

DYALOC

Glasgow 2024

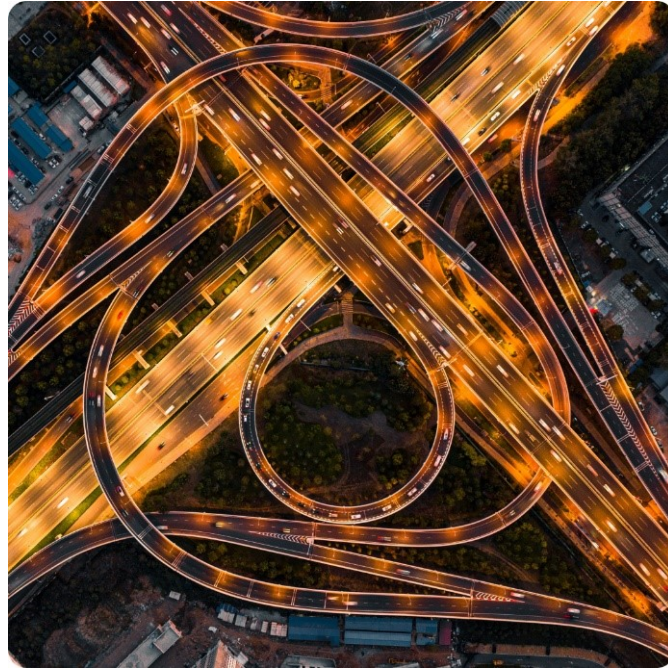
The Road Ahead



Morten Kromberg



Road Map - 2023

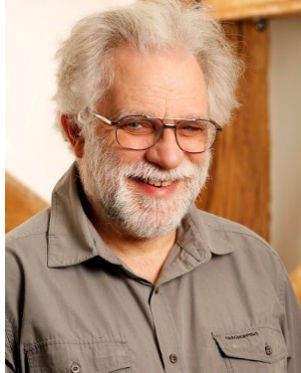


Road Map - 2024



Retirees

- Geoff
- Gitte
- Pete



U17 Weds 15:15: Let's Put the Future Behind Us (Panel Discussion)

Dyalog – The Next Generation

2010-2021



2022



2023



U18 Weds 15:55:
The New Breed Plugs In
(Panel Discussion)

Dyalog – The Next Generation



2024 Summer Interns

U03 Mon 15:15: raylib-apl (Brian Ellingsgaard)

U07 Tue 15:15: Climbing Trees and Catching Bugs (Asher Harvey-Smith)





Karl Holt

(February, Århus, DK)

- ◆ Karl is a member of the APL Tools Group and an APL Consultant
- ◆ Karl is working on the emulator for APL+Win GUI (GUI)

D09 Tue 14:25: Migrating APL+Win Applications



Brandon Wilson

(July, 昭和村 / Shōwa, Japan)

- ◆ Brandon is heading up the Static Analysis project
- ◆ Working with Aaron Hsu to enhance the co-dfns compiler to support the project

D06 Mon 13:55: Static Analysis of APL in APL

D14 Wed 09:35: Data Parallel Proof Verification in APL



Martina Crippa

(August, Copenhagen)

- ◆ Martina is a member of the APL Tools group and a Consultant
- ◆ Martina is learning APL and knows C++
- ◆ Her first project is to build a prototype of a Kafka interface for a client



Neil Kirsopp

(Gunzenhausen, Bavaria)

- ◆ Neil is a JavaScript developer (amongst many things)
- ◆ Also an APL enthusiast, had already signed up for Dyalog'24 before we met
- ◆ He will work on enhancing EWC, and also take over RIDE development
- ◆ ... and look at a VS Code / Emacs plugin

Highlights of Version 19.0 (Q1'24)

See Last Year's Presentations

Platform Support / Distribution

- 64-bit ARM support
 - New Macs
- Enhanced .NET Bridge
 - Framework vs new .NET versions
- Bound executables on all platforms

Building Production Systems

- Token range reservation
- WS FULL handling
- NCPY/□NMOVE callbacks

Developer Productivity / IDE

- Source "as typed" by default
- Multi-line input on by default
- HTMLRenderer updates
- Link 4.0: Config files, simple text arrays
- HttpCommand client, Jarvis web service

Installing & Managing APL

- Multiple session files
- Health Monitor

Big Things Heading Your Way

- ◆ Array Notation
- ◆ Set & Get Variables
- ◆ Token-by-Token Debugging
- ◆ Everywhere WC
- ◆ Reverse Compose



Array Notation

```
('Three'  
'Blind'  
'Mice')
```

≡

```
z ← , c 'Three'  
z , ← 'Blind'  
z , ← 'Mice'
```

```
[0 6 1 8  
1 4 1 4  
2 7 1 8  
3 1 4 2]
```

≡

```
z ← 0, 6 1 8  
z , ← 1 4 1 4  
z , ← 2 7 1 8  
z , ← 3 1 4 2
```

```
[10  
20  
30  
40]
```

≡

```
z ← 10  
z , ← 20  
z , ← 30  
z , ← 40
```


Namespace Notation

```
person ← (first: 'Max' ◊ last: 'Mustermann')
```

```
person.last, ', ', person.first  
Mustermann, Max
```

```
person.(first, ' ', last)  
Max Mustermann
```

D03 Mon 10:10: Array Notation: A Journey of Discovery (John Daintree)

Set and Get Variables

Get:

```
[source] □VGET 'Name' ('Height' ~1)  # Values w/optional defaults  
merged←□NS defaults inputs          # Merge namespaces
```

Name List with Values:

```
(names values)←[source] □VGET 2      # Name Matrix and values  
pairs←[source] □VGET ~2              # (Name Value) Pairs
```

Set:

```
ref←[target] □VSET ('First' 'Lieschen')('Last' 'Müller')  
ref←[target] □VSET (2 5p'FirstLast ')( 'Max' 'Munstermann')
```


Token by Token Debugging

- Screen shot on next page from Max Sun, (BCA Research)
- Max has his own fork of RIDE with VIM bindings and editor extensions:
<https://github.com/MaxCan-Code/ride/tree/master>

Left - C:\Users\max.sun\apl\dev\APLSource\ - Dyalog APL/W-64

1
2
3
4

returned:
1 2 3 4 5 6 7 8 9 10

Left Argument

Middle - C:\Users\max.sun\apl\dev\APLSource\ - Dyalog APL/W-64

1 (⊖⊖⊖,⊖)'tryapl.org'

Expression

Function - C:\Users\max.sun\apl\dev\APLSource\ - Dyalog APL/W-64

1
2

returned:
1 2 3 4 5 6 7 8 9 10

Previous Result

Right - C:\Users\max.sun\apl\dev\APLSource\ - Dyalog APL/W-64

1
2
3
4
5
6
7
8
9
10
11

returned:
tryapl.org

Right Argument

PLLeft - C:\Users\max.sun\apl\dev\APLSource\ - Dyalog APL/W-64

1 <no value>

Previous Left

PRight - C:\Users\max.sun\apl\dev\APLSource\ - Dyalog APL/W-64

1 tryapl.org

Previous Right

PFFunction - C:\Users\max.sun\apl\dev\APLSource\ - Dyalog APL/W-64

1
2

returned:
1 2 3 4 5 6 7 8 9 10

Previous Function

(this is using the "regular" https://github.com/Dyalog/ride/tree/tbt_support)

See John Daintree's presentation from Dyalog'23

EWC – "Everywhere" WC

- A JavaScript emulation of Dyalog's Win32 wrappers (□WC, □WG, □WS ...)
- "Everywhere" means
 - Windows, Linux, macOS Desktops
 - Dyalog APL on any platform acting as a Web Server

D07 Mon 15:45: Everywhere WC (Morten Kromberg)

WC ↓

EWC →

Function Table

File Colours

Name	Gender	Score	Expert
Amir	Male	12	<input type="checkbox"/>
Fatima	Female	13	<input checked="" type="checkbox"/>

Average Score: 12.5

- Q1
- Q2
 - Apr
 - May
 - Jun

10 x [dropdown] Calc [input type="text" value="*****"]

	A	B	C	D	E	F	G
1	1	2	3	4	5	6	
2	2	4	6	8	10	12	
3	3	6	9	12	15	18	
4	4	8	12	16	20	24	
5	5	10	15	20	25	30	
6	6	12	18	24	30	36	
7	7	14	21	28	35	42	
8	8	16	24	32	40	48	
9	9	18	27	36	45	54	
10	10	20	30	40	50	60	

EWC

localhost:22322

File Colours

Name	Gender	Score	Expert
Amir	Male	12	<input type="checkbox"/>
Fatima	Female	13	<input checked="" type="checkbox"/>

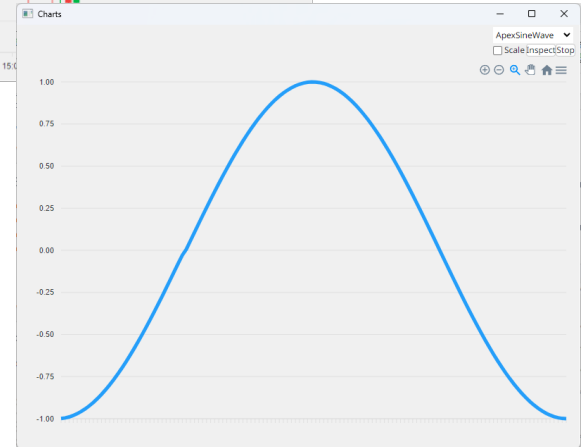
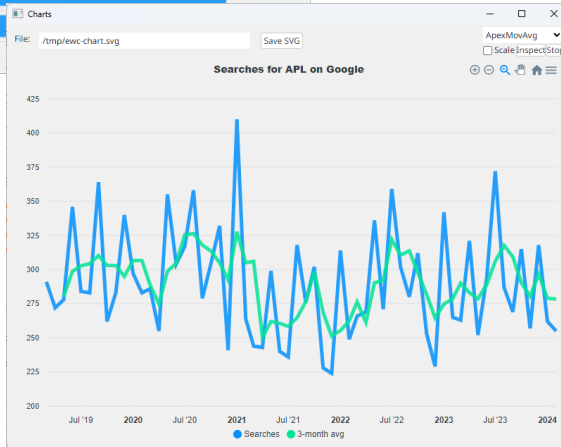
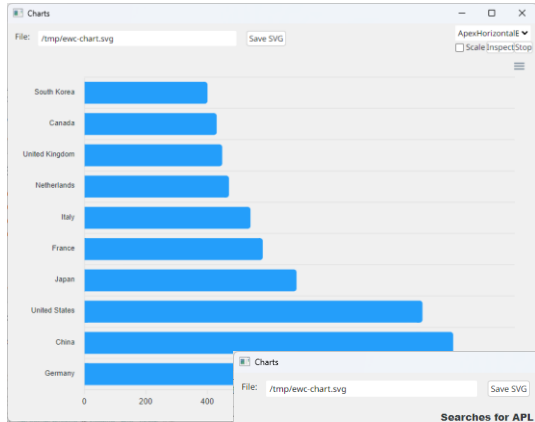
Average Score: 12.5

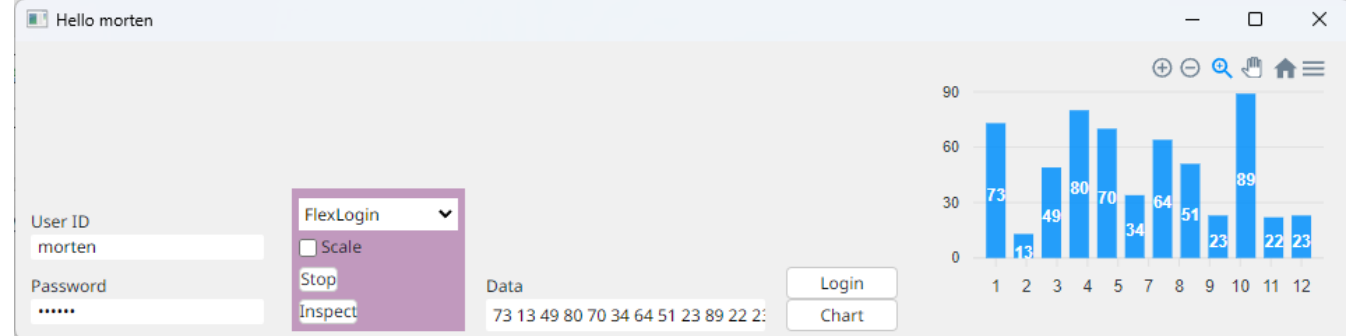
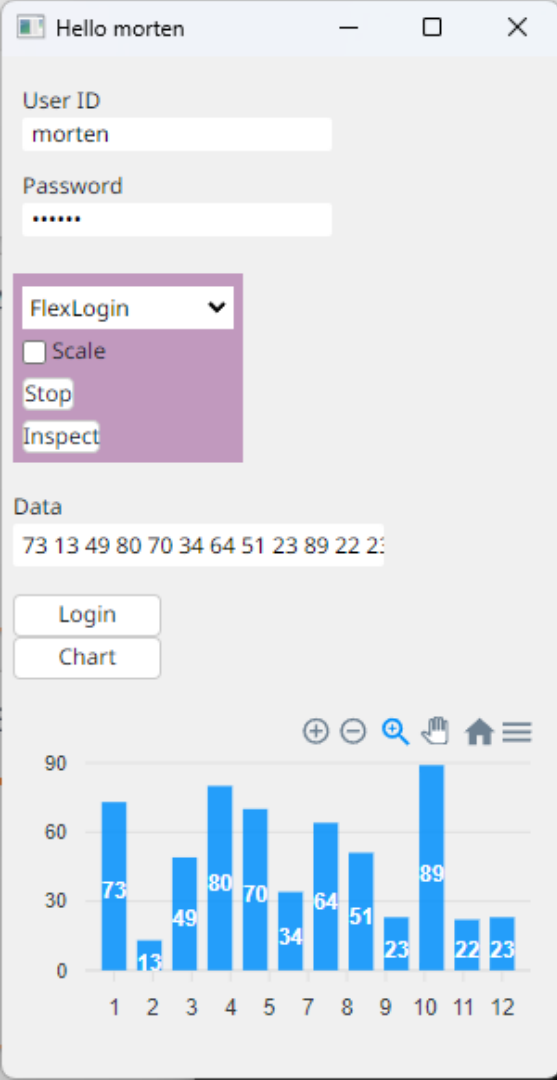
- Q1
- Q2
 - Apr
 - May
 - Jun

10 x [dropdown] Calc [input type="text" value="*****"]

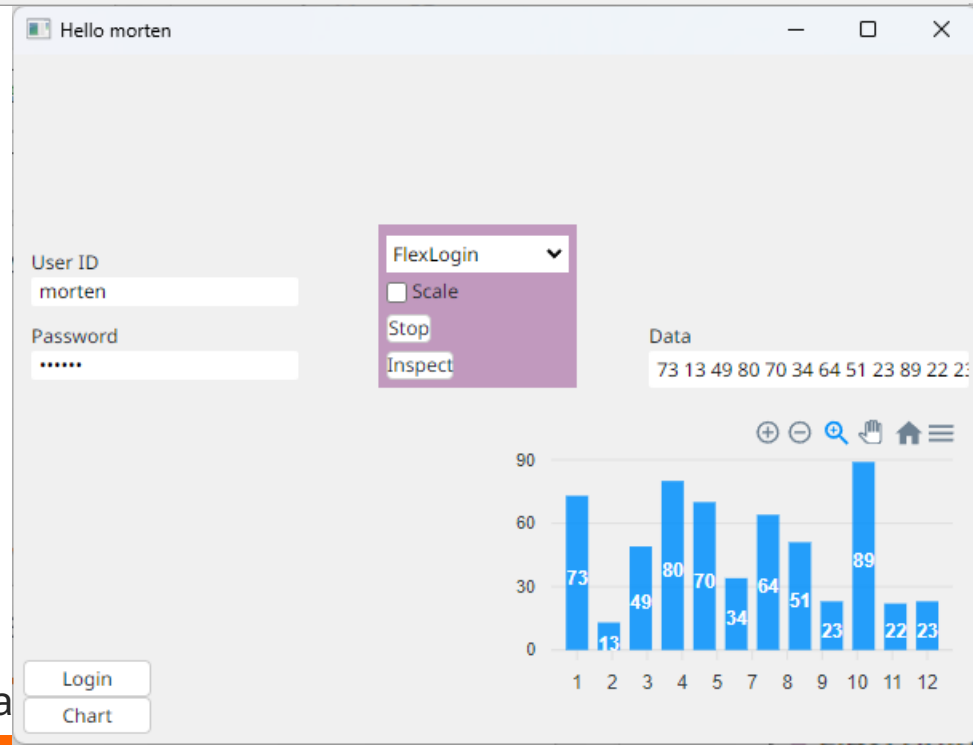
	A	B	C	D	E	F	G
1	1	2	3	4	5	6	
2	2	4	6	8	10	12	
3	3	6	9	12	15	18	
4	4	8	12	16	20	24	
5	5	10	15	20	25	30	
6	6	12	18	24	30	36	
7	7	14	21	28	35	42	
8	8	16	24	32	40	48	
9	9	18	27	36	45	54	
10	10	20	30	40	50	60	

+ ApexCharts





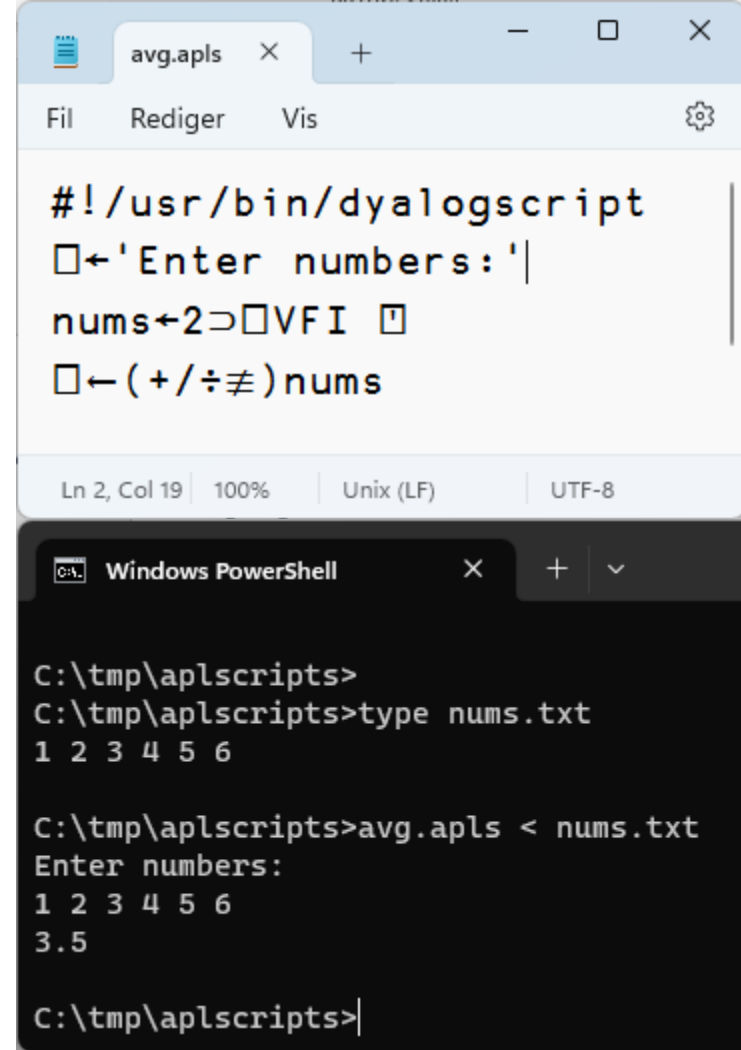
+ Flex
Layout



The Road Ahead

Script Support

- #! (hash bang) scripting
- Script engine is critical for new users
- Makes "Continuous Integration" easier
- Still a bit of a prototype in v19.0
- Will be improved in v20.0
 - Need to be able to debug scripts via RIDE



The image shows a code editor window titled 'avg.apls' with a menu bar containing 'Fil', 'Rediger', 'Vis', and a settings icon. The editor contains the following APL script:

```
#!/usr/bin/dyalogsript
⊞←'Enter numbers: '|
nums←2⊃⊞VFI ⊞
⊞←(+/÷≠)nums
```

Below the editor is a Windows PowerShell terminal window. The terminal shows the following commands and output:

```
C:\tmp\aplscripts>
C:\tmp\aplscripts>type nums.txt
1 2 3 4 5 6

C:\tmp\aplscripts>avg.apls < nums.txt
Enter numbers:
1 2 3 4 5 6
3.5

C:\tmp\aplscripts>
```

Behind / Reverse Compose

Monadic: $f \circ g \ \omega \iff (f \ \omega) \ g \ \omega$

From $\leftarrow c \circ \square$ A Enclose behind Index
1 2 2 1 From $\square A$

ABBA

$f \leftarrow 5 \circ <$ A Predicate function
 $f \circ /$ 2 7 1 8 2 8 A Filter by f

7 8 8

$[/ \circ =$ 2 7 1 8 2 8 3 A Max behind Equal

0 0 0 1 0 1 0

Behind / Reverse Compose

Dyadic: α f o g ω \leftrightarrow (f α) g ω

10 l o ϵ 2 3 5 8 A (l10) ϵ 2 3 5 8
0 1 1 0 1 0 0 1 0 0

Sort \leftarrow ϵ Δ o \square A Universal sorting function
Sort 'mississippi'
iiiimppssss

10 30 20 Sort ↑ 'Ten' 'Thirty' 'Twenty'
Ten
Twenty
Thirty

Behind / Reverse Compose



For Conor, with Thanks: Σ & Δ



Tatin

Package manager for Dyalog APL
(A tasty way to package APLs)

2023

```
]z←tatin.listPackages  
{α,≠ω}⊎{(-1+ωι'-')↑ω}¨3↓z[;1]
```

```
aplteam 42  
davin 4  
dyalog 2
```

```
¯2↑z
```

```
dyalog-HttpCommand 1  
dyalog-Jarvis 1
```

2024

```
]z←tatin.listPackages  
{α,≠ω}⊎{(-1+ωι'-')↑ω}¨3↓z[;1]
```

```
aplteam 44  
davin 4  
dyalog 5 ⌘ 150% growth!
```

```
¯5↑z
```

```
dyalog-APLProcess 1  
dyalog-HttpCommand 1  
dyalog-Jarvis 1  
dyalog-NuGet 1  
dyalog-OpenAI 1
```

Medium Things on The Way

- ◆ New Shell System Function
- ◆ .NET bridge Enhancements
- ◆ HTMLRenderer Enhancements
- ◆ Static Analysis of APL Code
- ◆ Health Monitor
- ◆ HTMLRenderer Enhancements



□ SHELL to ~~replace~~ complement □ SH

Invoke OS commands from APL

- ◆ Interruptible
- ◆ Optionally return data as an asynchronous stream
- ◆ Manage stdin, stdout & stderr (& other streams) independently
- ◆ Handle variety of data encodings

D12 Tue 16:15: New Function for Shell Calls (Peter Mikkelsen)

.NET Bridge Enhancements

- The v19.0 bridge to .NET 6/7/8 is roughly on par with the Framework bridge
- We are developing new features which will ONLY target the new bridge
 - Generics, Delegates
 - Async Methods (probably not v20.0)



Static Analysis of APL Code

- Static Analysis of application code is seen as a required "best practice" by some corporations
- We are building a prototype of a tool which will
 - Detect vulnerabilities and other bad practices
 - "Lint" APL Code
- This tool will initially be licensed separately
- A free "community edition" may follow



D06 Mon 13:55: Static Analysis of APL in APL (Brandon Wilson)

D13 Wed 09:05: Co-dfns Roadmap and Updates (Aaron Hsu)

Health Monitor

Version 19.0 contains a prototype. Ideas for v20.0 include:

- Complete feature to find last known location of a "hanging" interpreter
- Sending signals to interrupt or terminate tasks
- Discoverability: allow APL process to broadcast services that it provides
- Switch `PROFILE` on and off; collect data
- Possibly add OpenTelemetry data feed



HTMLRenderer Enhancements

The HTMLRenderer continues to grow in importance

- ◆ File Upload
- ◆ Modal HTMLRenderer Windows
- ◆ More control over "Chrome"
 - ◆ The EWC project will accelerate the evolution of the HTMLRenderer

Small Things

- ◆ `ININFO` w/Callbacks
- ◆ Set File Attributes
- ◆ Unicode decomposition / normal forms



Laying New Foundations

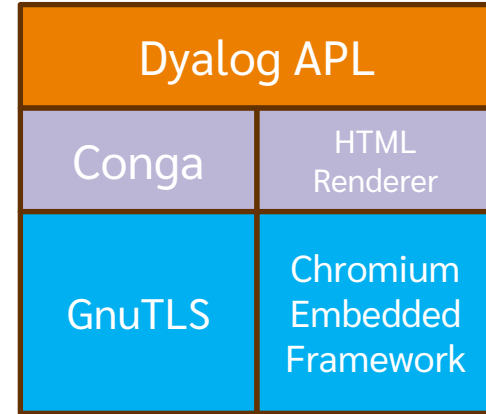
- ◆ □WC Plugin Mechanism
- ◆ Relax Interpreter Limits
- ◆ New UCMD mechanism



Plugin Architecture Benefits

- Allow users to move faster
 - Adopt new versions of Chromium or OpenSSL
- Make the Dyalog community more inclusive
 - Allow users to contribute to our eco-system
- Users can develop and share new extensions
- Makes our encryption tools transparent and verifiable
 - Easier to comply with FOSS licence constraints

D05 Mon 11:40: WC Plugins (John Daintree)



Relax Interpreter Limits

We plan to relax limitations in the interpreter, like:

- ◆ Max Rank (15)
- ◆ Number of tokens in a line (4,095)
- ◆ Number of token types
- ◆ (and many more)

Relax Limits – Why Now?

- ◆ We need more token types for new primitives, new control structures, **array notation**, ...
- ◆ Migrants have functions with >9,999 lines 😊
- ◆ Challenge: Structure has not changed for decades
 - ◆ (except new types like Unicode and Complex Numbers)
- ◆ Must avoid "big bang" data conversion
 - ◆ Different versions of APL must share data
 - ◆ (Restore & use archived component files)

D16 Wed 11:55: Interpreter Limits (John Daintree)

New UCMD Mechanism

- ◆ The existing User Command mechanism is based on old technology ("SALT")
- ◆ There are opportunities to make it
 - ◆ Easier to create utilities which provide both a proper API and User Command
 - ◆ Easier to install new User Commands
 - ◆ Faster to start a new APL session
- ◆ May be provided as an alternative in v20.0, unlikely to immediately replace old system

Tidying Up

- ◆ PCRE 10.x
- ◆ Documentation Format



Documentation

- ◆ help.dyalog.com and "core" documentation will be produced using **MkDocs** on **GitHub**
- ◆ Anyone (including **you**) can raise "issues"
 - ◆ Or even submit "Pull requests"
 - ◆ You **CAN** still email docs@ or support@
- ◆ Some tools already use MkDocs...



Link User Guide

Overview

Introduction

Technical Details and Limitations

Workspaces

History of source files as text in
Dyalog

Install and Upgrade

Installation

Version 4.0 Release Notes

Working with Link

Basic Usage

Array Formats

Configuration Files

Setting Up Your Application

Converting an Existing
Workspace to use Link

API & Command Reference

API Overview

Link.Add

...

Introduction

Link allows you to use Unicode text files to store APL source code, rather than "traditional" binary workspaces. The benefits of using Link and text files include:

- It is easy to use source code management (SCM) tools like Git or Subversion to manage your code. Although an SCM is not a requirement for Link, Dyalog **highly** recommends using Git or similar systems to manage source code that Link will load into your APL session.
- Changes to your code are **immediately** written to file: there is no need to remember to save your work. The assumption is that you will make the record permanent with a *commit* to your source code management system, when the time is right.
- Unlike binary workspaces, text source can usually be shared between different versions of APL - or even with human readers or writers who don't have APL installed at all.

Link is NOT...

- **A source code management system:** Link itself has no source code management features. As mentioned above, you will need to use a separate tool like Git to manage the source files that Link will allow you to use and modify from Dyalog APL.
- **A database management system:** although Link is able to store APL arrays using a pre-

Table of contents

Link is NOT...

Link fundamentals

Functions vs. User Commands

User commands

API functions

Further reading

Frequently Asked Questions

EWC User Guide

Overview

Introduction

Implementation

EWC versus `□WC`

Images

Getting Started

Installation

Initialising an EWC session

Configuration

Supported Features

Classes

ApexChart

BitMap

Button

Circle

Combo

Edit

Ellipse

Font

Introduction

EWC stands for "Everywhere Window Create". EWC is a cross-platform implementation of the `□WC` family of system functions (`□WC`, `□WS`, `□WG`, `□WN`, `□NQ` and `□DQ`) that are available in Dyalog APL for Microsoft Windows.

EWC only supports a subset of the functionality provided by `□wc`. This subset is growing, driven by the requirements of early adopters. The supported subset is [documented in the object reference](#).

Note

At this time, EWC is work in progress, and not supported via normal channels. A supported release of EWC is expected in 2025.

EWC can run in "Desktop" mode using an HTMLRenderer. In this mode, EWC supports multiple forms in the same way as `□wc`, creating one HTMLRenderer for each form.

Alternatively, EWC can be initialised in "Browser" mode, in which case it starts a listener on the configured port (22322 by default), and a Browser must be connected to that port. In this mode, it really only makes sense to have a single form, although modal `MsgBox`'s can be popped up if required.

Documentation: Why change?

- ◆ Easier to contribute
 - ◆ Internally and externally (and Pete has retired)
- ◆ Open formats, platform agnostic tools
- ◆ Better search
- ◆ Human-friendly, predictable URLs, like
<https://docs.dyalog.com/20.0/object-reference/properties/depth/>

Documentation

- Release Notes V19.0 >
- Windows Installation >
- UNIX Installation >
- Programmer's Guide >
- Language Reference >
- Object Reference >
- Windows UI Guide >
- Interface Guide >
- .NET Interface >
- UNIX User Guide >

Dyalog APL v20.0 Documentation

Welcome! This is the official documentation for Dyalog APL version 20.0.

Release Notes v20.0

New and improved since the last release

[→ Release Notes](#)

Installation and Configuration

How to install and configure Dyalog APL

[→ Windows Installation and Configuration Guide](#)

[→ UNIX Installation and Configuration Guide](#)

Reference Guides

Reference guides for Dyalog APL and system interfaces

[→ Programming Reference Guide](#)

[→ Dyalog APL Language Reference Guide](#)

[→ Object Reference Guide](#)

[→ Microsoft Windows: Interface Guide](#)

[→ Microsoft Windows: .NET Framework Interface Guide](#)

UI Guides

The Dyalog APL Development Environment

[→ Microsoft Windows UI Guide](#)

[→ UNIX User Guide](#)

Retiring

- ◆ Intel-based Mac versions
- ◆ 819I (replaced by IIC)
- ◆ Bundled SynCFusion Libraries
- ◆ David Liebttag's Array Editor ("replaced" by Array Notation)
- ◆ 32-bit Raspberry Pi



Other Stuff

- Link v4.1
- Kafka Interface
- Telemetry
- BSIMM Audit
- Isolates connect back to server
 - (Docker containers)
- Performance
 - Set functions
 - NSs kept alive by children
 - Garbage Collector
- Exhaustive Primitive Tests
- 64-bit ARM support
 - (Pi & AWS Image)
- New Mac installer
- Directory naming
- Keyboard layouts
- Platform Features

D15 Wed 10:55: ullu – A Test Framework for Dyalog APL (Aarush Bhat)



Big Things

- ◆ Array Notation
- ◆ Set & Get Variables
- ◆ Token-by-Token Tracing
- ◆ Everywhere WC
- ◆ Script Support
- ◆ Reverse Compose



New Foundations

- ◆ WC Plugin Mechanism
 - ◆ Open-source more components
 - ◆ HTMLRenderer, Conga, Crypto Library
- ◆ Relax Interpreter Limits
- ◆ New UCMD mechanism

Thank You!

