DUMMY
CH C L A CHARACTER DUMMY
CHA C L CHARACTER DUMMY
CHA1 C L --
CHA2 C L DUMMY

F1=Help F3=End F6=Prompt F7=Up F8=Down
```

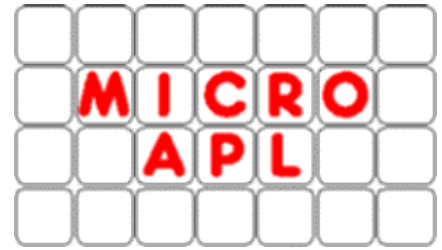
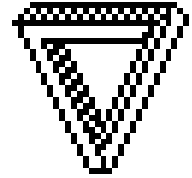
# Migrated APL2 Mainframe UI



# From APL+Win or MicroAPL APLX

Same language differences as APL2, plus:

- Many system functions & control structures not found in Dyalog APL
- Double quotes ("Don't do this!")
- More advanced Graphical User Interfaces
- Calls to external libraries



# APLX Migrations

- MicroAPL stopped developing APLX in 2016
  - Dyalog hosts a download of the last free version
- Dyalog developed migration tools in 2016
- A handful of users migrated using these tools



- Files
- master
- Go to file
- APLX.dyalog
  - Differences.md**
  - FixCovers.dyalog
  - LICENSE
  - README.md
  - ReadCovers.atf
  - TestAPLX.dyalog
  - tools.dyalog
  - xfrcode.dws
  - xfrdefs.txt
  - xfrpc.aws
  - xfrpcV5.aws

mkromberg Update comments regarding Quad-WC f2c369e · now History

Preview Code Blame 147 lines (115 loc) · 12.5 KB Raw Copy Download Edit

## Differences between APLX and Dyalog APL

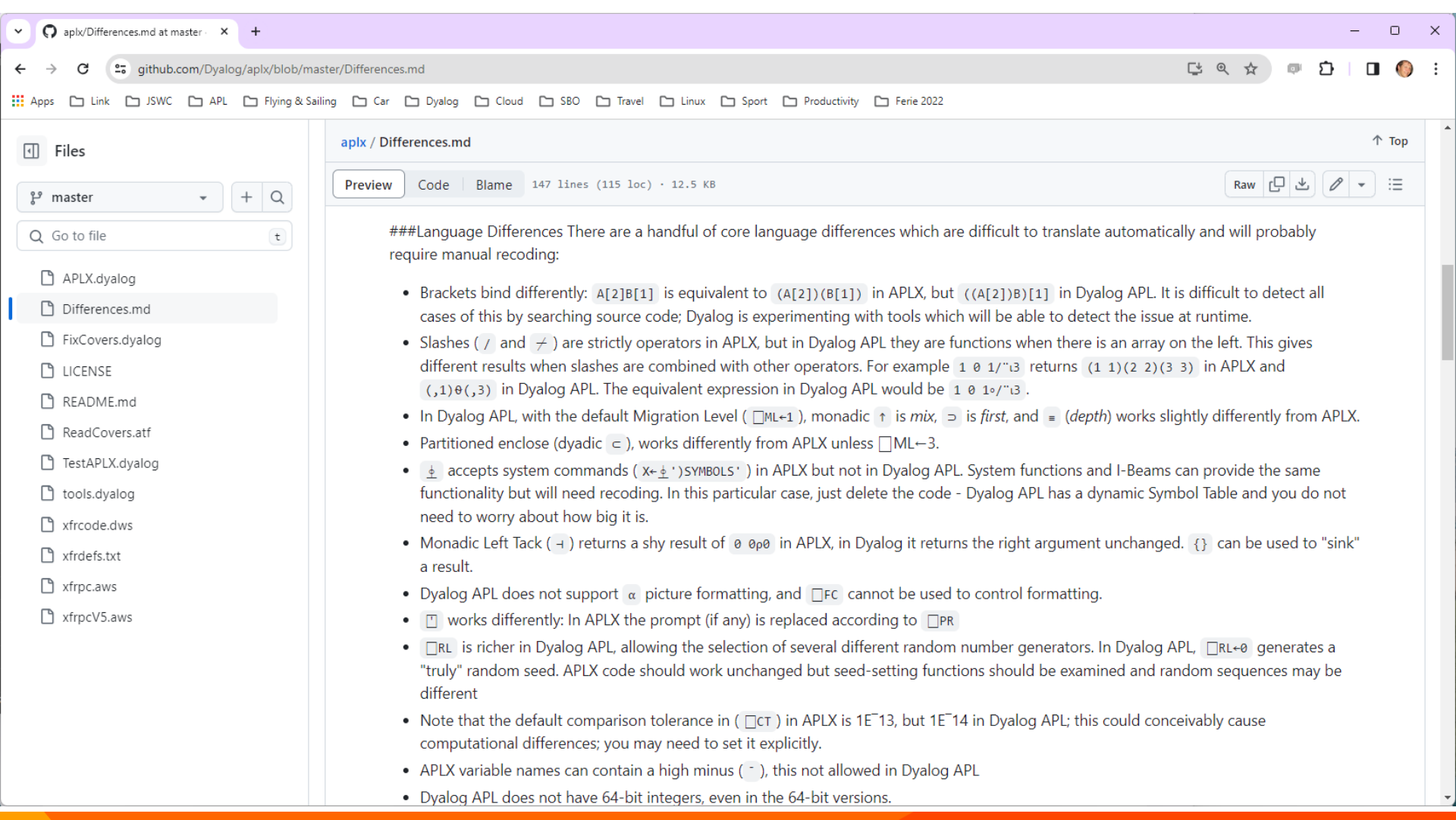
Both APL systems are variations on IBM APL2, and most computational code will work unchanged, with automatic translation, or minor manual changes. This repository contains tools to perform automated translation and provide emulations of frequently-used features that are missing from Dyalog APL.

This document lists the emulations provided, the limitations that we are currently aware of, and a discussion of differences that we are unlikely to address unless someone explains why we should. It is very much work in progress; and if you find that there are features that you desperately need, please get in touch to discuss. Contributions of enhancements or additions to the emulations, or simply failing test cases, are all very welcome.

### Important Differences

In addition to the emulated features, and language constructs which can be automatically transformed, there are a number of features of APLX which are not supported at all in Dyalog APL, and which we are not currently planning to emulate:

- The `□WI` user interface tool is not provided. Under Microsoft Windows, Dyalog APL provides a similar tool called `□WC`. An emulator for `□WI`, based on `□WC`, is available from Joachim Hoffman; an example of an application converted this tool can be found at <https://condim.at/downloads>.
  - Dyalog is developing a cross-platform emulator for `□WC`, which will work on all platforms and also run as a web server, expected to



### Language Differences There are a handful of core language differences which are difficult to translate automatically and will probably require manual recoding:

- Brackets bind differently: `A[2]B[1]` is equivalent to `(A[2])(B[1])` in APL, but `((A[2])B)[1]` in Dyalog APL. It is difficult to detect all cases of this by searching source code; Dyalog is experimenting with tools which will be able to detect the issue at runtime.
- Slashes (`/` and `⌈`) are strictly operators in APL, but in Dyalog APL they are functions when there is an array on the left. This gives different results when slashes are combined with other operators. For example `1 0 1/⌈3` returns `(1 1)(2 2)(3 3)` in APLX and `(,1)0(,3)` in Dyalog APL. The equivalent expression in Dyalog APL would be `1 0 1÷⌈3`.
- In Dyalog APL, with the default Migration Level (`⎕ML←1`), monadic `↑` is *mix*, `⊃` is *first*, and `≡` (*depth*) works slightly differently from APLX.
- Partitioned enclose (dyadic `⊆`), works differently from APLX unless `⎕ML←3`.
- `⊥` accepts system commands (`x←⊥'SYMBOLS'`) in APLX but not in Dyalog APL. System functions and I-Beams can provide the same functionality but will need recoding. In this particular case, just delete the code - Dyalog APL has a dynamic Symbol Table and you do not need to worry about how big it is.
- Monadic Left Tack (`⌊`) returns a shy result of `0 0 0` in APLX, in Dyalog it returns the right argument unchanged. `{}` can be used to "sink" a result.
- Dyalog APL does not support `⍝` picture formatting, and `⎕FC` cannot be used to control formatting.
- `⎕` works differently: In APLX the prompt (if any) is replaced according to `⎕PR`
- `⎕RL` is richer in Dyalog APL, allowing the selection of several different random number generators. In Dyalog APL, `⎕RL←0` generates a "truly" random seed. APLX code should work unchanged but seed-setting functions should be examined and random sequences may be different
- Note that the default comparison tolerance in (`⎕CT`) in APLX is  $1E^{-13}$ , but  $1E^{-14}$  in Dyalog APL; this could conceivably cause computational differences; you may need to set it explicitly.
- APLX variable names can contain a high minus (`⌊`), this not allowed in Dyalog APL
- Dyalog APL does not have 64-bit integers, even in the 64-bit versions.

### Language Differences There are a handful of core language differences which are difficult to translate automatically and will probably require manual recoding:

- Brackets bind differently: `A[2]B[1]` is equivalent to `(A[2])(B[1])` in APLX, but `((A[2])B)[1]` in Dyalog APL. It is difficult to detect all cases of this by searching source code; Dyalog is experimenting with tools which will be able to detect the issue at runtime.
- Slashes (`/` and `÷`) are strictly operators in APLX, but in Dyalog APL they are functions when there is an array on the left. This gives different results when slashes are combined with other operators. For example `1 0 1/"/13` returns `(1 1)(2 2)(3 3)` in APLX and `(,1)0(,3)` in Dyalog APL. The equivalent expression in Dyalog APL would be `1 0 1o"/13`.
- In Dyalog APL, with the default Migration Level (`⎕ML←1`), monadic `↑` is *mix*, `⊃` is *first*, and `≡` (*depth*) works slightly differently from APLX.
- Partitioned enclose (dyadic `⊂`), works differently from APLX unless `⎕ML←3`.
- `⊥` accepts system commands (`x←⊥'SYMBOLS'`) in APLX but not in Dyalog APL. System functions and I-Beams can provide the same functionality but will need recoding. In this particular case, just delete the code - Dyalog APL has a dynamic Symbol Table and you do not need to worry about how big it is.
- Monadic Left Tack (`⊖`) returns a shy result of `0 0p0` in APLX, in Dyalog it returns the right argument unchanged. `{}` can be used to "sink" a result.
- Dyalog APL does not support `α` picture formatting, and `⎕FC` cannot be used to control formatting.
- `⎕` works differently: In APLX the prompt (if any) is replaced according to `⎕PR`
- `⎕RL` is richer in Dyalog APL, allowing the selection of several different random number generators. In Dyalog APL, `⎕RL←0` generates a "truly" random seed. APLX code should work unchanged but seed-setting functions should be examined and random sequences may be different
- Note that the default comparison tolerance in (`⎕CT`) in APLX is  $1E^{-13}$ , but  $1E^{-14}$  in Dyalog APL; this could conceivably cause computational differences; you may need to set it explicitly.
- APLX variable names can contain a high minus (`¯`), this not allowed in Dyalog APL
- Dyalog APL does not have 64-bit integers, even in the 64-bit versions.

# Recent / Active APL+Win Migrations

- ◆ Two European Insurance companies
  - ◆ One with GUI, completely rewritten in Dyalog APL, the other a pure service converted to Jarvis in Linux containers
  - ◆ Handled by a European consulting partner
- ◆ METSIM® - in progress
  - ◆ Migration being handled by Dyalog
  - ◆ Will be used to develop tools to automate migration, including the Graphical User Interface
- ◆ Meeting one more potential migrant next week

# Differences which are "easy"

1 0 1 /'' 'ABC' 'DEF' 'GHI'

Convert /'' to °/''

'ABC' 'DEF' 'GHI' vs  
'ABC' 'GHI'

←Y

Convert to {}Y

Not supported in Dyalog

# Other "Easy" Differences

□ XLIB

System function not in Dyalog

```
R←ΔXLIB X
X,←' * '↓~≠X
:If 0∈ρR←↑⇒□NINFO□1←X
  :If v/'?* '∈X
    R←0 0ρ' '
  :Else
    'XFHOST ERROR FindFirstFile 1 0 3 The system cannot find the path specified.'
    □SIGNAL 22
  :EndIf
:Else
  R←R[ΔR;]
:EndIf
```

# Difficult Differences

`A B[I]`

`f.g` when `f` or `g`  
are not scalar functions

`:LeaveIf`

`A (B[I])` or  
`(A B)[I]` ?

Detect and rewrite

Enhance Interpreter

# The Hard Parts

- ◆ User Interfaces (especially Graphical)
- ◆ Component Files
- ◆ Other I/O (e.g. SQL Databases)
- ◆ External Library Calls

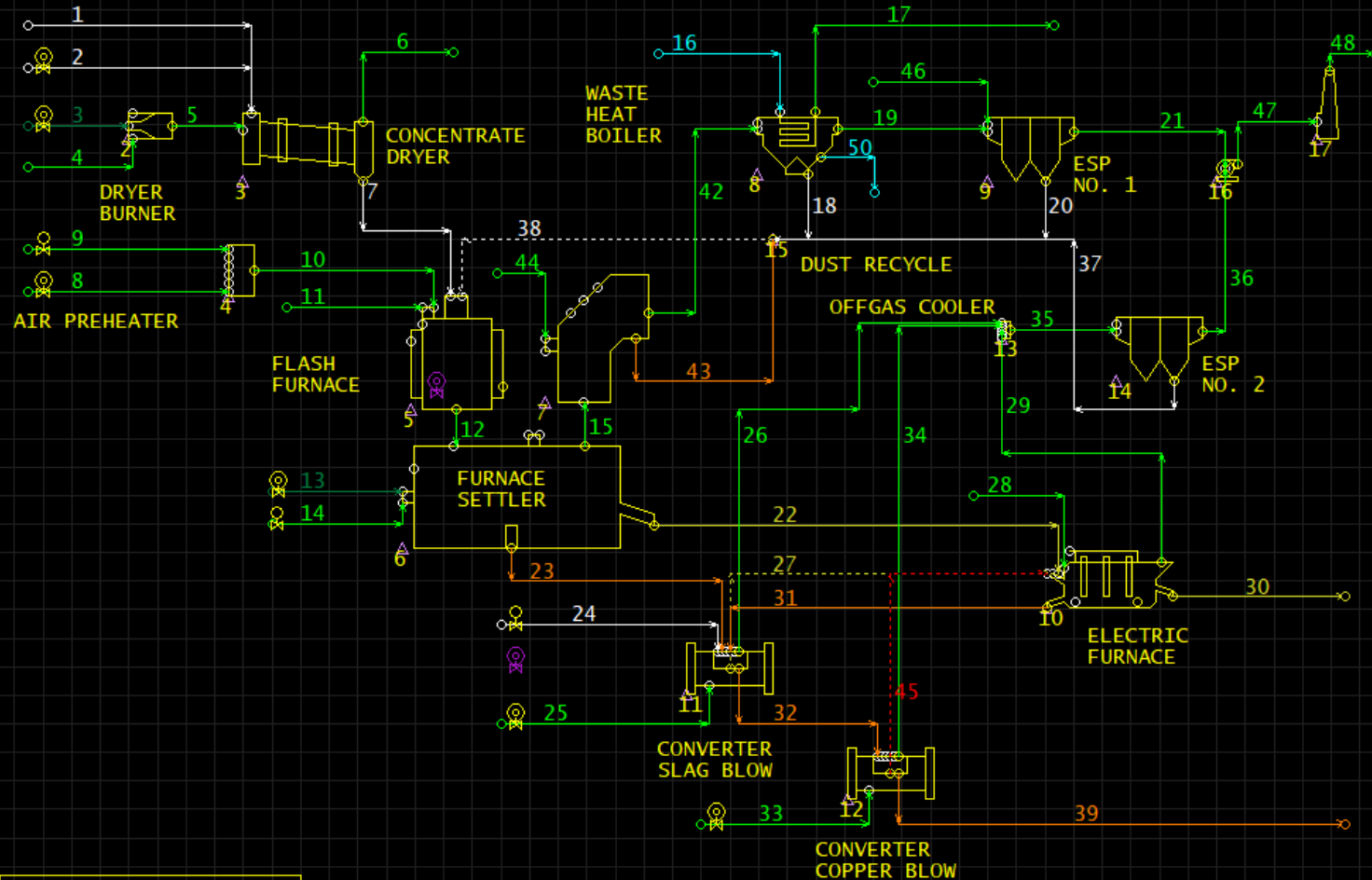


# The Elephant in the Room: OWI ?

- ◆ Austrian APL consultant Joachim Hoffman has an emulator for sale as part of consulting projects
- ◆ Dyalog is building a new emulator for the METSIM<sup>®</sup> migration
  - ◆ Our goal is that no significant changes to application code will be required
  - ◆ METSIM<sup>®</sup> screen shots follow
    - ◆ Many thanks to Alex Holtzapple, CEO of MSI

# FLASH SMELTER EXAMPLE

Stream Number



Stream 322

Output Level: 0 Design Factor: 0 Maximum Flow: 0  
 Box Number: 0 Variables 1 2 3

322 SI LI  
 IR Slurry Label SO GC OK Cancel

	MT/DY		Wt. Frac.	gpl	MT/DY		Wt. Frac.	gpl	MT/DY
SOLIDS	2819.6251	aH2O	0.8245512	977.40919	17055.895	H	1	0.0922685	109.37361
SLD-ORG	0	aH2SO4	0.0000101	0.012	0.2094012	C	6	0	0
AQUEOUS	20685.064	aH2CO3	0	0	0	N	7	0	0
ORGANIC	0	aNiSO4	0.0119156	14.124588	246.47556	O	8	0.8235821	976.26044
MOLTEN	0	aCoSO4	0.0004037	0.4786549	8.3525794	Na	11	0.0000077	0.0091656
MAITE	0	aCo2(SO4)	0	0	0	Mg	12	0.0309770	36.719704
SLAG	0	aFeSO4	0.0000014	0.00167	0.0291416	Al	13	0.0002659	0.3152151
GAS	0	aFe2SO43	0.0002499	0.2962831	5.1701727	Si	14	0	0
TOTAL	23504.689	aAl2SO43	0.0016860	1.9985948	34.875693	S	16	0.0457426	54.222563
% SOLID	0.1199601	aCa(OH)2	0	0	0	Cl	17	0	0
Contrl C	0	aCaSO4	0.0015753	1.8674523	32.587241	Ca	20	0.0004637	0.5497767
Temp C	25	aCr2SO43	0.0000057	0.0067845	0.1183903	Sc	21	0.0006129	0.7265744
Temp F	77	aCuSO4	0.0004087	0.4845496	8.4554426	Cr	24	0.0000015	0.0017990
Pres kPa	101.325	aMgSO4	0.1533737	181.80664	3172.5452	Mn	25	0.0011638	1.3795873
Pres kPag	0	aMnSO4	0.0031988	3.7918587	66.168338	Fe	26	0.0000703	0.0833716
Pres psia	14.695949	aNaCl	0	0	0	Co	27	0.0001535	0.1819971
Pres psig	0	aNa2CO3	0	0	0	Ni	28	0.0045199	5.3579245
Time	1	aNa2SO4	0.0000238	0.0283146	0.4940931	Cu	29	0.0001627	0.1929071
Gal/min	3408.8166	aNaOH	0	0	0	Zn	30	0.0000072	0.0086240
L/sec	215.06279	aSc2(SO4)	0.0025775	3.0553814	53.316732				
L/min	12903.767	aZnSO4	0.0000179	0.0212951	0.3716020				
M3/hr	774.22605	aSO4-	0	0	0				
NM3/hr	772.28958	aNH3	0	0	0				

# Status

- ◆ We just started reworking and enhancing the APLX tools
- ◆ Will enumerate differences between APL+Win and Dyalog
- ◆ Will create emulation functions as required
  
- ◆ We \*may\* also decide to add new features to Dyalog v20 (which should start user testing in late 2024)
  - ◆ For example :LeaveIf

# Status

- ◆ Dyalog has been contracted to port the METSIM<sup>®</sup> application
- ◆ Hired one new APL developer, thinking about another
- ◆ We will have ~1.5-2 full time equivalent resources working on migration tools until further notice
  
- ◆ **All the resulting tools and documentation will be free and open source**

# Limbering Up

- ◆ APL+WIN comes with a handful of GUI demonstration applications

My copy of APL+Win is a little dated...

```
)load Examples\DEMODRAW
C:\APLWIN11\EXAMPLES\DEMODRAW SAVED 12. september 2010
This DEMODRAW workspace is distributed with the APL+Win.
It contains a demonstration of the Draw method.
Run:

    DemoDraw

Copyright 2005-2006, APLNow LLC.
    DemoDraw
```

DemoDraw -- rect

File Demo Help!

- Bitmap
- Circle
- Howdy
- Icon
- Poly**
- Rect
- Text

## Now export the APL+Win code to APL Transfer Format

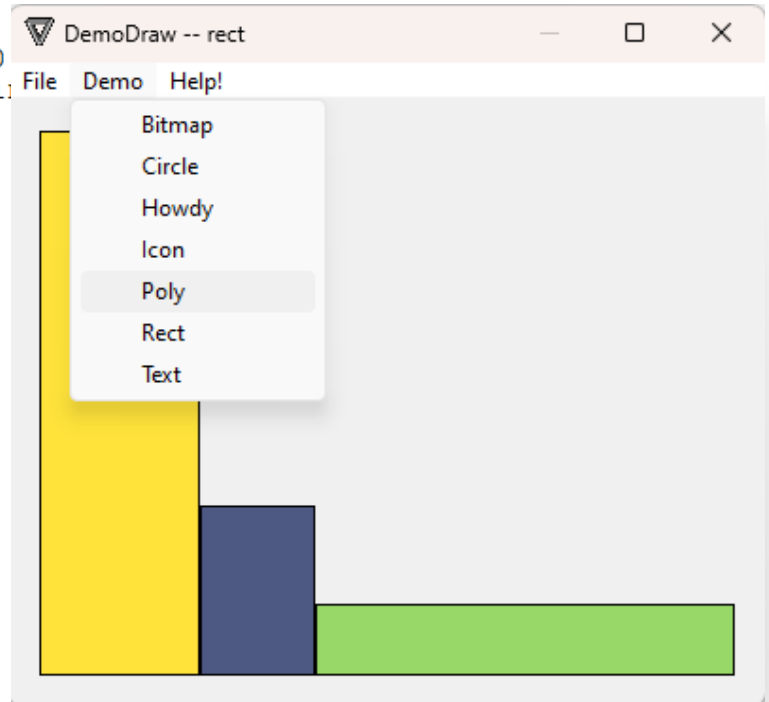
```
)load Examples\DEMODRAW  
C:\APLWIN11\EXAMPLES\DEMODRAW SAVED 12. september 2010  
This DEMODRAW workspace is distributed with the APL+Win.  
It contains a demonstration of the Draw method.  
Run:
```

```
DemoDraw
```

```
Copyright 2005-2006, APLNow LLC.
```

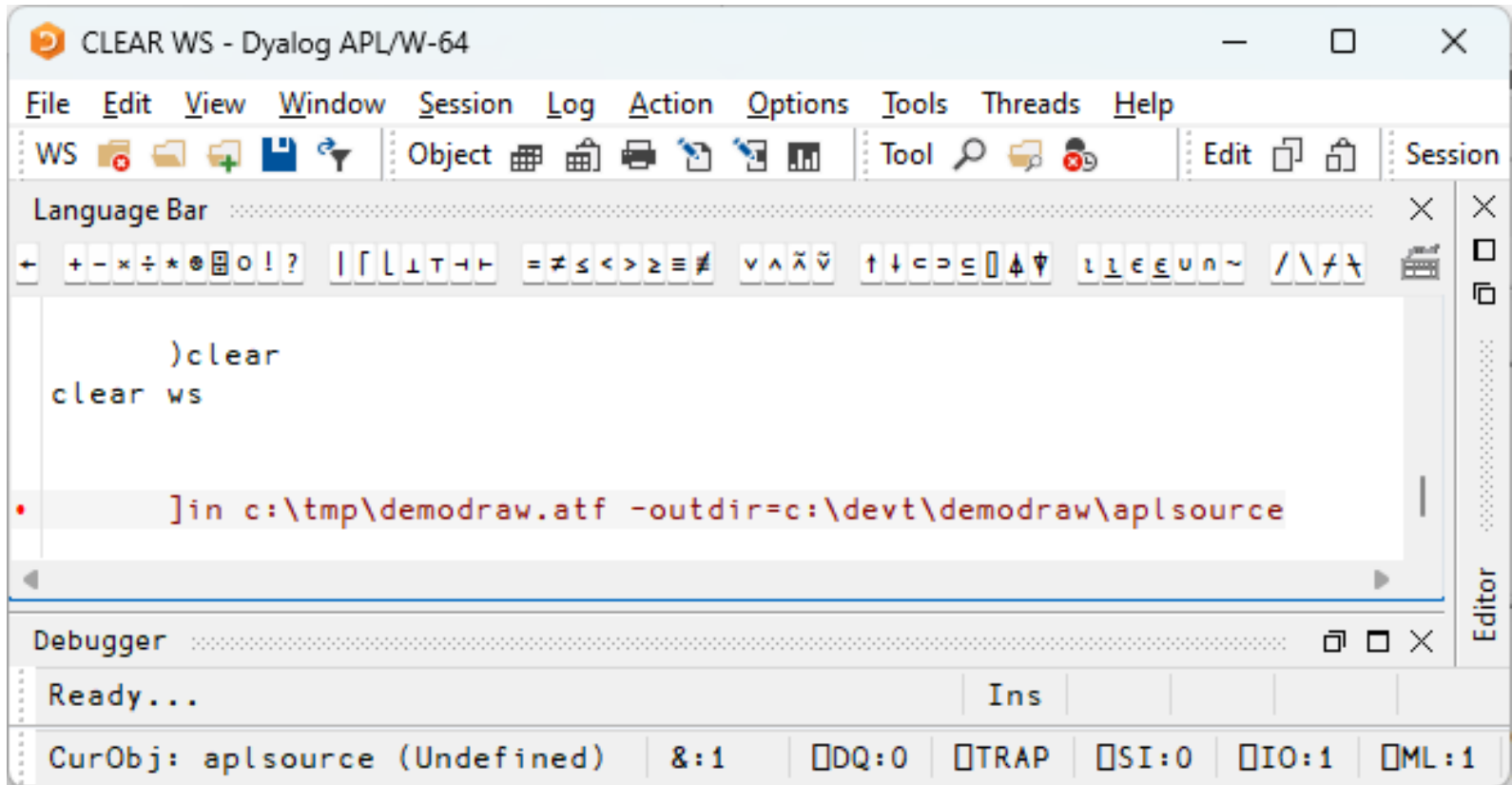
```
DemoDraw
```

```
]out c:\tmp\demodraw.atf|
```





# Convert the ATF file to Dyalog Source files



The screenshot shows the CLEAR WS - Dyalog APL/W-64 application window. The menu bar includes File, Edit, View, Window, Session, Log, Action, Options, Tools, Threads, and Help. The toolbar contains icons for file operations and editing. The Language Bar is visible above the editor. The editor contains the following APL code:

```
)clear  
clear ws  
  
]in c:\tmp\demodraw.atf -outdir=c:\devt\demodraw\aplsource
```

The status bar at the bottom displays "Ready..." and "CurObj: aplsource (Undefined)".

c:\tmp\demodraw.dws - Dyalog APL/W-64

File Edit View Window Session Log Action Options Tools Threads Help

WS Object Tool Edit Session APL385 Unicode 16

Language Bar

```

j in c:\tmp\demodraw.atf -outdir=c:\devt\demodraw\aplsource
Linked: □SE.input.c.inout.AtIn ↔ C:\Devt\AtfIn\APLSOURCE\AtIn
PROCESSING "c:\tmp\demodraw.atf" (9266 bytes):
  Variable: c:\devt\demodraw\aplsource/□PP.apla
  Variable: c:\devt\demodraw\aplsource/□IO.apla
  Variable: c:\devt\demodraw\aplsource/□CT.apla
  Variable: c:\devt\demodraw\aplsource/□RL.apla
  Variable: □PR *** FAILED ***
  Variable: □LX *** FAILED ***
  Function: c:\devt\demodraw\aplsource/DemoDraw.aplf
  Variable: c:\devt\demodraw\aplsource/Describe.apla
  Function: c:\devt\demodraw\aplsource/IsForm.aplf
  Function: c:\devt\demodraw\aplsource/Wfree.aplf
  Function: c:\devt\demodraw\aplsource/fmDrawDemoClearH_Click.aplf
  Function: c:\devt\demodraw\aplsource/fmDrawDemoClear_Click.aplf
  Function: c:\devt\demodraw\aplsource/fmDrawDemoPrint_Click.aplf
  Function: c:\devt\demodraw\aplsource/fmDrawDemoReplay_Click.aplf
  Function: c:\devt\demodraw\aplsource/fmDrawHelp_Click.aplf
  Function: c:\devt\demodraw\aplsource/fmDraw_Make.aplf
  Function: c:\devt\demodraw\aplsource/fmDraw_MouseDown.aplf
  Function: c:\devt\demodraw\aplsource/fmDraw_Paint.aplf
  Function: c:\devt\demodraw\aplsource/fmDraw_Show.aplf
  Function: c:\devt\demodraw\aplsource/fmDrawΔ.aplf
  Function: c:\devt\demodraw\aplsource/fmDrawΔDemo.aplf
SUCCESS:
  □SAVE 'c:\tmp\demodraw' A to save as Dyalog Workspace
  †QuadThings A to see the systems variables modified
  †PROBLEMS A to see unfixed objects
  |

```

Debugger Ready... Ins

CurObj: tmp (Undefined) 8:1 □DQ:0 □TRAP □SI:0 □IO:1 □ML:1

Editor



Name	Date modified	Type	Size
□CT.apla	06-04-2024 13:35	APLA File	1 KB
□IO.apla	06-04-2024 13:35	APLA File	1 KB
□PP.apla	06-04-2024 13:35	APLA File	1 KB
□RL.apla	06-04-2024 13:35	APLA File	7 KB
📄 DemoDraw.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 Describe.apla	06-04-2024 13:35	APLA File	1 KB
📄 fmDraw_Make.aplf	06-04-2024 13:35	Dyalog APL Source	2 KB
📄 fmDraw_MouseDown.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDraw_Paint.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDraw_Show.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDrawΔ.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDrawΔDemo.aplf	06-04-2024 13:35	Dyalog APL Source	2 KB
📄 fmDrawDemoClear_Click.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDrawDemoClearH_Click.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDrawDemoPrint_Click.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDrawDemoReplay_Click.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 fmDrawHelp_Click.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 IsForm.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB
📄 Wfree.aplf	06-04-2024 13:35	Dyalog APL Source	1 KB

```
DemoDraw;R;WinDir;fmDrawΔdemo;fmDrawΔfonts;fmDrawΔhist;fm
[ a▽DemoDraw -- Run the Draw demo
  a Build the form it doesn't already exist
  :if ~IsForm 'fmDraw'
    fmDraw_Make
  :end

  a Wait on the form
  R←'fmDraw' □wi 'Wait'
```

**NB this is original APL+Win Source**

For METSIM®, we plan to update the APL+Win environment to run off text files too.

# APL Source in Text Files

- Text files are now the recommended vehicle for Dyalog APL source code
- Workspaces will continue to be supported "forever"
  - For distribution
  - For crash dump analysis
  - For old timers who refuse to change 😊

Extension	Content
.apla	Array
.aplc	Class
.aplf	Function
.apln	Namespace
.aplo	Operator

# Benefits of Text Source

- **New developers will feel at home immediately**
- Apply mainstream Source Code management tools
  - Git, Subversion, Mercurial, ...
- Also use 3<sup>rd</sup> party frameworks for Continuous Integration, Testing
- **Your auditors will be pleased**

# Update Setup.aplf

Browse files

Addresses [issue #2](#)

main (#9)

aplteam committed on Jan 15 Verified

1 parent 289f9a5 commit 71a3268

Showing 1 changed file with 2 additions and 1 deletion.

Whitespace Ignore whitespace Split Unified

```

APLSource/NuGet/Setup.aplf
@@ -2,7 +2,8 @@
2      2      :If 0=NC 'force' ⋄ force←0 ⋄ :EndIf
3      3      project_name←{(1-[/(φω)ι'\'])τω}project_dir
4      4
5      -      :If 'net4'≡4↑(TFM RID)+GetDotNetIDS ⍉
6      +      (TFM RID)+GetDotNetIDS ⍉
7      +      :If 'net4'≡4↑TFM
8      7      ('dotnet ',TFM,' currently not supported')⊞SIGNAL 11
9      8      :EndIf
10     9      A Ask dotnet to create a .csproj file

```

## Next, map code from APL+Win to Dyalog

```
]todyalog aplsource c:\devt\demodraw\dyalog a2k  
Using c:\devt\demodraw\aplsource\atfmap.txt  
20 files processed
```

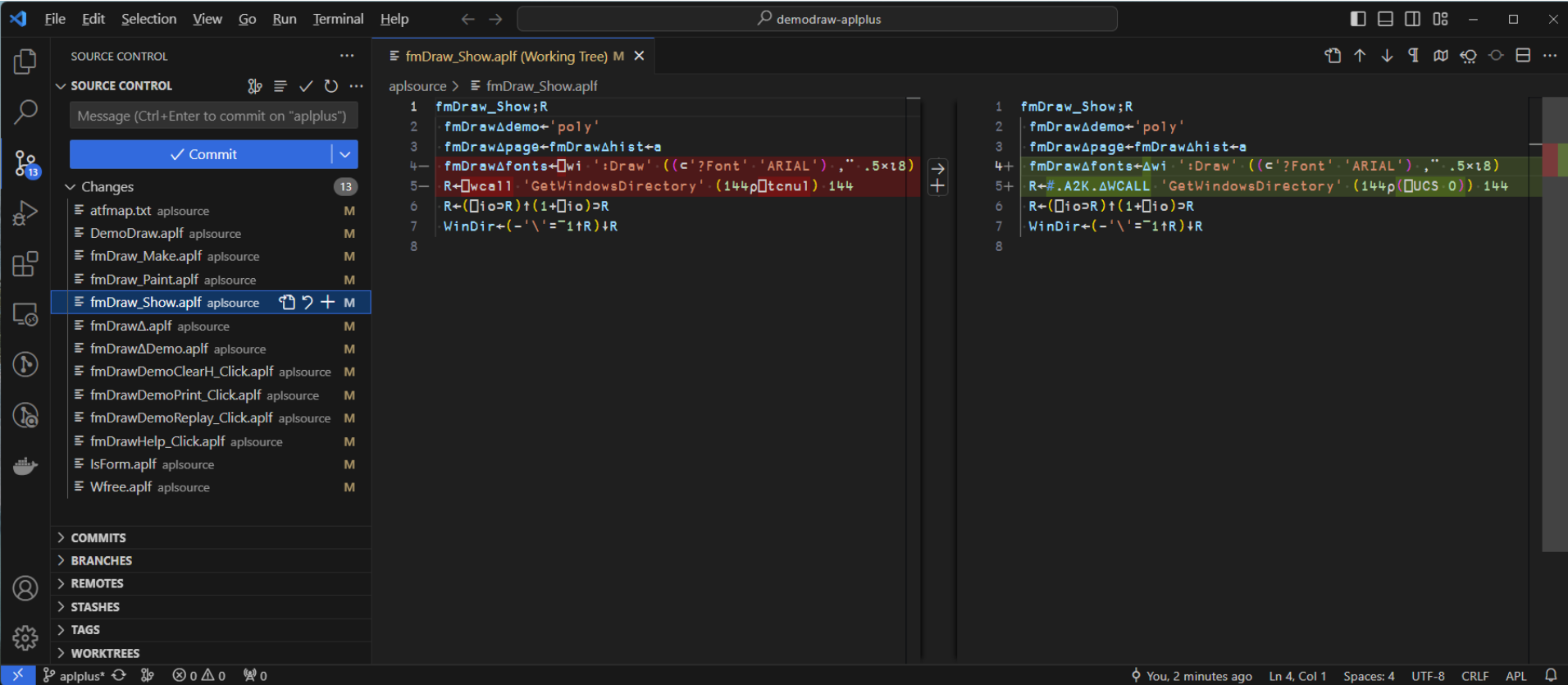
```

:catch%:else
:catchall%:else
:endtry%:endtrap
:returnif%→0/↵
:try *%:trap 0
:try%:trap 0
;□ALX%;ΔQALX
;□ELX%;ΔQELX
;□SA%;ΔQSA
;□WSELF%;ΔWSELF
□ALX%ΔQALX
□ALX←%#.A2K.ΔSetALX
□AV%#.A2K.ΔAV
□CHDIR%#.A2K.ΔCHDIR
□CHDIR%#.A2K.ΔCHDIR
□CN*%□N
□CRLF%(□UCS 13 10)
□CURSOR%#.A2K.ΔCURSOR
□DEF%□FX
□DR%#.A2K.ΔDR
□ELX%ΔQELX
□ENLIST%{□ml←1◊εω}
□FSTIE%#.A2K.ΔFSTIE
□FTIE%#.A2K.ΔFTIE
□HTOPIC%#.A2K.ΔHTOPIC
□IDLIST%#.A2K.ΔIDLIST
□IDLOC%#.A2K.ΔIDLOC
□INT%#.A2K.ΔINT
□KEYLOG%#.A2K.ΔKEYLOG
□KEYW%#.A2K.ΔKEYW
□LIB%#.A2K.ΔLIB
□LIBD%#.A2K.ΔLIBD
□LIBS%#.A2K.ΔLIBS
□LOG%#.A2K.ΔLOG
□MF%□MONITOR
□MIX%#.A2K.ΔMIX
□NA%#.A2K.ΔNA
□PEEK%#.A2K.ΔPEEK
□PENCLOSE%ε
□PFKEYS%#.A2K.ΔPFKEYS
□POKE%#.A2K.ΔPOKE
□POKES%#.A2K.ΔPOKES
□REPL%/
□SA%ΔQSA
□TCBEL%(□UCS 7)
□TCBS%(□UCS 8)
□TCESC%(□UCS 27)
□TCFF%(□UCS 12)
□TCHT%(□UCS 9)
□TCLF%(□UCS 10)
□TCNL%(□UCS 13)
□TCNUL%(□UCS 0)
□TYPE%#.A2K.ΔTYPE
□UCMD%#.A2K.ΔUCMD
□UCS%#.A2K.ΔUCS
□USERID%□AN
□VI%#.A2K.ΔVI
□WCALL%#.A2K.ΔWCALL
□WGIVE%#.A2K.ΔWGIVE
□WI%#.A2K.ΔWI
□WIN%#.A2K.ΔWIN
□WINDOW%#.A2K.ΔWINDOW
□WKEYS%#.A2K.ΔWKEYS
□WSELF%ΔWSELF
□WSSIZE%(2000□0)
□XFDUP%#.A2K.ΔXFDUP

```



# Take advantage of Git and VS Code



The screenshot displays the Visual Studio Code interface with the Git extension. The left sidebar shows the SOURCE CONTROL view with a list of files in the 'Working Tree'. The main editor area shows a diff view for the file 'fmDraw\_Show.aplf'. The diff highlights changes between the current working tree and the last commit. The code in the diff is as follows:

```
1 fmDraw_Show;R
2 fmDrawΔdemo+'poly'
3 fmDrawΔpage+fmDrawΔhist+a
4- fmDrawΔfonts+□wi ':Draw' ((c'?Font' 'ARIAL')',','.5x18)
5- R←□wcall 'GetWindowsDirectory' (144p□tcnu1) 144
6 R←(□io>R)†(1+□io)>R
7 WinDir+(-'\ '=™1†R)†R
8
```

The status bar at the bottom indicates the current file is 'aplfus\*', the workspace is 'aplfus\*', and the encoding is 'UTF-8'.

Original  
APL+Win code

```
fmDraw_Show.aplf (Working Tree) M X
aplsourc > fmDraw_Show.aplf
1 fmDraw_Show;R
2   fmDrawΔdemo←'poly'
3   fmDrawΔpage←fmDrawΔhist←a
4-  fmDrawΔfonts←wi·':Draw'·((c'?Font'·'ARIAL')·,·'.5×18)
5-  R←wcall·'GetWindowsDirectory'·(144p[tcnu1]·144
6   R←(io>R)↑(1+io)⊃R
7   WinDir←(-'\ '=~1↑R)↓R
8
```

Converted to  
Dyalog

```
1 fmDraw_Show;R
2   fmDrawΔdemo←'poly'
3   fmDrawΔpage←fmDrawΔhist←a
4+  fmDrawΔfonts←Δwi·':Draw'·((c'?Font'·'ARIAL')·,·'.5×18)
5+  R←#.A2K.ΔWCALL·'GetWindowsDirectory'·(144p[UCS·0])·144
6   R←(io>R)↑(1+io)⊃R
7   WinDir←(-'\ '=~1↑R)↓R
8
```

So what is #.A2K.ΔWCALL?

devt > Atfln > APLSource > xfr > U > A2K > Search A2K

Sort View Preview

APLPΔSYS.aplc	ΔIDLOC.aplf	ΔTYPE.aplf	<pre>r←ΔWCALL args :Select 1&gt;,⊆args :Case 'W_Init'     r←'' (⌘1+2 ⌘NQ '.' 'GetCommandLineArgs') :Else     'This WCALL not currently supported' ⌘SIGNAL 11 :EndSelect</pre>
bigstring	ΔINT.aplf	ΔVI.aplf	
ΔAV.apla	ΔLIB.aplf	ΔWCALL.aplf	
ΔCHDIR.aplf	ΔLIBD.aplf	ΔWGIVE.aplf	
ΔCOPY.aplf	ΔLIBS.aplf	ΔWSELF.apla	
ΔDR.aplf	ΔLOG.aplf	ΔXFDUP.aplf	
ΔENLIST.aplf	ΔMIX.aplf	ΔXLIB.aplf	
ΔERASE.aplf	ΔNA.aplf	⌘IO.apla	
ΔESC.aplf	ΔNUL.aplf	⌘ML.apla	
ΔFI.aplf	ΔPCOPY.aplf	GLOBALΔTRAP.apla	
ΔFIRST.aplf	ΔPEEK.aplf	GLOBALTRAP.aplf	
ΔFSTIE.aplf	ΔPOKE.aplf		
ΔFTIE.aplf	ΔSEG.aplf		
ΔHT.aplf	ΔSPLIT.aplf		
ΔIDLIST.aplf	ΔTCBEL.aplf		

So what is #.A2K.ΔWCALL ?

```
r←ΔWCALL args
```

```
□ :Select 1▷,⊆args
```

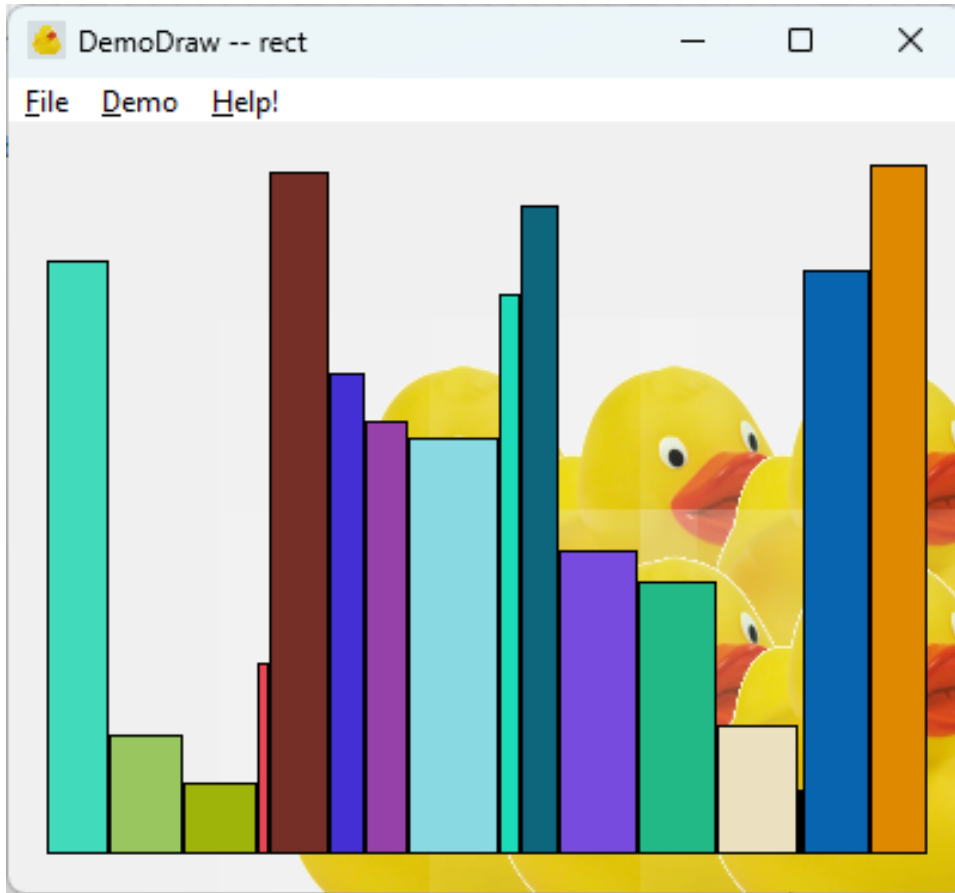
```
  :Case 'W_Init'
```

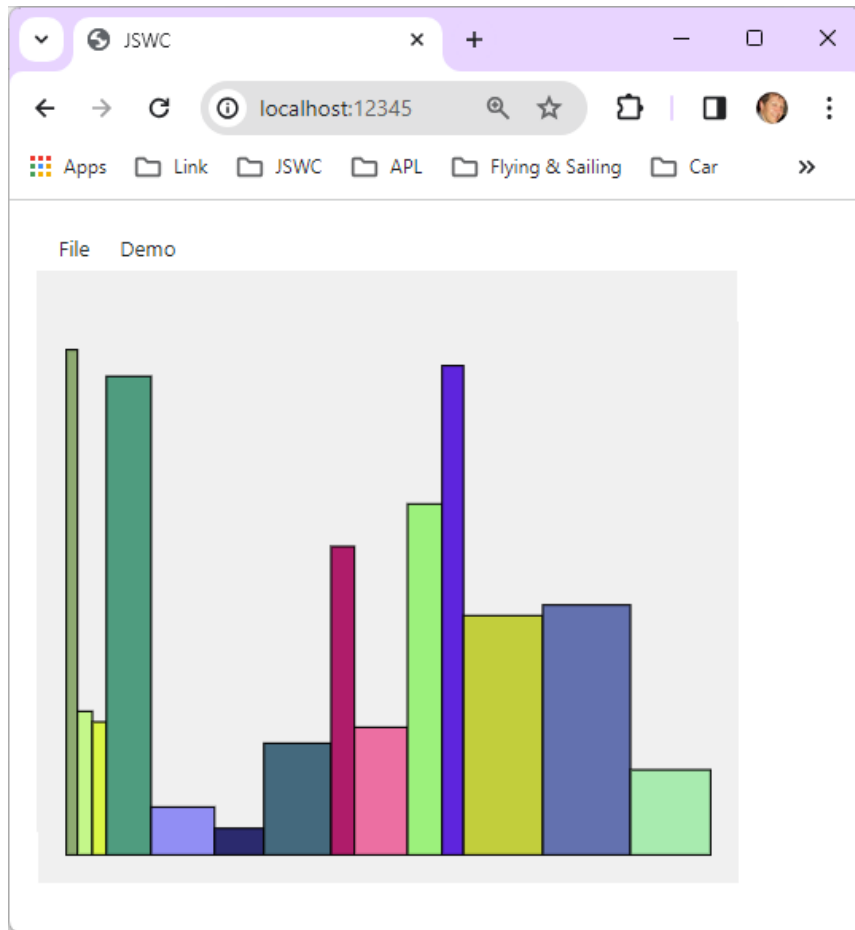
```
    r←'' '' (⌘1↓2 □NQ '.' 'GetCommandLineArgs')
```

```
  :Else
```

```
    'This WCALL not currently supported' □SIGNAL 11
```

```
  :EndSelect
```





# "DeltaWi" status

- We started a few weeks ago, while also training a new APL recruit
- Priorities will be driven by METSIM<sup>®</sup> migration
  - (and any samples that you send us to help us prepare)

# Component Files

- ◆ We are planning to develop an APL+Win COM server that will allow the use of APL+Win component files from Dyalog APL
- ◆ Avoid the need for "big bang" data migrations
  - ◆ Component files can be migrated over time
- ◆ You may need a runtime license for APL+Win



# Planning a Migration

- ◆ 1:1 GUI emulation of APL+Win GUI?
  - ◆ Grid components may demand some recoding
- ◆ Service Orientation
- ◆ Cloud Deployment
- ◆ Web-based UI
- ◆ 64-bit?
- ◆ Unicode?

# Preparing for a Migration

- Will you need to run the original and migrated versions in parallel?
  - In most cases, this *\*will\** be necessary for some weeks, months – or years
- Will the original code continue to change during the period?
  - Consider a compatibility layer to insulate code from platform differences
  - Consider using Git to merge changes from old to new platform

# Preparing for a Migration

- ◆ While you wait for the migration to start, write regression tests (if you don't already have them)
- ◆ The value of this cannot be overstated
- ◆ Regression tests will be useful before the migration, of course 😊

# Let us know your plans!

- ◆ We're actively working on a migration now
- ◆ Knowledge about other potential migrations will help us get ready for yours

