DYNA

Dyalog North America Meetup, 7 April 2025

# News from Dyalog Spring 2025

*Morten Kromberg, CTO*

## Morten Kromberg

- CTO of Dyalog since 2005 (20 years + 2 days)
- CTO of Adaytum Software (BI vendor) 1995-1999
  - (20 => 200 employees, sold to Cognos for $160M)
- APL Consultant for ~20 years
  - Focus on interfacing/integration, client/server architectures and developer tools / frameworks

- Still writes APL code most days ❤️
  - Contributor to GUI emulation & Source Code Mgt tools

## Agenda

- General News from Dyalog
- New Tools, Products and Services
- Releases
  - 19.0 (Q1'24) and 19.4.1
  - 20.0 (Q2'25 – in Beta Test)
  - 21.0 (Q2'26) – and beyond…

# Introducing Dyalog

- Dyalog is the ”New Kid” on The APL Block
  - Only 42 years since release of Dyalog version 1.0 in 1983
  - Has emerged as the Market Leader for APL

- Management Buy-In in 2005
  - Financed by two clients and an APL distributor / consulting firm

# The Last 20 Years

- Headcount and Turnover Increased 5x
  - Headcount now ~29 full time equivalents (+3 from 2024)
  - Steadily increasing R&D budget

- Largest sustained investment in APL technology in the history of the language

## Revenue Splits

Very Roughly

- USA vs ROW: 40 / 60
- UNIX & Linux vs Windows: 25 / 75
  - UNIX revenue is currently 90% AIX
  - We expect Linux to grow, replacing both AIX and Windows
- USA and UNIX: a smaller number of larger clients
  - Slowly changing due to migrants from US-based APL vendors

| Owners | 2008 | 2018 | Present |
|---|---|---|---|
| SimCorp (Denmark) | 32.33% | 40% | 24% |
| APL Italiana (Italy) | 32.33% | | |
| Management & Employees | 35.33% | 60% | 76% |

# Financial Status

- Ownership Today
  - 24% owned by SimCorp (Deutsche Börse)
  - 76% owned by management and employees

- Revenue steadily increasing
- Adding 1 or 2 "significant" clients each year

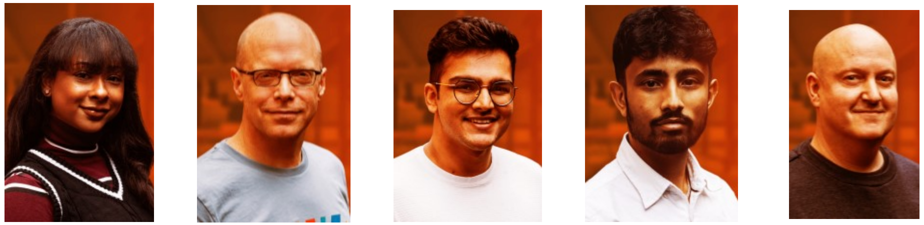
| Year | Growth |
|---|---|
| 2024 | 4.7% |
| 2023 | 8.6% |

# Dyalog – The Next Generation

2010-2021

New CEO

2022



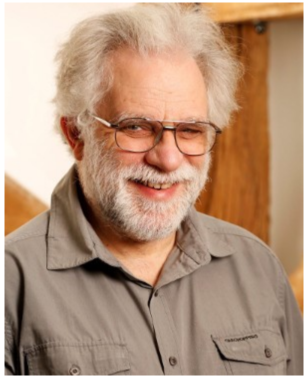
2023

2024

2025

## Retirees

- Geoff Streeter
- Gitte Christensen
- Pete Donnelly



Dyalog'24 recording at dyalog.tv:
"Let's Put the Future Behind Us"

Dyalog'24 recording at dyalog.tv:
"The New Breed Plugs In"

The New Breed Plugs In
(Panel Discussion)

Host:
Stephen Taylor (Lambent Technology)

Panellists:
Gilgamesh Athoraya (Tiamatica AB)
Martina Crippa (Dyalog Ltd)
Josh David (Dyalog Ltd)
Sandra Persson (Tiamatica AB)

## New Tools, Products & Services

- Kafka Interface
- Static Analysis of APL Code
- GUI Emulations
- LLM Interfaces
- Training & Consulting

## Maturing Technologies

- But first, a few words about some important "maturing" technologies

## The "Jarvis" Web Service Framework

Jarvis is involved in many, many new client projects, and is continuously enhanced. Recent examples:

- Support multipart/form-data for file upload and form input
- Support for serving static files
  - Automatic response-type set for 79 common content file types
- Support for application/x-www-form-urlencoded content type allows Jarvis to accept HTML form input from a served page
- Added zipping response payload if client supports it

## Tatin

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

2023

```
      ]z←tatin.listPackages
      {⍺,≢⍵}⌸{(¯1+⍵⍳'-')↑⍵}¨3↓z[;1]
 aplteam  42
 davin     4
 dyalog    2

      ¯2↑z
 dyalog-HttpCommand   1
 dyalog-Jarvis        1
```

2024

```
      ]z←tatin.listPackages
      {⍺,≢⍵}⌸{(¯1+⍵⍳'-')↑⍵}¨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
```

## Link

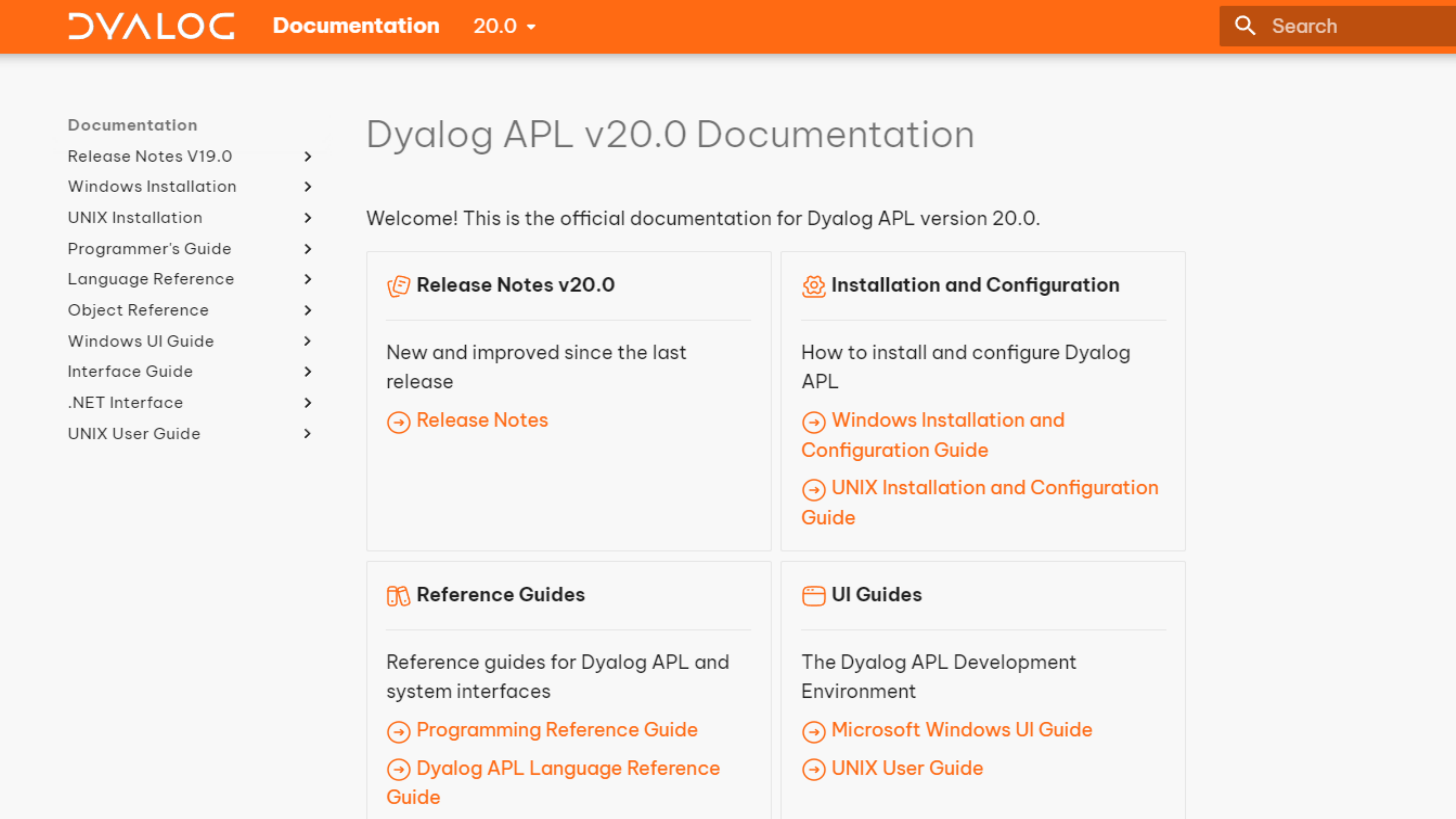
- Link maps the source code of objects in the workspace to text source files
- Link enables the use of Git or other Source Code management systems
- New users generally start with Link; old dogs are moving slowly

# Documentation

- We are moving towards open-source markdown-based documentation, produced using **MkDocs** on **GitHub**
- Anyone (including **you**) can raise issues
  - Or even submit pull requests
  - You **CAN** also email docs@ or support@dyalog.com

# Documentation: Why change?

- Easier to contribute
  - Internally and externally
- Open formats, platform agnostic tools
- Better search
  - Provide better training input to Language Models
- Human-friendly, predictable URLs



# 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

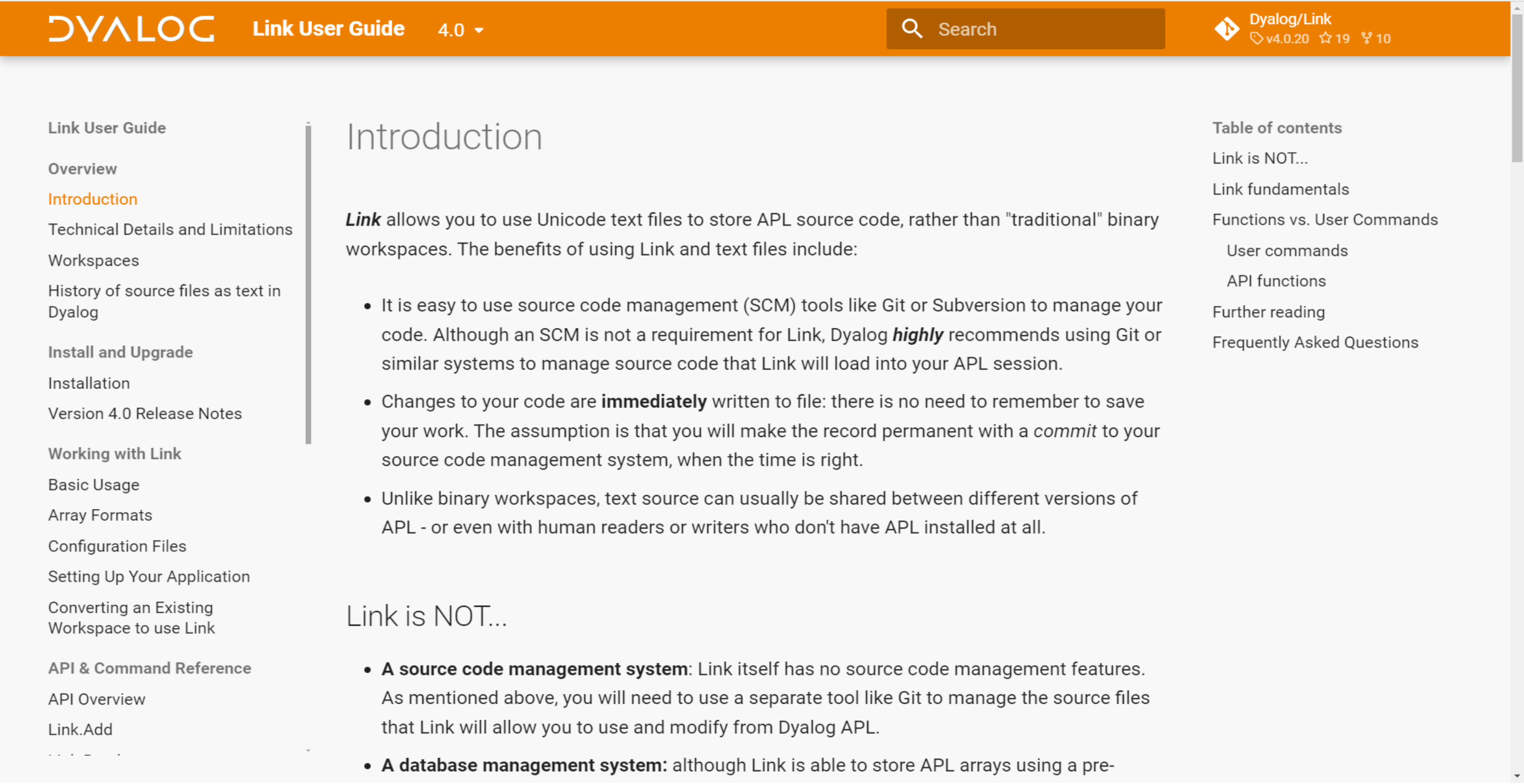
Dyalog APL Language Reference Guide

**UI Guides**

The Dyalog APL Development Environment

Microsoft Windows UI Guide

UNIX User Guide


Link User Guide 4.0

Search

Dyalog/Link v4.0.20 19 10

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...

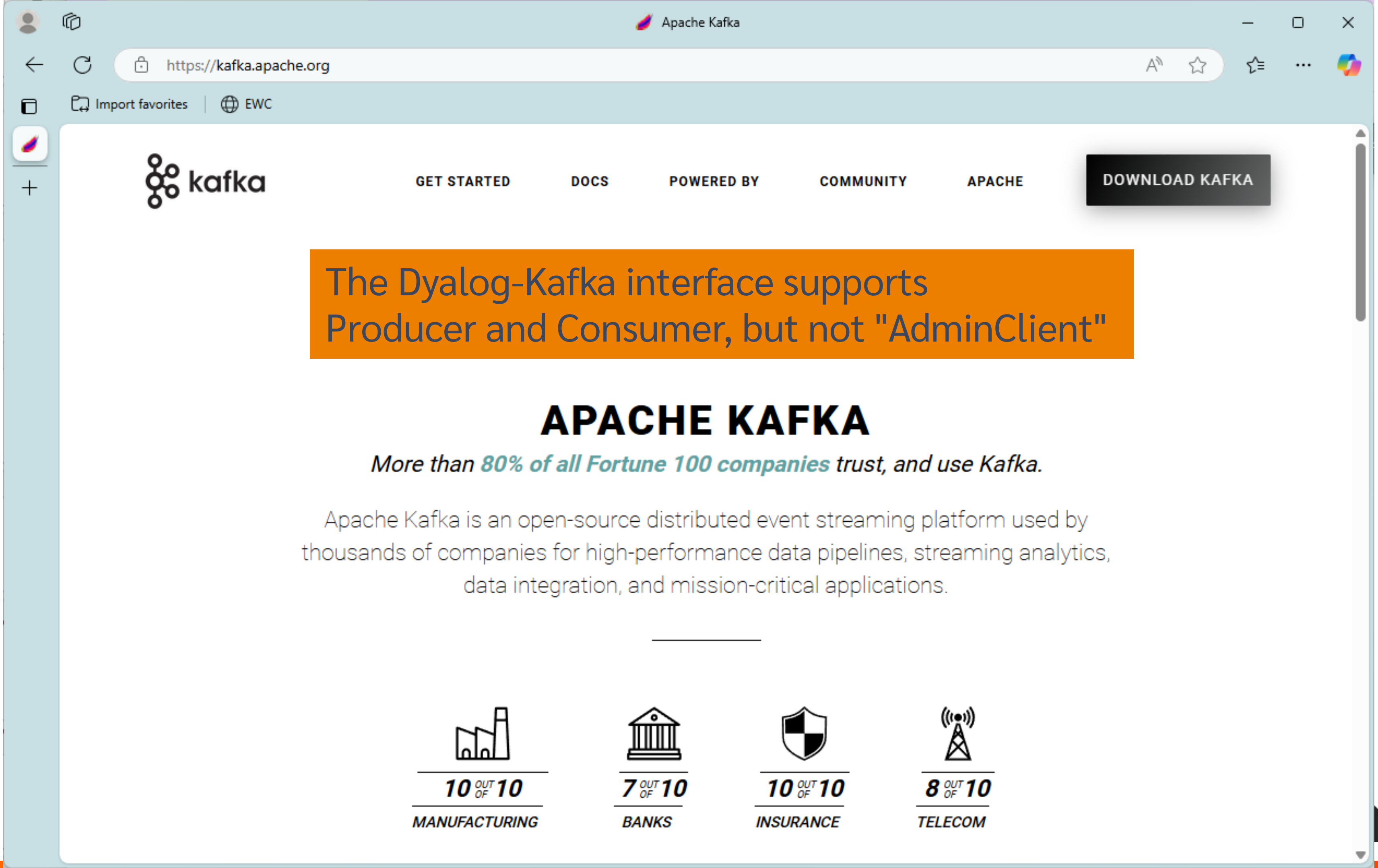
- **A source code management system**: Link itself has no source code management features. As mentioned above, you will need to use a separate tool like Git to manage the source files that Link will allow you to use and modify from Dyalog APL.
- **A database management system:** although Link is able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

Table of contents

## NB: PDF / Print is deprecated

- We will continue to produce some PDFs
- Will not spend as much energy on formatting them nicely
- Offline versions of the documentation WILL be available

## New Tools...



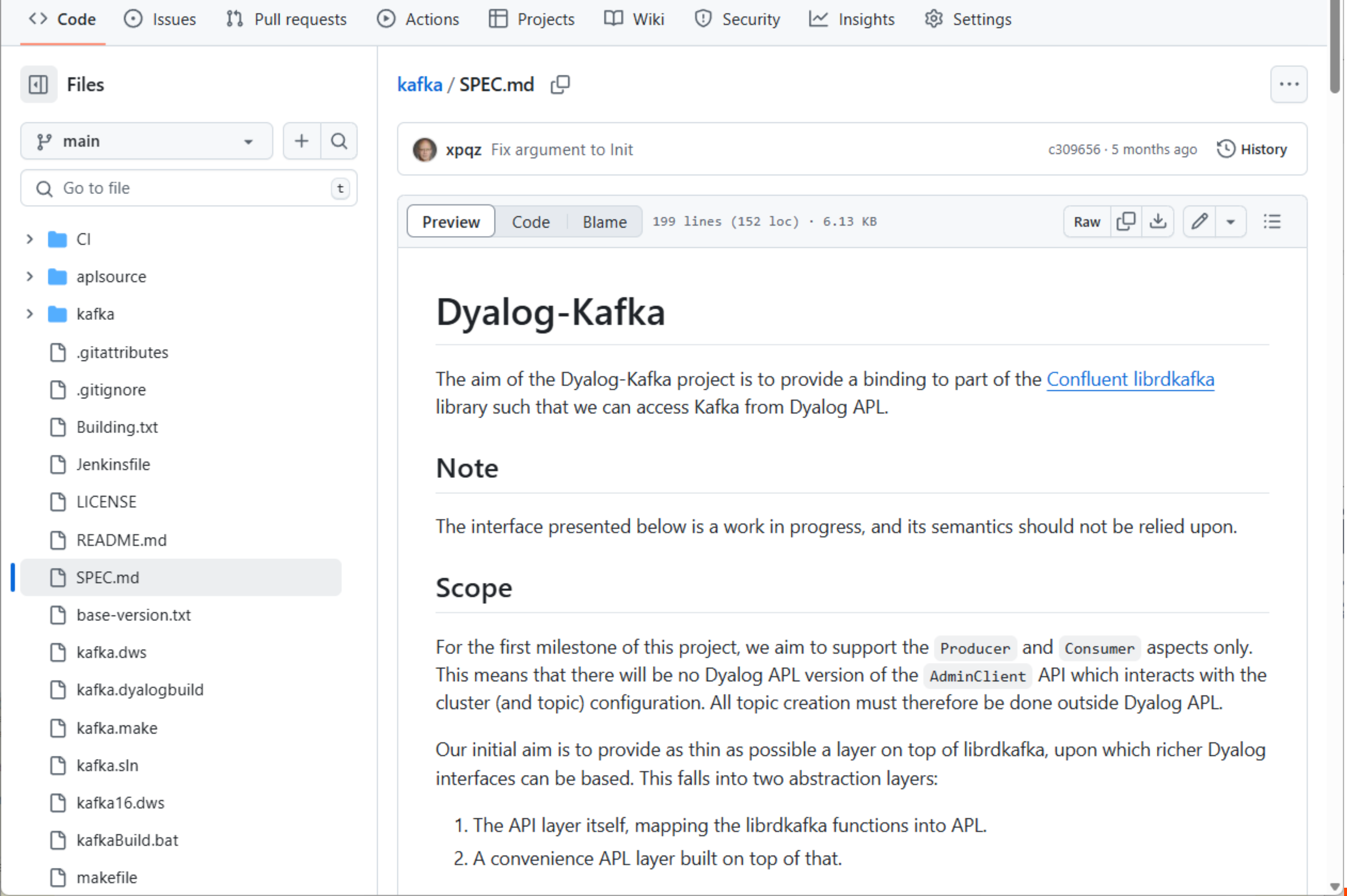
The Dyalog-Kafka interface supports Producer and Consumer, but not "AdminClient"

**APACHE KAFKA**

*More than 80% of all Fortune 100 companies trust, and use Kafka.*

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

| 10 OUT OF 10 | 7 OUT OF 10 | 10 OUT OF 10 | 8 OUT OF 10 |
|---|---|---|---|
| MANUFACTURING | BANKS | INSURANCE | TELECOM |



# Dyalog-Kafka

The aim of the Dyalog-Kafka project is to provide a binding to part of the Confluent librdkafka library such that we can access Kafka from Dyalog APL.

## Note

The interface presented below is a work in progress, and its semantics should not be relied upon.

## Scope

For the first milestone of this project, we aim to support the `Producer` and `Consumer` aspects only. This means that there will be no Dyalog APL version of the `AdminClient` API which interacts with the cluster (and topic) configuration. All topic creation must therefore be done outside Dyalog APL.

Our initial aim is to provide as thin as possible a layer on top of librdkafka, upon which richer Dyalog interfaces can be based. This falls into two abstraction layers:

1. The API layer itself, mapping the librdkafka functions into APL.
2. A convenience APL layer built on top of that.

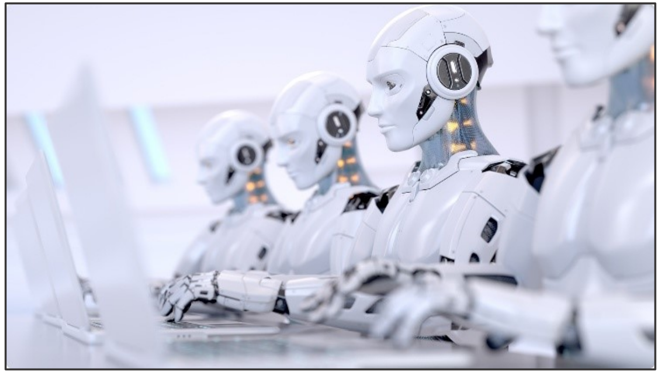
## Kafka Support

- We're trying a new approach
  - The interface is FOSS (Free and Open Source)
    - https://github.com/dyalog/kafka
  - If you want Dyalog to build and test for you and provide support, you can sign up for paid support
- Two customers currently testing the prototype
  - Meeting another potential user this week

# Static Analysis of APL Code

- Static Analysis of application code is seen as a required "best practice" by many 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 later
- Will be tested by first two clients late this year

# "Artificial Intelligence"



- AI in some form is likely here to stay
- Users (or at least their managers) expect us to support AI in a practical way
- [New] developers expect some degree of AI support
- AI providers are unlikely to prioritise APL improvements without our involvement

# AI – Potential Use Cases

- Enabling the use of LLMs "etc" from Dyalog
- Improving developer productivity with co-pilots
- Improving code quality by identifying common errors and automating boiler plate
- Generating test cases and analyzing  test logs
- Summarizing and explaining existing code
- Smart documentation

# AI for APL – "Yes But"

- Compared to other languages, there is relatively little online training data available
  - Large APL users are unwilling to make code public
- Coding styles and standards vary enormously
  - Often using domain specific notations unique to each solution
- Public code tends to be "modern"
  - (dfns and tacit functions)
- There is less boiler-plate to write, so less of a productivity boost to be had ;)

# AI – Dyalog's Action Plan

- Implement [free and open source] interfaces
- Provide good public training data: Dyalog will put [almost] all the APL code that WE write in the public domain
- Experiment with
  - Using AI for internal testing
  - Experiment with "smart searching" of our online documentation
  - Experiment with "co-pilots" at large customer sites, using private source code

## Training

- We make all our course materials available online
- Many different tutorials available, including a revised version of Gary Bergquist's tutorial
- We also run training courses and workshops
    - Inhouse or Online – your choice!
- Helping a client produce internal training videos on secure coding practices

## Consulting

- Traditionally, Dyalog has not offered consulting services
- We are now slowly growing a team
- In most cases, we will still refer clients to partners

## Open-Source

- Dyalog APL will remain closed source
- Almost everything else we produce is open-source
  - Documentation and training materials
  - Almost all tools and interfaces written in APL
  - The Kafka interface and co-dfns compiler
  - Some old C components will become open-source
    - Conga (TCP), HTMLRenderer (CEF), Cryptographic Library

## Dyalog Versions
## Recent, Present and Future

# Highlights of Version 19.0 (Q1'24)

For more, see Dyalog'24 Presentations @ https://dyalog.tv

- 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
  - ⎕NCOPY/⎕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

## Version 19.4.1

- The tools that applications are built upon are changing at an astonishing rate
- V19.4.1 re-targets many obsolete system names to modern tools and interfaces

## Revised System Names in v19.4.1

AI-related:

| | |
|---|---|
| ⎕AI | Artificial Intelligence |
| ⎕DF | LLM Degrees of Freedom |
| ⎕DL | Deep Learning level |
| ⎕DQ | Data Query |
| ⎕FIX | Fix code automatically |
| ⎕ML | Machine Learning |

Communications:

| | |
|---|---|
| ⎕AT | Bluesky protocol |
| ⎕DM | send Direct Message |
| ⎕FCHK | Fact Check |
| ⎕IO | universal Input/Output |
| ⎕RL | Real Life (inverse of ⎕SM) |
| ⎕SM | Social Media access |
| ⎕VR | Virtual Reality support |

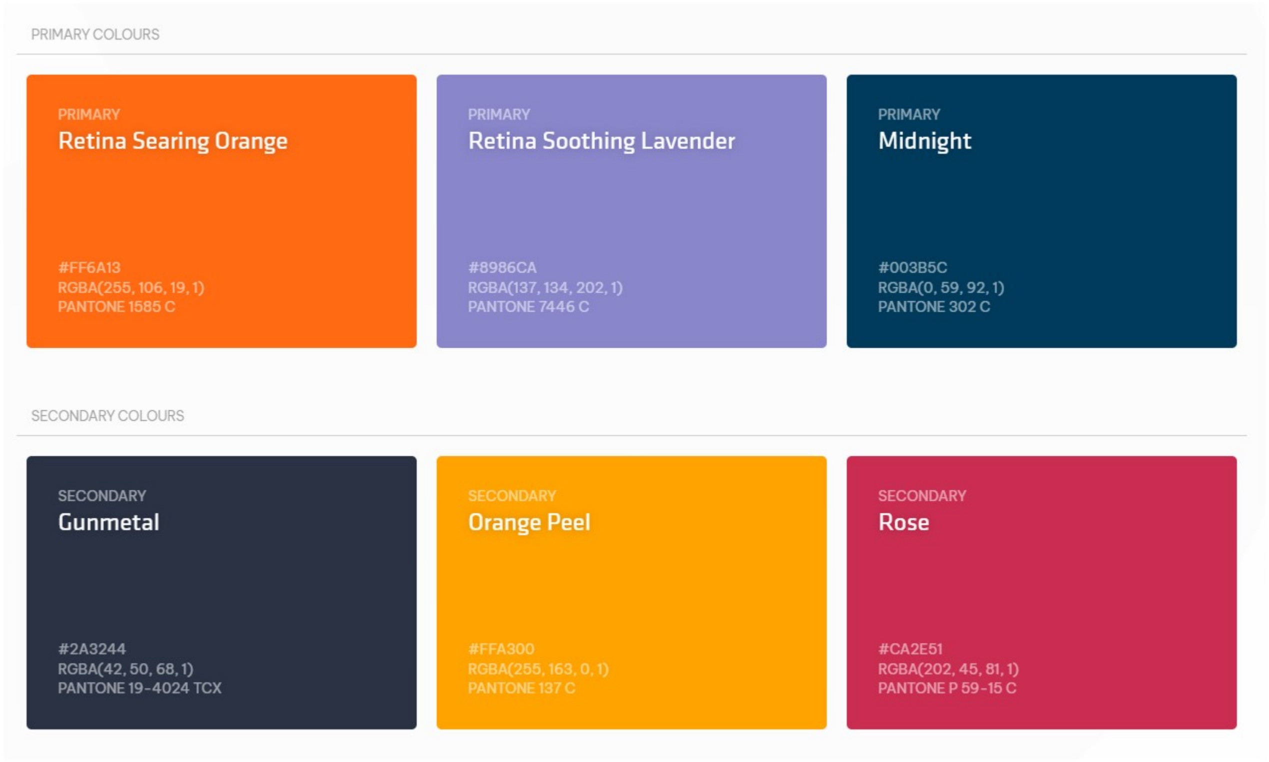
Online safety:

| | |
|---|---|
| ⎕CT | Counter-Terrorism event |
| ⎕DR | Disaster Recovery event |
| ⎕PW | Password manager |
| ⎕SHADOW | deep state integration |
| ⎕STATE | official government integration |
| ⎕WC | for when you really need to go |
| ⎕WX | weather control |

Read all about it at dyalog.com/blog

| | |
|---|---|
| ⎕ATX | motherboard properties |
| ⎕FUNTIE | deliver clothing |
| ⎕FX | toggle special effects |
| ⎕NA | (not applicable) |
| ⎕PP | PowerPoint mode |
| ⎕RTL | order of execution |

## Mike Mingard – Brand manager



PRIMARY COLOURS

PRIMARY
Retina Searing Orange
#FF6A13
RGBA(255, 106, 19, 1)
PANTONE 1585 C

PRIMARY
Retina Soothing Lavender
#8986CA
RGBA(137, 134, 202, 1)
PANTONE 7446 C

PRIMARY
Midnight
#003B5C
RGBA(0, 59, 92, 1)
PANTONE 302 C

SECONDARY COLOURS

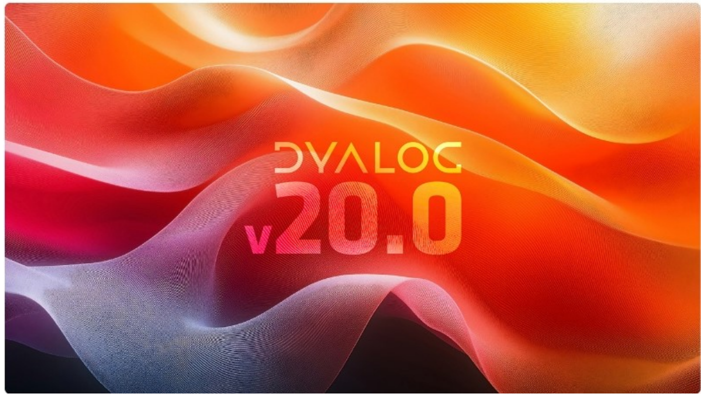
SECONDARY
Gunmetal
#2A3244
RGBA(42, 50, 68, 1)
PANTONE 19-4024 TCX

SECONDARY
Orange Peel
#FFA300
RGBA(255, 163, 0, 1)
PANTONE 137 C

SECONDARY
Rose
#CA2E51
RGBA(202, 45, 81, 1)
PANTONE P 59-15 C

DYALOG
v20.0

## Highlights of Version 20.0



In Beta, planned release in Q2

- Array Notation
- Set & Get Variables
- Token-by-Token Debugging
- Reverse Compose
- Shell System Function
- .NET Bridge Enhancements
- New Platform: ARM64

# Array Notation

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

≡

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

```
z←⍉⍪0 6 1 8
z⍪← 1 4 1 4
z⍪← 2 7 1 8
z⍪← 3 1 4 2
```

≡

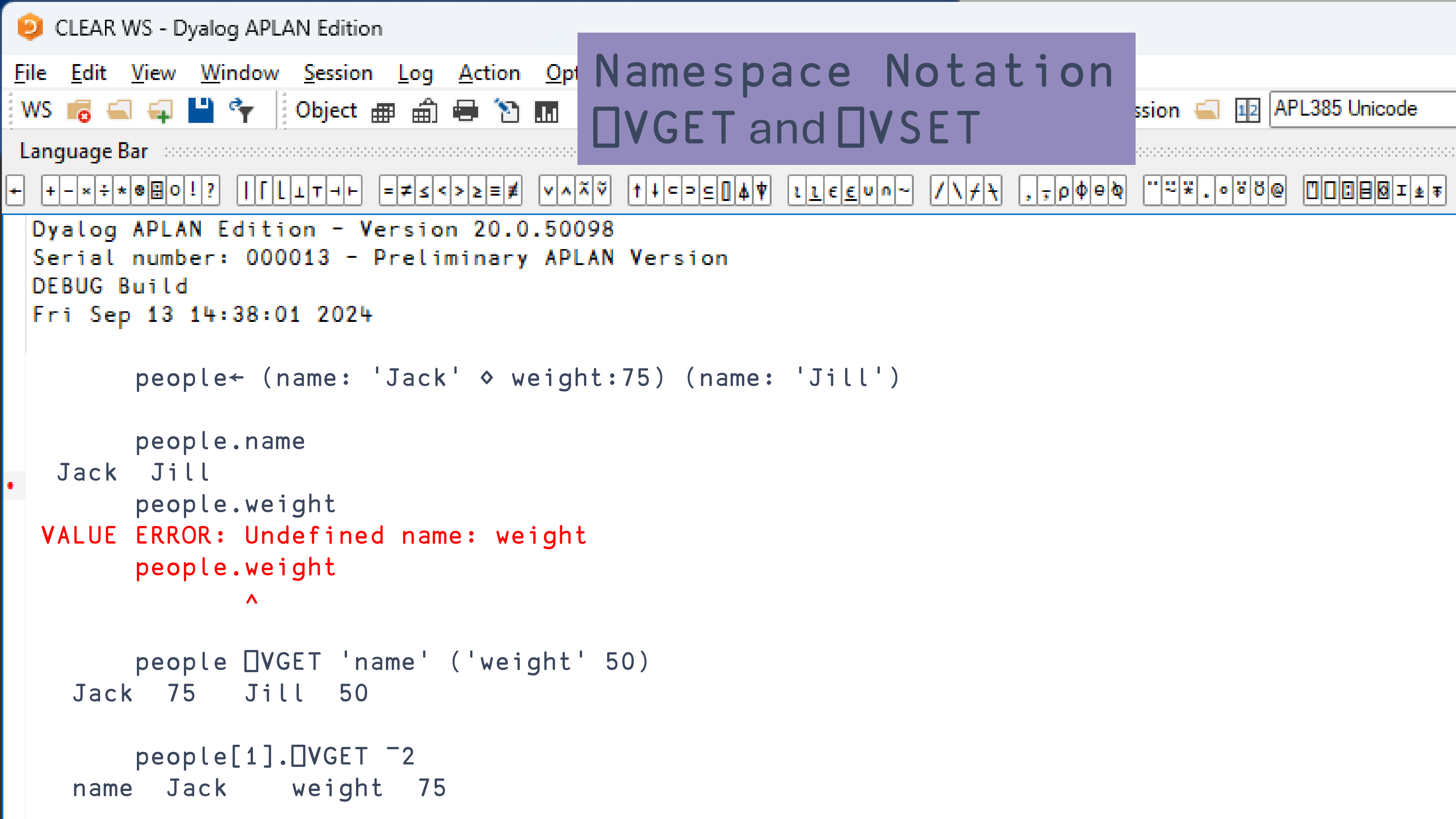
```
[0 6 1 8
 1 4 1 4
 2 7 1 8
 3 1 4 2]
```

```
z←⍪10
z⍪←20
z⍪←30
z⍪←40
```

≡

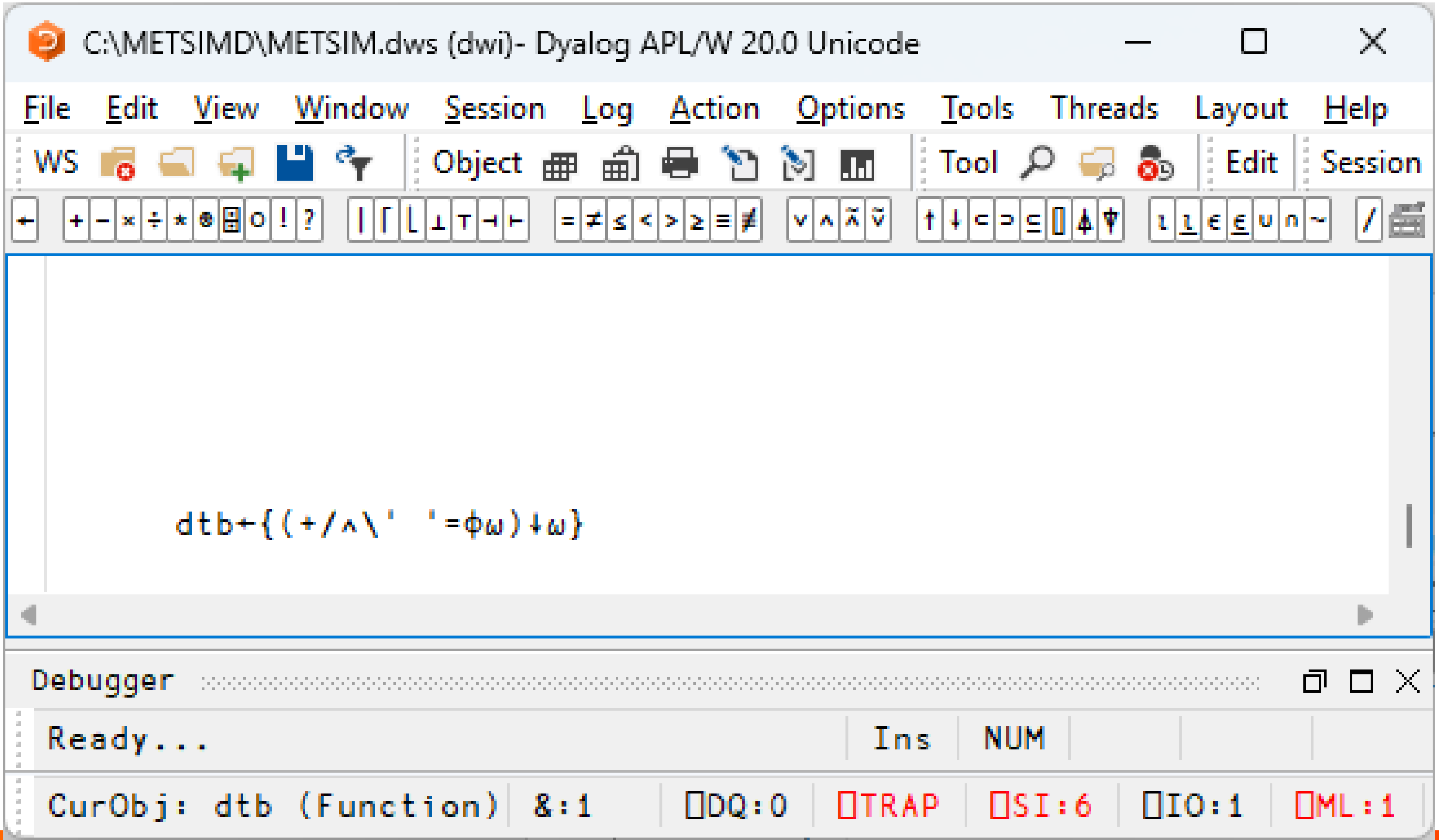
```
[10
 20
 30
 40]
```

# Namespace Notation ⎕VGET and ⎕VSET


CLEAR WS - Dyalog APLAN Edition
File Edit View Window Session Log Action Opt
WS Object
APL385 Unicode
Language Bar

```
Dyalog APLAN Edition - Version 20.0.50098
Serial number: 000013 - Preliminary APLAN Version
DEBUG Build
Fri Sep 13 14:38:01 2024

      people← (name: 'Jack' ⋄ weight:75) (name: 'Jill')

      people.name
 Jack  Jill
      people.weight
VALUE ERROR: Undefined name: weight
      people.weight
           ∧

      people ⎕VGET 'name' ('weight' 50)
  Jack  75   Jill  50

      people[1].⎕VGET ¯2
  name  Jack    weight  75
```

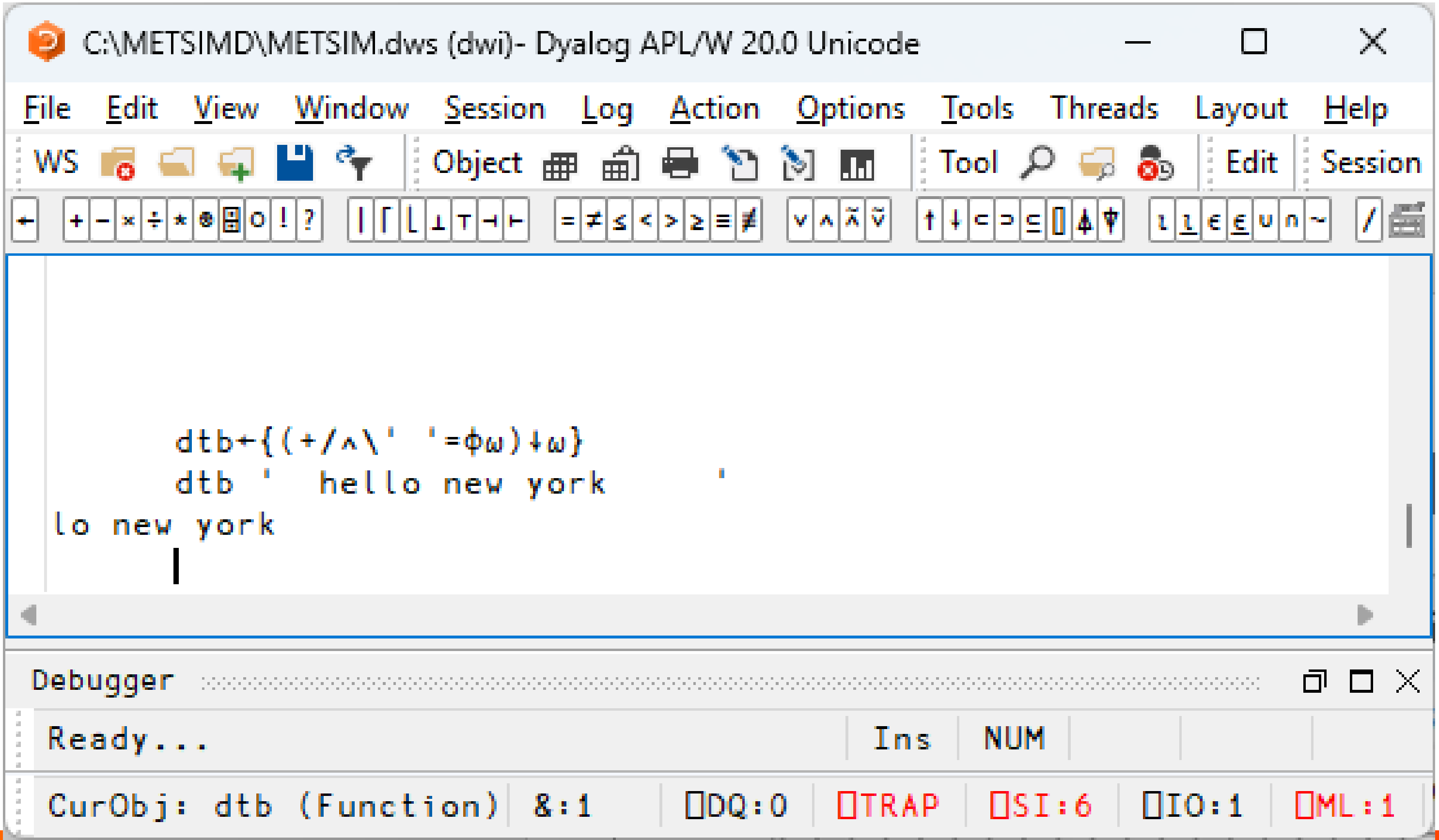
# Token by Token Debugging


C:\METSIMD\METSIM.dws (dwi)- Dyalog APL/W 20.0 Unicode
File Edit View Window Session Log Action Options Tools Threads Layout Help
WS Object Tool Edit Session
Debugger

```
dtb←{(+/∧\' '=⌽ω)↓ω}
```

```
Ready...                    Ins   NUM
CurObj: dtb (Function)  &:1   ⎕DQ:0  ⎕TRAP  ⎕SI:6  ⎕IO:1  ⎕ML:1
```

# Token by Token Debugging



C:\METSIMD\METSIM.dws (dwi)- Dyalog APL/W 20.0 Unicode

File Edit View Window Session Log Action Options Tools Threads Layout Help
WS Object Tool Edit Session

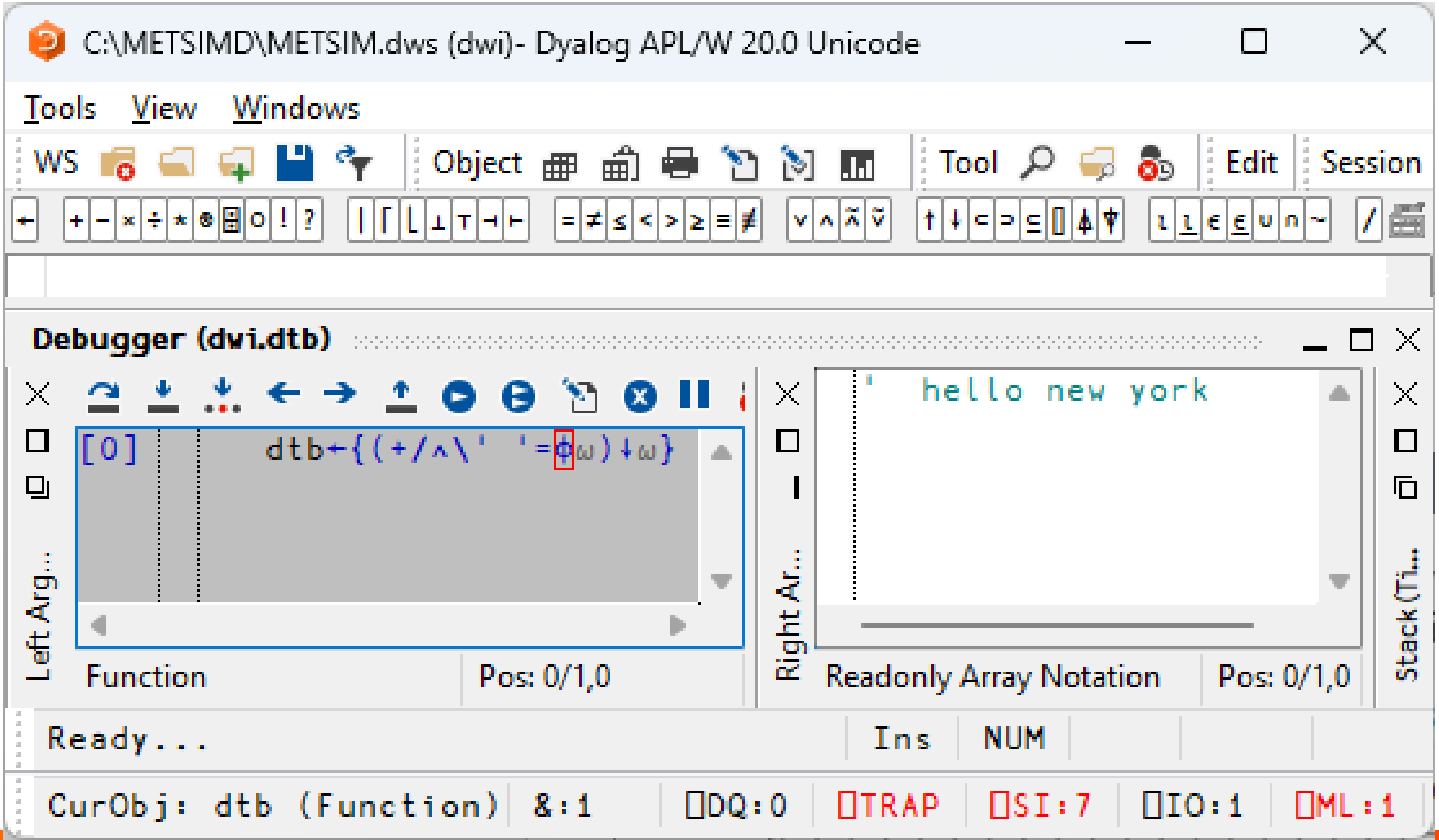
```
      dtb←{(+/∧\' '=⌽ω)↓ω}
      dtb '   hello new york       '
lo new york
```

Debugger

Ready... Ins NUM

CurObj: dtb (Function) &:1 ⎕DQ:0 ⎕TRAP ⎕SI:6 ⎕IO:1 ⎕ML:1

# Token by Token Debugging



C:\METSIMD\METSIM.dws (dwi)- Dyalog APL/W 20.0 Unicode

```
[0]      dtb←{((+/∧\' '=⌽ω)↓ω}
```

Function    Pos: 0/1,0

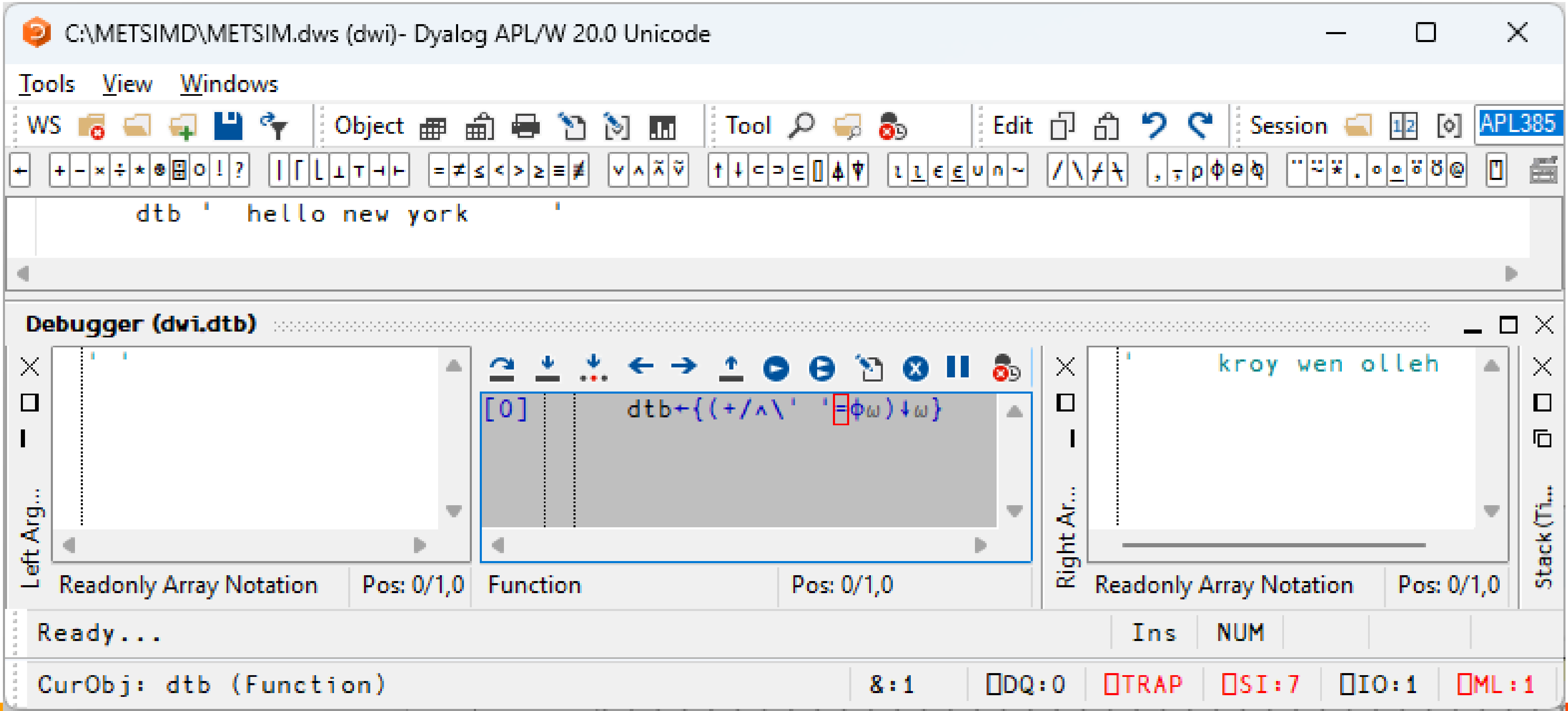
```
'   hello new york
```

Readonly Array Notation    Pos: 0/1,0

Ready...    Ins    NUM

CurObj: dtb (Function)    &:1    ⎕DQ:0    ⎕TRAP    ⎕SI:7    ⎕IO:1    ⎕ML:1

# Token by Token Debugging



C:\METSIMD\METSIM.dws (dwi)- Dyalog APL/W 20.0 Unicode

```
      dtb '   hello new york     '
```

Debugger (dwi.dtb)

```
 '  '
```

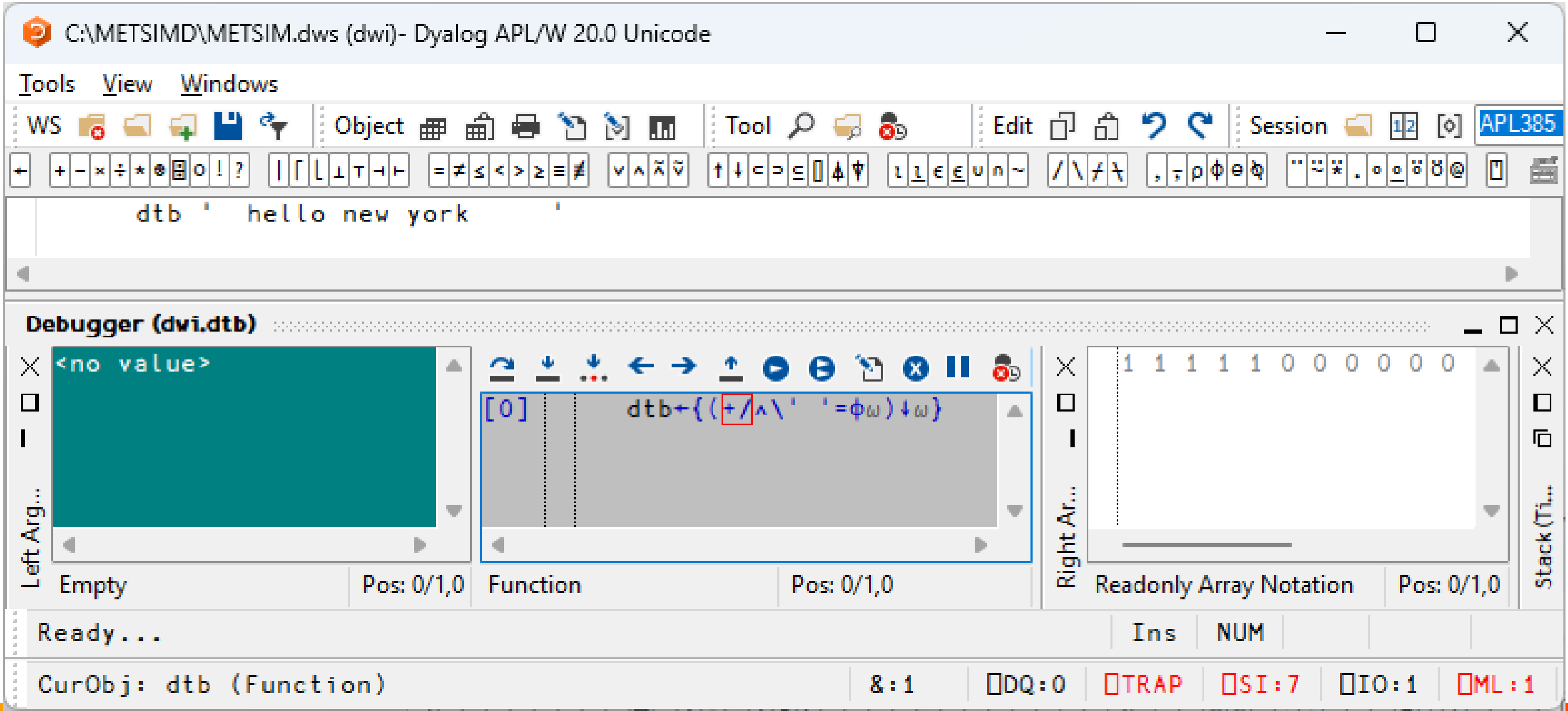
```
[0]        dtb←{((+/∧\' '=⌽⍵)↓⍵}
```

```
'     kroy wen olleh
```

Readonly Array Notation    Pos: 0/1,0    Function    Pos: 0/1,0    Readonly Array Notation    Pos: 0/1,0

```
Ready...                Ins  NUM
CurObj: dtb (Function)    &:1   ⎕DQ:0   ⎕TRAP   ⎕SI:7   ⎕IO:1   ⎕ML:1
```

# Token by Token Debugging



C:\METSIMD\METSIM.dws (dwi)- Dyalog APL/W 20.0 Unicode
Tools View Windows
WS Object Tool Edit Session APL385
Debugger (dwi.dtb)
Left Arg...
Right Ar...
Stack (Ti...

```
      dtb '   hello new york      '
```

```
<no value>
```

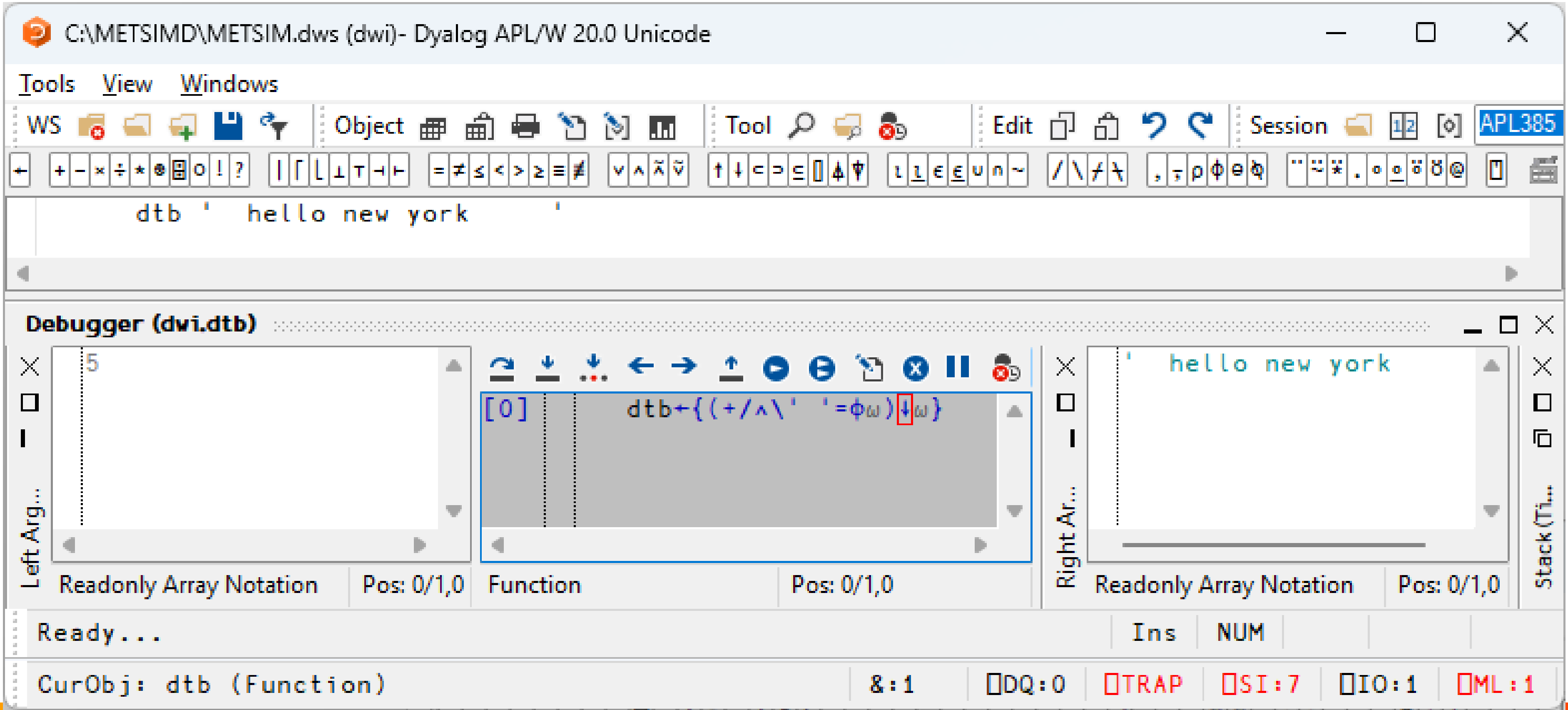
```
[0]       dtb←{((+/∧\' '=⌽⍵)↓⍵}
```

```
1 1 1 1 1 0 0 0 0 0 0
```

Empty | Pos: 0/1,0 | Function | Pos: 0/1,0 | Readonly Array Notation | Pos: 0/1,0

```
Ready...                                   Ins   NUM
CurObj: dtb (Function)      &:1   ⎕DQ:0   ⎕TRAP   ⎕SI:7   ⎕IO:1   ⎕ML:1
```

# Token by Token Debugging



```
      dtb '   hello new york     '
```

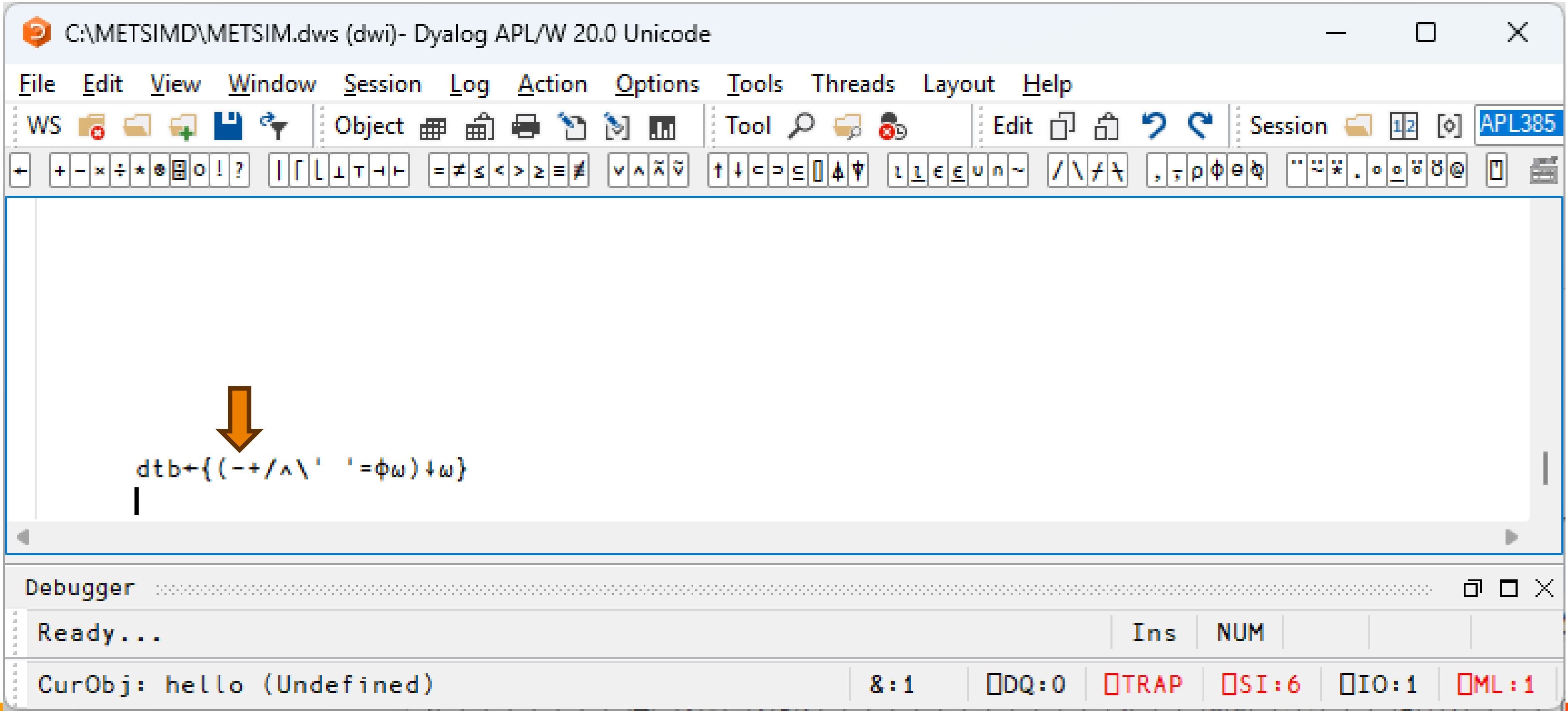
Left argument: `5`

```
[0]     dtb←{(+/∧\' '=⌽⍵)↓⍵}
```

Right argument: `'   hello new york`

```
CurObj: dtb (Function)     &:1   ⎕DQ:0   ⎕TRAP   ⎕SI:7   ⎕IO:1   ⎕ML:1
```

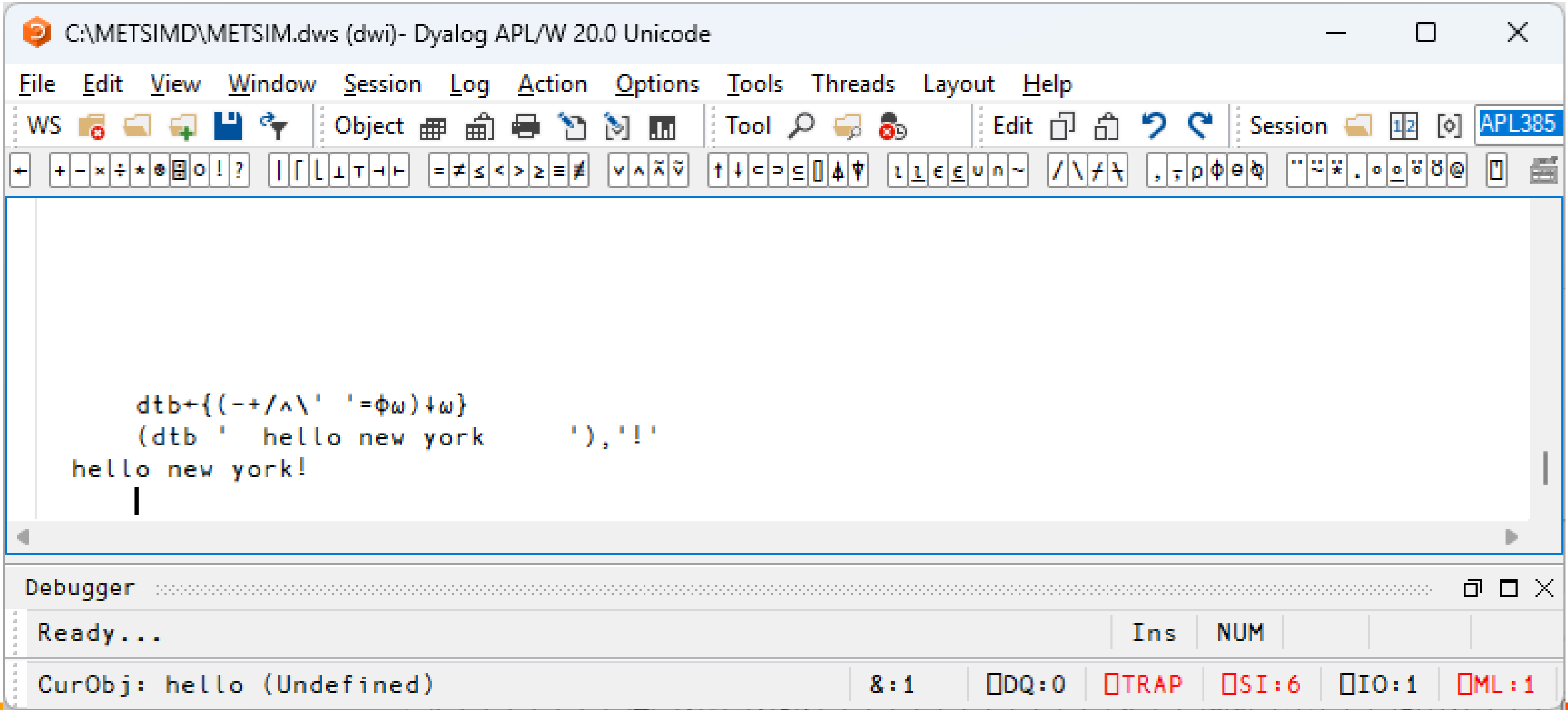
## Token by Token Debugging


C:\METSIMD\METSIM.dws (dwi)- Dyalog APL/W 20.0 Unicode
File Edit View Window Session Log Action Options Tools Threads Layout Help
WS Object Tool Edit Session APL385
Debugger

```
dtb←{(-+/∧\' '=⌽⍵)↓⍵}
```

```
Ready...                                   Ins  NUM
CurObj: hello (Undefined)   &:1   ⎕DQ:0   ⎕TRAP   ⎕SI:6   ⎕IO:1   ⎕ML:1
```

# Token by Token Debugging


C:\METSIMD\METSIM.dws (dwi)- Dyalog APL/W 20.0 Unicode
File Edit View Window Session Log Action Options Tools Threads Layout Help
WS Object Tool Edit Session APL385
Debugger

```
      dtb←{(-+/∧\' '=⌽⍵)↓⍵}
      (dtb '  hello new york     '),'!'
hello new york!
```

```
Ready...                                          Ins   NUM
CurObj: hello (Undefined)      &:1   ⎕DQ:0   ⎕TRAP   ⎕SI:6   ⎕IO:1   ⎕ML:1
```

# Behind / Reverse Compose

Dyadic: ⍺ f⍛g ⍵ ←→ (f ⍺) g ⍵

```
      10⍳⍛∊2 3 5 8      ⍝ (⍳10)∊2 3 5 8
0 1 1 0 1 0 0 1 0 0
```

## Behind / Reverse Compose

```
Monadic:  f⍛g ⍵ ←→ (f ⍵) g ⍵
```

```
      f←5∘<                    ⍝ Predicate function
      f⍛⌿ 2 7 1 8 2 8          ⍝ Filter by f
7 8 8

      ⌈/⍛= 2 7 1 8 2 8 3 ⍝ Max behind Equal
0 0 0 1 0 1 0
```

## ⎕SHELL to ~~replace~~ complement ⎕CMD

Invoke Shell Commands from APL

- Interruptible
- Optionally return data as an asynchronous stream
- Manage stdin, stdout & stderr (& other streams) independently
- Handle variety of data encodings
- Defaults to PowerShell under MS Windows

Dyalog'24: New Function for Shell Calls (Peter Mikkelsen)

# .NET Bridge Enhancements

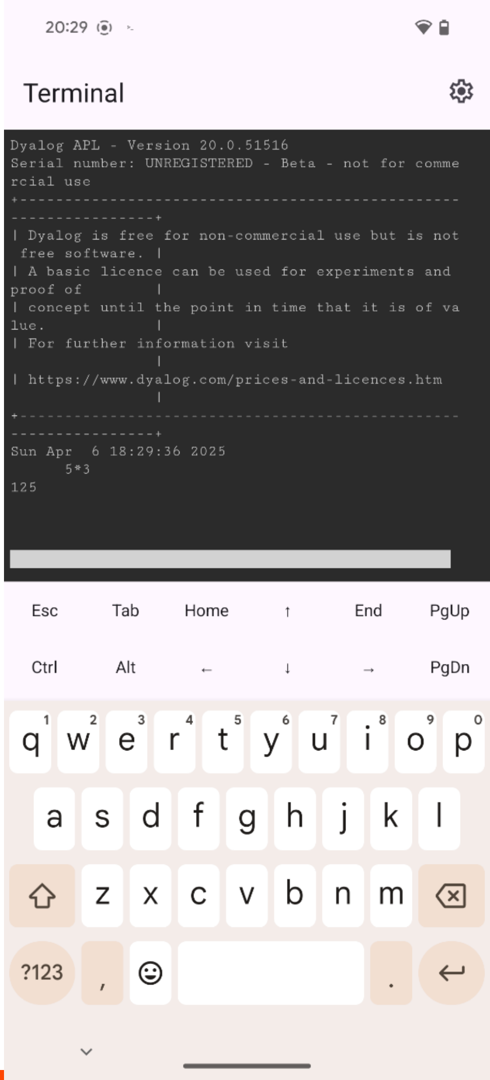
- The v19.0 bridge to .NET 8.0 is on par with the Framework bridge
- New features will ONLY target the new .NET versions (8+):
  - In v20.0: Generic Methods and Classes

# New Platform: ARM64

- macOS since v19.0
- v20.0: Linux / ARM
  - Amazon Web Service "Graviton" images
  - 64-bit Raspberry Pi
  - Seems to work on Android (but no development environment)

# New Platform: ARM64

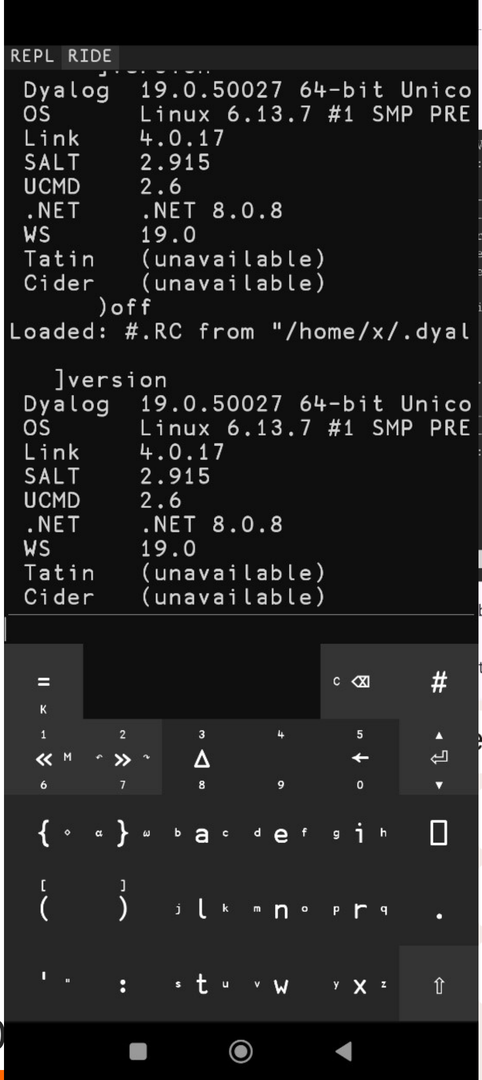
- macOS since v19.0
- v20.0: Linux / ARM
  - Amazon Web Service "Graviton" images
  - 64-bit Raspberry Pi
  - Seems to work on Android (but no development environment)



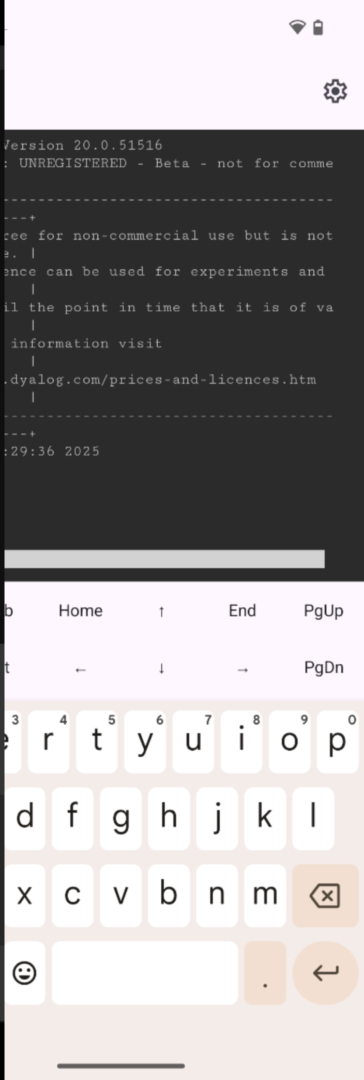
```
Dyalog APL - Version 20.0.51516
Serial number: UNREGISTERED - Beta - not for comme
rcial use
+-------------------------------------------------
----------------+
| Dyalog is free for non-commercial use but is not
 free software. |
| A basic licence can be used for experiments and
proof of        |
| concept until the point in time that it is of va
lue.            |
| For further information visit
                |
| https://www.dyalog.com/prices-and-licences.htm
                |
+-------------------------------------------------
----------------+
Sun Apr  6 18:29:36 2025
      5*3
125
```

# New Platform: ARM64

- macOS since v19.0
- v20.0: Linux / ARM
  - Amazon Web Service "Graviton" images
  - 64-bit Raspberry Pi
  - Seems to work on Android (but no development environment)



```
 Dyalog   19.0.50027 64-bit Unico
 OS       Linux 6.13.7 #1 SMP PRE
 Link     4.0.17
 SALT     2.915
 UCMD     2.6
 .NET     .NET 8.0.8
 WS       19.0
 Tatin    (unavailable)
 Cider    (unavailable)
      )off
Loaded: #.RC from "/home/x/.dyal

   ]version
 Dyalog   19.0.50027 64-bit Unico
 OS       Linux 6.13.7 #1 SMP PRE
 Link     4.0.17
 SALT     2.915
 UCMD     2.6
 .NET     .NET 8.0.8
 WS       19.0
 Tatin    (unavailable)
 Cider    (unavailable)
```



```
Version 20.0.51516
: UNREGISTERED - Beta - not for comme

------------------------------------
---+
ree for non-commercial use but is not
e. |
ence can be used for experiments and
   |
il the point in time that it is of va
   |
 information visit
   |
.dyalog.com/prices-and-licences.htm
   |
------------------------------------
---+
:29:36 2025
```

# Sketch of Version 21.0

- Open-Source Components
- .NET Generics
- Parse date-times
- Open Telemetry
- Script Support
- Language
  - Enhanced Key Operator
  - New Selection Primitive
  - :Disposable, :Finally
- Reap dividends of Namespace Notation

# Selected v21.0 Features...

OpenTelemetry

- Growing use of APL as an engine for services in multi-tier or microservice architectures
- Monitoring and debugging these complex systems requires recording telemetry
- Version 21.0 will make it easy to emit logs using the OpenTelemetry framework

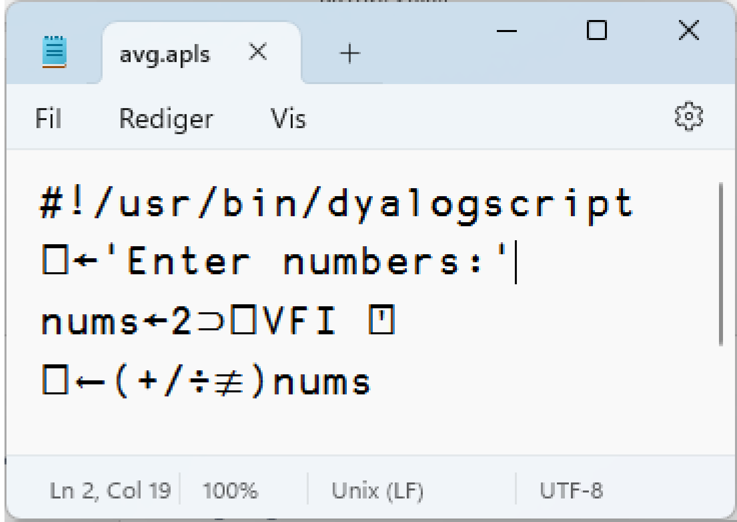
# Open Source Components

Release as open-source "plugins":

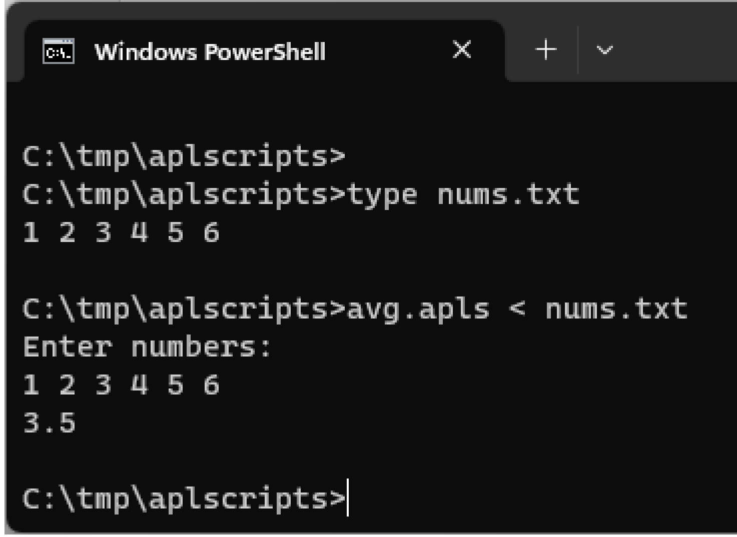
- HTMLRenderer (CEF)
- Conga (TCP/IP)
- Cryptographic Library

# "Script Engine"

- #! (hash bang) scripting
- Script engine is popular with new users **AND** "Continuous Integration" engineers
- Has been gradually enhanced for several releases
- Will be further improved in v21.0
  - RIDE debugging of scripts



```
#!/usr/bin/dyalogscript
⎕←'Enter numbers:'
nums←2⊃⎕VFI ⍞
⎕←(+/÷≢)nums
```



```
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>
```

## Parse Dates & Times

Version 20.0:

```
      ⎕←now←1 ⎕DT 'J' ⍝ current local time
45751.57121
      1 ¯1 ⎕DT now     ⍝ Convert from Day No (1) to ⎕TS (¯1) format
 2025 4 5 13 42 32 632
      format←'__es__ hh:mm Dddd DD Mmmm'
      format (1200⌶) now
  13:42 Sábado 05 Abril
```

Version 21.0: Absorb `1200⌶` and its inverse into ⎕DT:

```
      1 format ⎕DT now
  13:42 Sábado 05 Abril
      format ¯1 ⎕DT '13:42 05 Abril'
 2025 4 5 13 42 0 0
```

DYNA

## Dividends of Namespace Notation

-
  ```
  ⎕SIGNAL (EN:11 ⋄ EM:'USER ERROR' ⋄
      Message:'Problem between screen and chair')
  ```

-
  ```
  ⎕NEW 'Timer' (Active:0 ⋄ Interval:500)
  ```
- .NET & COM named arguments:

```
pop3.Connect (port: 1234 ⋄ SecureSocketOptions: secure)
```

vs

```
pop3.Connect 1234 secure ⎕NULL
```

(to leave CancellationToken as *default*)

# Research – v21.0 or Later

- Async
- Module System
- Pocket Restructuring
- External IDE integration
  - Debug Protocol
  - Copilots, etc?
- Harness AI
- "Zero Copy" data sharing using Arrow or similar?
- Embed APL in Python?
- Integration of co-dfns compiler

## Key with Vocabulary

- Allow operand to be an array

```
      ≢⍤('aeiou'⌸)'an elephantine appetizer'
3 5 2 0 0
```

# Last & From (aka "Sane Indexing)

- Last

```
      ⊇1 2 3 4
4
```

- From

```
      1 2 2 1 ⊇ 'ABCD'
ABBA
      2 1 ⊇ 3 4⍴⍳12
5 6 7 8
1 2 3 4
```

## Leading Axis Agreement

- Allow scalar functions to handle arrays where the shape of one array is the prefix of another
- Below, shape of left arg is 2, right is 2 3:

```
      10 20 + 2 3⍴⍳6
11 12 13
24 25 26
```

- Implemented in J and BQN

# Selected Features 2006-2025

- Web Server and Web Service Frameworks
- Run APL as a Windows Service
- Public Docker Containers
- Remote IDE for debugging services
- 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* size limits
- Unicode Support; 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
- Array Notation

Vast majority of features are identical across all platforms

# The Real Reason to Pick Dyalog