



DYALOG



APL Germany

News from Dyalog

Gitte Christensen, CEO

➔ *Morten Kromberg, CTO*

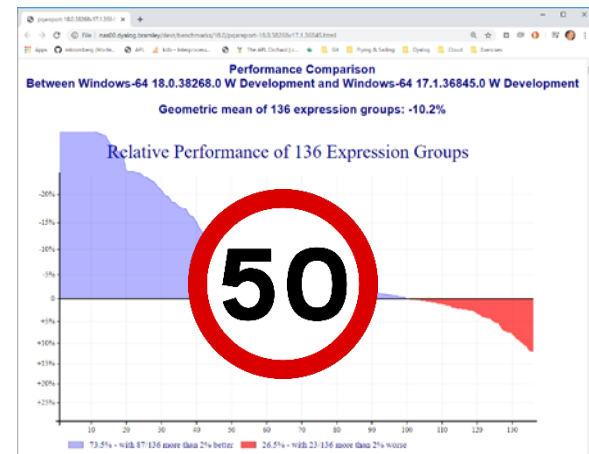
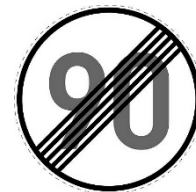
Agenda

- Current Technical Themes
- v18.0 (July 2020) - Review
- v18.1 (July 2021) - Key Features
- Next Big Things



We are slowing down

- Dyalog APL has developed at "breakneck" speed for four decades
- We nearly did break our neck in 2021
- We will invest more in
 - Quality and process (BSIMM)
 - Documentation and training materials
 - Development and integration tools



Building Security In Maturity Model

- Client organisations in sensitive sectors face demands to audit core technology providers like Dyalog
- We have decided to undertake a BSIMM audit of our development and operational practices
- In addition to allowing clients to check this box, we think this is a valuable step on the road to consistent quality and security

BSIMM



Groups of Users - Overview

- Established users of Dyalog APL
 - Large (and some small) corporations building products upon APL
 - "Internal Users"
(departments, small groups of planners & actuaries)
 - Revenue stable or very slowly growing
- New Users
 - Refugees from other APL systems (not really "new")
 - Functional programmers attracted to
"functional array oriented programming" (dfns)
 - Domain Experts
(these days, users of Python, Julia, Matlab, etc)
 - Revenue still small compared to established users



ComputerHope.com



Groups of Users - Requirements

Established users: Support for IT "best practices"

- ◆ Cloud Computing
- ◆ Secure [containerised] deployment
- ◆ Continuous Integration Pipelines
- ◆ And also: Attract and employ new APLers (see below)

Make APL attractive to new users

- ◆ All of the above (fortunately)
- ◆ Modern, on line training materials
- ◆ A lively on line community and open-source tools they can see and contribute to
- ◆ Ability to combine APL with Python, etc...



ComputerHope.com



Upcoming Versions – Technical Themes

- ◆ System Integration: Tools and Frameworks
 - ◆ Editors & Source Code Management Systems
 - ◆ Scripts / Batch Processes
 - ◆ Continuous Integration
 - ◆ Containers
 - ◆ WebServices

- ◆ Portability
 - ◆ Remote IDE (RIDE) Improvements
 - ◆ .NET Bridge (Windows, macOS, Linux)
 - ◆ 64-bit ARM (MacOS, Raspbery Pi etc)



Version 18.0 – Recap: Language

- ◆ New Primitive Operators
 - ⋄ Constant
 - ⋄ Atop
 - ⋄ Over
- ◆ New Primitive Function
 - ≠ Unique Mask
- ◆ Integer (as opposed to Boolean) arguments to
 - ⌈ Where
 - ⊖ Partition



New

`□C` Case convert
`fög` Over
`fög` Atop
`≠Y` Unique mask
`A~` Constant
`□DT` Date-time
`1200±` Format date-time

Improved

`□JSON□ 'HighRank'`
`□JSON□ 'Dialect'`
`□R/□S '\f&' □ 'Regex'`

See Adam's five excellent webinars on version 18.0 Language features

`X<Y`
`↑[k]Y`



Version 18.0 Recap - Interfaces

- Extensions to JSON (new Variant options):
 - HighRank:** As JavaScript does not have arrays of rank >1 , when exporting from APL, split to create vectors of vectors (of vectors...).
 - Dialect:** Support "JSON5" extensions which include comments and more user-friendly formatting (for our own Configuration files).
- .NET [Core] Bridge
 - The Microsoft.NET **Framework** is being replaced by an open source, portable platform available for Windows, macOS, Linux, Android, iOS, tvOS and watchOS
 - At this time, Dyalog APL is only available for Windows, macOS, Linux
 - Was called ".NET Core", in the future just ".NET 5.0"
 - Version 18.0 added a bridge which supports **USE** of .NET Libraries



Version 18.0 Recap - Application Tools

- Case folding with `⎕C` (replaces 819I)

```
1 ⎕C 'Hello' (⎕NS'') (1 2 3)
HELLO  #.[Namespace] 1 2 3
```

- Date / Time manipulation with `⎕DT`

```
1 ⎕DT <⎕TS ⌞ TS to fractional day number
44319.66611
```

```
'__de__Dddd, DDoo mmmm YYYY; hh:mm:ss' (1200I) dt
Dienstag, 04. mai 2021; 15:59:12
```



Version 18.0 Recap: Launch on Source Files

- Pre-18.0 interpreters can be launched on
 - a binary workspace (.dws)
 - a "dyalog application file" (.dyapp, now deprecated)
- Version 18.0 allows ANY APL source file.
- APL can automatically start running:
 - Functions (.aplf)
 - Namespaces (.apln)
 - Classes (.aplc)
- For other types of files, you must also specify the LX= parameter



Version 18.0 Recap: Configuration Files

- Identical across platforms
- Easily readable & editable
- Cascading configuration files provide flexible configuration of
 - each application
 - each version of APL
 - each user

```
App.dcfg - Notepad
File Edit Format View Help
{ Settings: {
  /* Dyalog Configuration */
  maxws: "128M",

  /* Application Configuration */
  app_server_port: 5556,
  use_secure_sockets: 1,
}
}
```



Version 18.1 Summary

A "small" release, with a lot of internal work on testing frameworks, and new tests to go with them, to guarantee future quality

- ◆ Shebang (#!) scripting
- ◆ Link 3.0 for text source management
- ◆ Remote IDE (RIDE) 4.4 (focus on debugging threaded services)
- ◆ Easier-to-use docker containers
- ◆ Jarvis for HTTP/REST services
- ◆ □JSON representation of tables
- ◆ Non-Linear Random Distributions
- ◆ A lot of work on the "issue backlog" + lots of testing



Version 18.1: Application Tools

- Non-linear Random Distributions (Experimental)
- Jarvis – for RESTful or simple HTTP/JSON services
- JSON extension for representing matrices



Non-Linear Distributions (168081)

Some distributions are difficult to generate accurately **AND** efficiently in APL

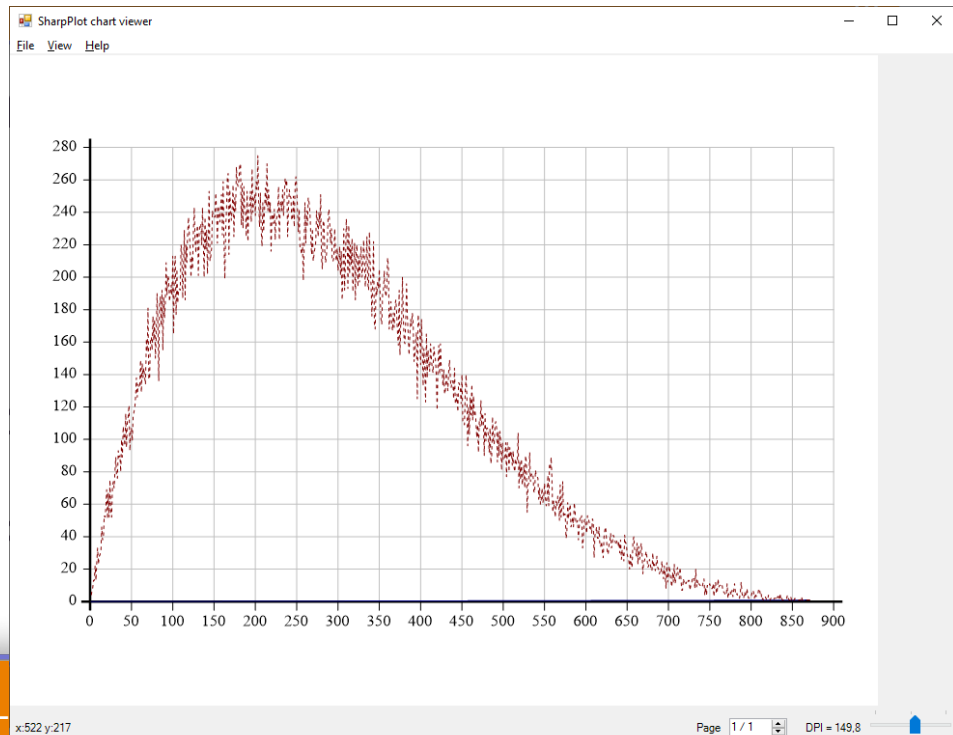
Distribution	Parameter 1	Parameter 2	Domain Rules
"Uniform"	a	b	$a < b$; A numeric interval. Example: 1.0 7.6
"Beta"	a	b	$a > 0$ AND $b > 0$
"Bernoulli"	probability		$\text{probability} \geq 0$ AND $\text{probability} \leq 1$
"Binomial"	trials	probability	trials is an integer ≥ 0 ; $\text{probability} \geq 0$ AND $\text{probability} \leq 1$
"Cauchy"	location	scale	location unrestricted; $\text{scale} > 0$
"Chi Squared"	degree_of_freedom		$\text{degree_of_freedom} \geq 0$
"Exponential"	rate		$\text{rate} \geq 0$
"F"	a	b	$a \geq \text{eps}$ AND $b \geq \text{eps}$; eps is smallest non-zero positive float number
"Gamma"	a	b	$a \geq 0$ AND $b \geq \text{eps}$; eps is smallest non-zero positive float number
"Inverse Gamma"	a	b	$a \geq 0$ AND $b \geq 0$
"Laplace"	location	scale	location unrestricted; $\text{scale} \geq 0$
"Logistic"	location	scale	location unrestricted; $\text{scale} \geq 0$
"Log Normal"	location	scale	location unrestricted; $\text{scale} \geq 0$
"Normal"	location	scale	location unrestricted; $\text{scale} \geq 0$
"Poisson"	rate		$\text{rate} \geq 0$
"Student T"	degree_of_freedom		$\text{degree_of_freedom} \geq \text{eps}$; eps is smallest non-zero positive float number
"Weibull"	a	b	$a \geq \text{eps}$ AND $b \geq \text{eps}$; eps is smallest non-zero positive float number



Non-Linear Distributions ...

Syntax: parameters (16808I) distribution shape

```
RAND←16808I  
rv←2 5 RAND 'Beta' 100000  
bc←1000 BucketCounts rv  
]chart bc
```



v18.1

Application Tools

- Generate JSON from 3 common APL representations of tables
- Useful when writing web services

```

ML←3 A Germany
Fields←'Item' 'Price' 'Qty'
Items←'Knife' 'Fork'
Price←3 4 ♦ Qty←23 45

```

```

M←mat←Fields;=>[1]Items Price Qty

```

Item	Price	Qty
Knife	3	23
Fork	4	45

```

JSON ←2 mat
JSON ←3 (Fields(>[1]Items Price Qty))
JSON ←4 (Fields(Items Price Qty))

[ {
  "Item": "Knife",
  "Price": 3,
  "Qty": 23
}, {
  "Item": "Fork",
  "Price": 4,
  "Qty": 45
}]

```



Multi-Line Input

- Experimental in v18.0, enabled with an optional switch
- On by default in v18.1 scripting mode (see next presentation)
- Allows multi-line input statements in the session
 - Control structures
 - Multi-line dfns
 - In the (hopefully near) future: "Array Notation"





Link 3.0

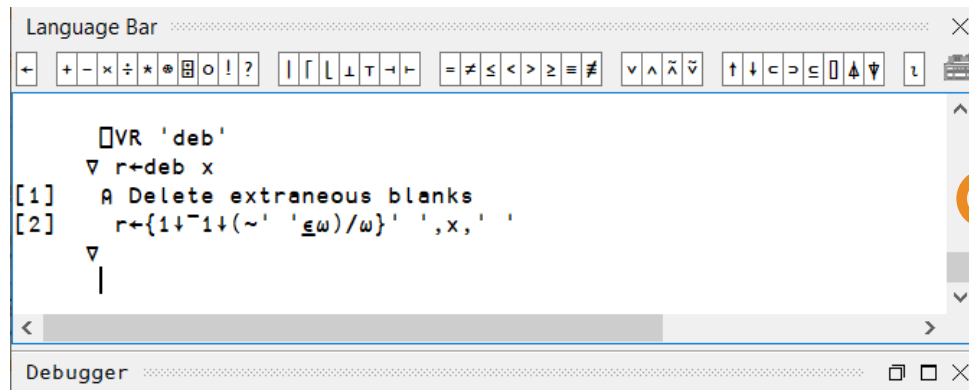




Link 3.0



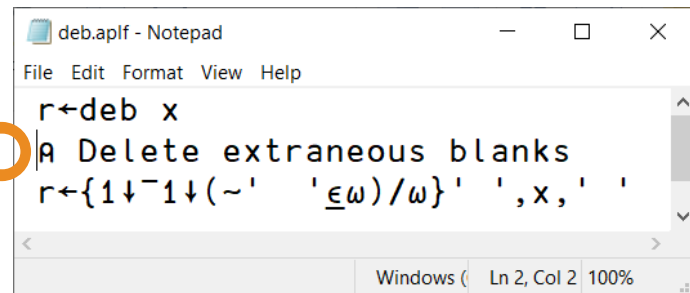
What is Link?



The screenshot shows the Dyalog APL IDE workspace. At the top is a 'Language Bar' with various icons. Below it is a code editor with the following APL code:

```
VR 'deb'  
▽ r←deb x  
[1] A Delete extraneous blanks  
[2] r←{1↓~1↓(~' '⊆ω)/ω}' ',x,' '  
▽  
|
```

Below the code editor is a 'Debugger' panel. An orange chain-link icon is positioned between this workspace and the Notepad window on the right, indicating a link.



The screenshot shows a Notepad window titled 'deb.aplf - Notepad'. The menu bar includes 'File', 'Edit', 'Format', 'View', and 'Help'. The text content is identical to the workspace:

```
r←deb x  
A Delete extraneous blanks  
r←{1↓~1↓(~' '⊆ω)/ω}' ',x,' '  
< >
```

The status bar at the bottom shows 'Windows (Ln 2, Col 2 100%'. The orange chain-link icon connects this file to the workspace.

- Each code item in the active workspace is **linked** to a file
- If the object is edited, the file is updated
- If the file is changed, the workspace is updated
- More in the next session



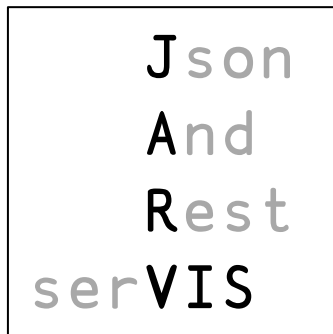
Jarvis (released separately)

If you have one or more APL functions, Jarvis can make them available as a web service

- Deploy APL code as a pragmatic HTTP/JSON service, or as a "RESTful" service if you master that paradigm
- Jarvis** combines and replaces **JSONServer** and **RESTServer**
- Can also replace **MiServer** for serving static HTTP
- About to add WebSocket publish/subscribe capability

A high percentage of new projects use Jarvis.

- Public repository at <https://github.com/Dyalog/Jarvis>



RIDE 4.4 (with Dyalog v18.1)

- Tested and enhanced for debugging **multi-threaded server applications**
- More sensitive to the active version of APL
 - Language Bar, Online Help, Syntax Colouring ...
- Remember responses to confirmation prompts
- Support multi-line input
- Documentation / examples of secure RIDE configuration
 - E.g. how to require client-side certificates



Version 18.1 Summary

A "small" release, with a lot of internal work on testing frameworks, and new tests to go with them, to guarantee future quality

- ◆ Shebang (#!) scripting
- ◆ Link 3.0 for text source management
- ◆ Remote IDE (RIDE) 4.4 (focus on debugging threaded services)
- ◆ Easier-to-use docker containers
- ◆ Jarvis for HTTP/REST services
- ◆ □JSON representation of tables
- ◆ Non-Linear Random Distributions
- ◆ A lot of work on the "issue backlog" + lots of testing



Some of the Next "Big Things"

Targeting v19.0 (Q3 2022)

- Package Manager
- .NET bridge: Support Async & Generics
- 64-Bit ARM Ports (macOS & Pi/Linux)
- Array Notation



Tatin

Finally! (We've been talking about this for years...)



Tatin: APL Package Manager

<https://tatin.dev>

- Developed by Kai Jaeger / APLTeam
Co-funded by Dyalog
- Still a prototype with rudimentary documentation
- Currently only has Kai's own tools as packages
- Dyalog will add tools as packages once 18.1 is done





Main page

This is the Tatin Registry. It holds packages designed to be used under Dyalog APL.

1. [List of all groups](#)
2. [List of all packages, aggregated by major version no.](#)
3. [List of all tags](#)
4. [List of all packages](#)
5. [Documentation Center](#)
6. [Version information regarding Tatin](#)

This server operates a "delete none" policy: once published, packages cannot be deleted.



Tatin Registry

List of all packages

Package name	Description	Major versions	Link to project
aplteam-APLGit	Git interface from Dyalog APL via Git Bash	1	github.com
aplteam-APLProcess	Start an APL process from within Dyalog APL	1	github.com
aplteam-APLTreeUtils2	General utilities required by most members of the APLTree library	1	github.com
aplteam-CodeCoverage	Monitors which parts of an application got actually executed	1	github.com
aplteam-Compare	Allows comparing and merging objects in the workspace with a file or a file with another file	1	github.com
aplteam-CompareSimple	Allows comparing objects in the workspace with a file or a file with another file	1	github.com
aplteam-DateAndTime	Utilities related to Date and Time, including doing math	1	github.com
aplteam-DotNetZip	Zippping and unzipping with .NET Core on all major platforms	1	github.com
aplteam-EventCodes	Constants with meaningful names for Dyalog error codes	1	github.com
aplteam-Execute	Start a process from within APL	1	github.com
aplteam-FilesAndDirs	Utilities for doing gymnastics with files and directories	1	github.com
aplteam-GitHubAPIv3	Utilities for dealing with GitHub repositories	1	github.com
aplteam-HandleError	Allows to catch errors on an application level; saves information that allow analyzing the error	1	github.com
aplteam-IniFiles	Allows instantiating good old INI files in APL; comes with extended syntax supporting APL-like data structures	1	github.com
aplteam-Laguntza	Managing and displaying help pages based on markdown files	1	github.com
aplteam-Logger	Allows writing to LOG files; (almost) guaranteed to never break the application	1	github.com
aplteam-MarkAPL	Converts Markdown to HTML5	1	github.com
aplteam-OS	OS-related tools for all major platforms	1	github.com

Tatin Demo











File Edit View Session Log Action Options Tools Threads Help

WS Object Tool Edit Session APL385 Unicode 16

Language Bar

```

)ns myns
#.myns
  myns.⊂FX 'r+goo x' ':If x=1' 'r+'x is one'' ':Else' 'r+'x is not one'' ':EndIf'
  ⊂VR 'myns.goo'
  ▽ r+goo x
[1]   :If x=1
[2]   r+'x is one'
[3]   :Else
[4]   r+'x is not one'
[5]   :EndIf
  ▽

]tatin.loadpackage [tatin]/aplteam-CodeCoverage #.MyPkgs
#.MyPkgs.CodeCoverage
C←NEW #.MyPkgs.CodeCoverage(,c'#.myns')
C.filename←'/tmp/myns-coverage'
C.Start 0

myns.goo 1
x is one

C.Stop 0
C.Finalize 0
/tmp/myns-coverage.dcf
⊂+1 #.MyPkgs.CodeCoverage.ProcessDataAndCreateReport C.filename
/tmp/myns-coverage.html

```

NB: If CodeCoverage had any dependencies, they would also have been loaded

Debugger Ready... Ins

CurObj: aplteam (Undefined) &:1 ⊂DQ:0 ⊂TRAP ⊂SI:0 ⊂IO:1 ⊂ML:1



The screenshot shows the APL385 IDE interface. The code editor contains the following APL code:

```

)ns myns
#.myns
  myns.⊞FX 'r+goo x' ':If x=1' 'r+'x is one'' ':Else' 'r+'x is not one'' ':EndIf'
  ⊞VR 'myns.goo'
  ▽ r+goo x
[1] :If x=1
[2]   r+'x is one'
[3] :Else
[4]   r+'x is not one'
[5] :EndIf
  ▽

]tatin.loadpackage [tatin]/aplteam-CodeCoverage #.MyPkgs
#.MyPkgs.CodeCoverage
  C+⊞NEW #.MyPkgs.CodeCoverage(, <'#.myns')
  C.filename+ '/tmp/myns-coverage'
  C.Start 0

  myns.goo 1
x is one

  C.Stop 0
  C.Finalize 0
/tmp/myns-coverage.dcf
  ⊞+1 #.MyPkgs.CodeCoverage.ProcessDataAndCreateReport C.filename
/tmp/myns-coverage.html

```

A red rectangle highlights the following code block:

```

  C+⊞NEW #.MyPkgs.CodeCoverage(, <'#.myns')
  C.filename+ '/tmp/myns-coverage'
  C.Start 0

```



The screenshot shows the APL IDE interface. The main window is the code editor, which contains the following APL code:

```

)ns myns
#.myns
  myns.⊂FX 'r+goo x' ':If x=1' 'r+'x is one'' ':Else' 'r+'x is not one'' ':EndIf'
  ⊂VR 'myns.goo'
  ▽ r+goo x
[1] :If x=1
[2]   r+'x is one'
[3] :Else
[4]   r+'x is not one'
[5] :EndIf
  ▽

]tatin.loadpackage [tatin]/aplteam-CodeCoverage #.MyPkgs
#.MyPkgs.CodeCoverage
C+⊂NEW #.MyPkgs.CodeCoverage(,⊂'#.myns')
C.filename+ '/tmp/myns-coverage'
C.Start 0

  myns.goo 1
x is one

  C.Stop 0
  C.Finalize 0
/tmp/myns-coverage.dcf
  ⊂+1 #.MyPkgs.CodeCoverage.ProcessDataAndCreateReport C.filename
/tmp/myns-coverage.html

```

The code is executed, and the output "x is one" is displayed in the console. The debugger window at the bottom shows the status "Ready..." and the current object "CurObj: aplteam (Undefined)".

Coverage Report

Watched: 1 fns/opr within #.goo

Overall 67% of the testable code is covered.

(Comment lines, empty lines, all :End* lines etc. are all ignored)

0 of the fns/opr are 100% covered.

1 are partly covered:

Function/Operator	Lines not executed	Coverage	≠
#.goo.goo	4	67%	3

Listings

#.goo.goo

```
r←goo x
[1] :If x=1
[2]   r←'x is one'
[3] :Else
→[4]   r←'x is not one'
[5] :EndIf
```

.NET Async and Generics

- Asynchronous methods are growing in popularity
 - We need to find a good way to integrate this into APL
 - (perhaps using Futures)
- The same is true for "generic" classes and methods
- We will do research into this in the v19.0 timeframe
- Also extend .NET bridge to allow export of APL code as .NET assemblies and executables



ARM 64 Ports

- Apple is moving from Intel/x64 to ARM-64
- Raspberry Pi and similar small machines are switching from 32- to 64-bit ARM



Array Notation

- Currently "modelled" in Link, will become a core language feature
- Will make it easier to
 - Write (and read) code
 - Define and edit data in the APL session
 - Or in external editors
 - Effectively use source code management systems
 - Configure and deploy systems
- Proposals refined for 5+ years
 - Many thanks to Phil Last!
 - Planning a final round of community review

```
colours←[
  'red'    (255  0  0)
  'orange' (255 165  0)
  'purple' (128  0 128)
  'green'  (  0 255  0)
  'blue'   (  0  0 255)
  'gray'   (128 128 128)
]
```



"Slowing Down" ...

...

- Package Manager
- .NET bridge: Support Async & Generics
- 64-Bit ARM Ports (macOS & Pi/Linux)
- Array Notation

- Documentation work, co-dfns compiler, etc...
- Increased testing of interpreter + all tools

