DYALOG

APL Germany

# New User Commands in Dyalog 18.2 and Ideas for the Future

*Adám Brudzewsky*

# Overview

]OUTPUT.View

]TOOLS.APLCart

]WS.Names

]OUTPUT.Repr

]EXPERIMENTAL.Get

+ Ideas for the Future



# ]OUTPUT.View



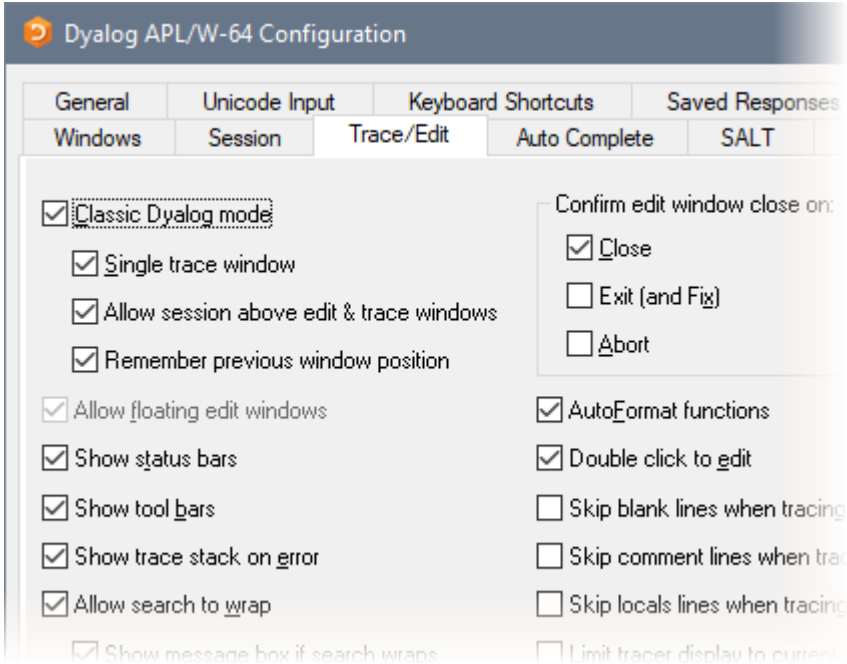
# ]OUTPUT.View

- Classic window mode



# ]OUTPUT.View

- Classic window mode



# ]OUTPUT.View

- Classic window mode
- Inspect long output



# ]OUTPUT.View

- Classic window mode
- Inspect long output

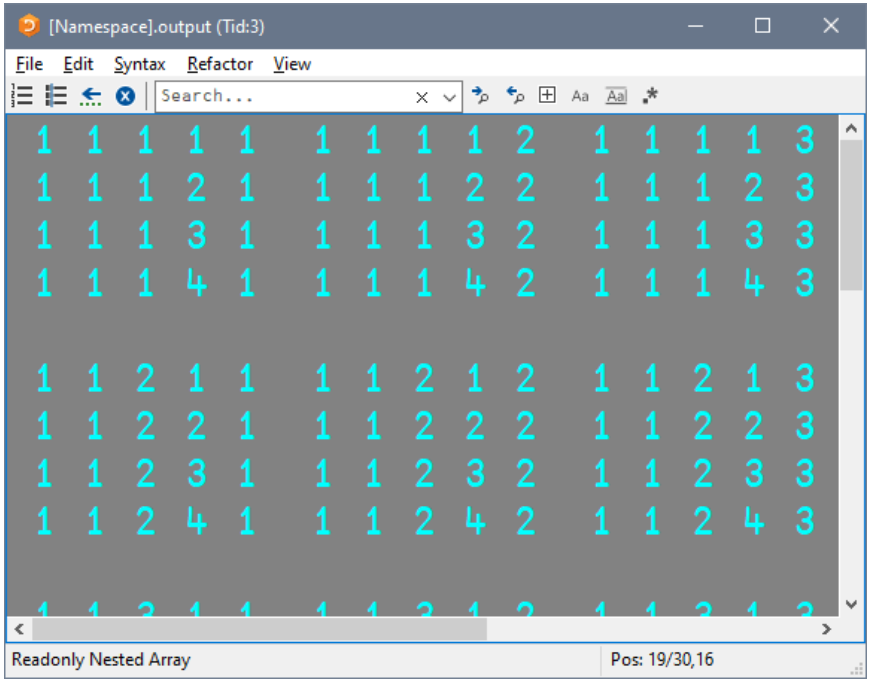
```
]view ⍳⍳5
```



# ]OUTPUT.View

- Classic window mode
- Inspect long output

```
]view ⍳⍳5
```



# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item

# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item

```
var←'oldvalue'
```

# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item

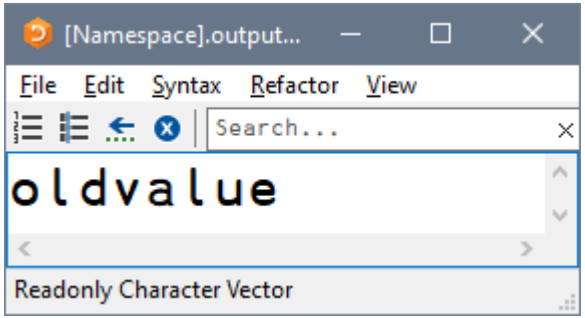
```
var←'oldvalue'
]view var
```



# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item

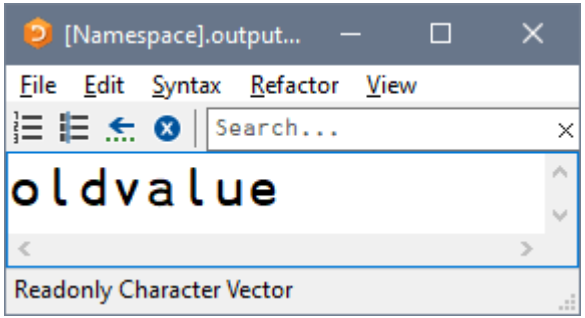
```
var←'oldvalue'
]view var
```



# ]OUTPUT.View

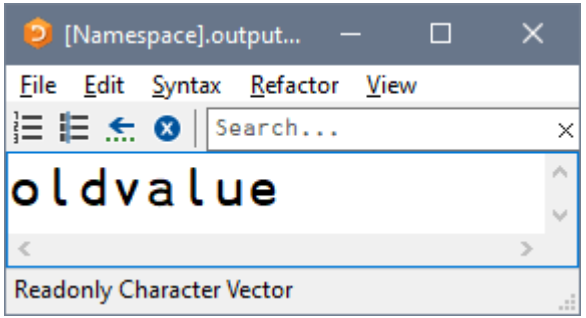
- Classic window mode
- Inspect long output
- Snapshot an APL item

```
var←'oldvalue'
]view var
(3↑var)←'new'
```



# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item

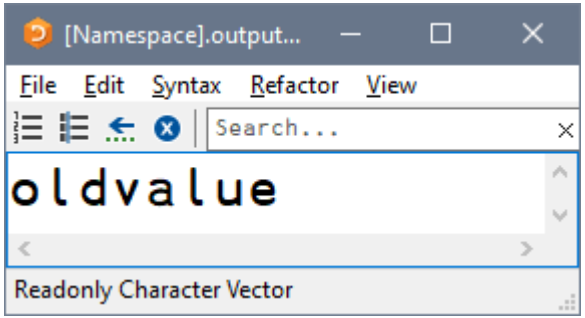


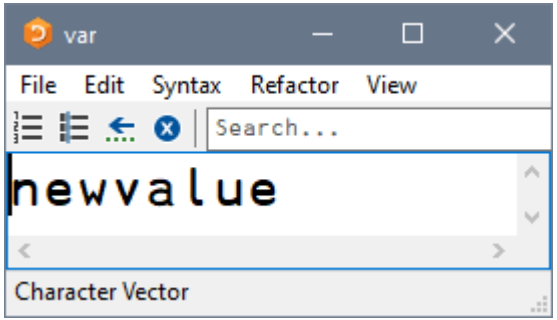

```
var←'oldvalue'
]view var
(3↑var)←'new'
)ed var
```

# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item

```
var←'oldvalue'
]view var
(3↑var)←'new'
)ed var
```



# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item
- Settings: `]Box -view=`



# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item
- Settings: `]Box -view=`
  `]box on -v=max`



# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item
- Settings: `]Box -view=`

```
      ]box on -v=max
Was OFF -view=min
```



# ]OUTPUT.View

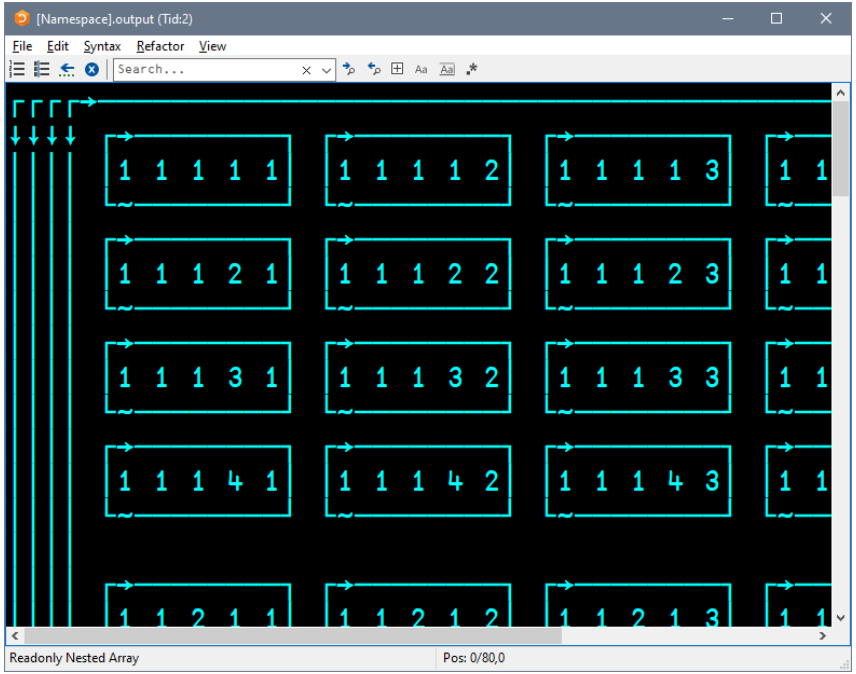
- Classic window mode
- Inspect long output
- Snapshot an APL item
- Settings: ]Box -view=
  ]box on -v=max
Was OFF -view=min
  ]view ⍳⍳5



# ]OUTPUT.View

- Classic window mode
- Inspect long output
- Snapshot an APL item
- Settings: `]Box -view=`

```
      ]box on -v=max
Was OFF -view=min
      ]view ⍳⍳5
```



```
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
```

+ Ideas for the Future



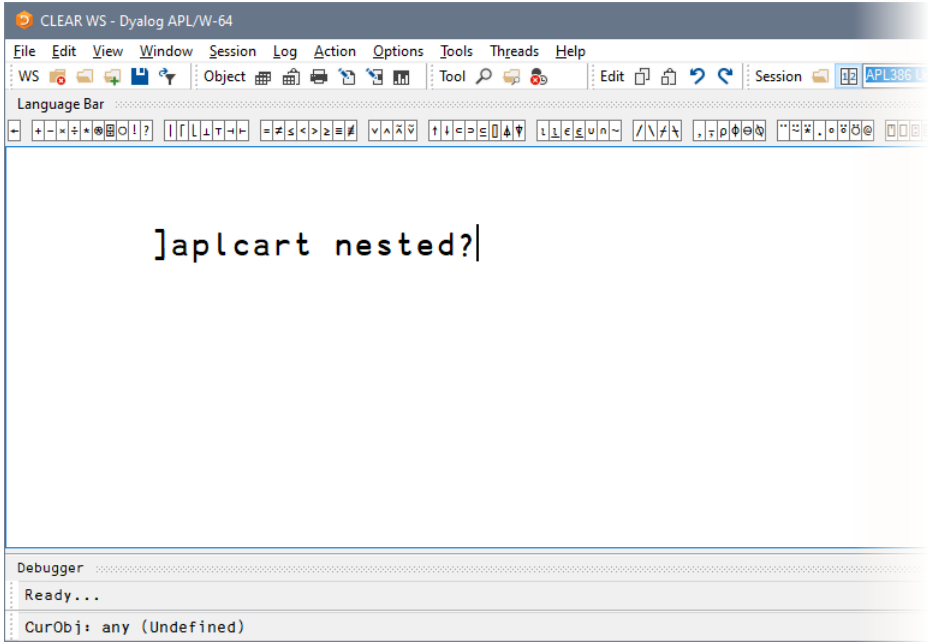
# ]TOOLS.APLCart

- Display in session



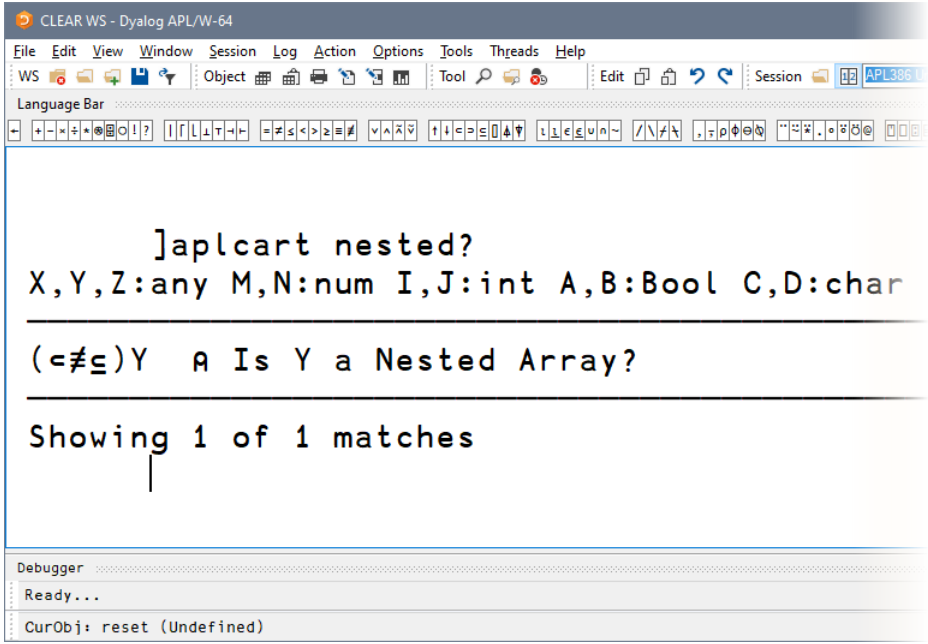
# ]TOOLS.APLCart

- Display in session



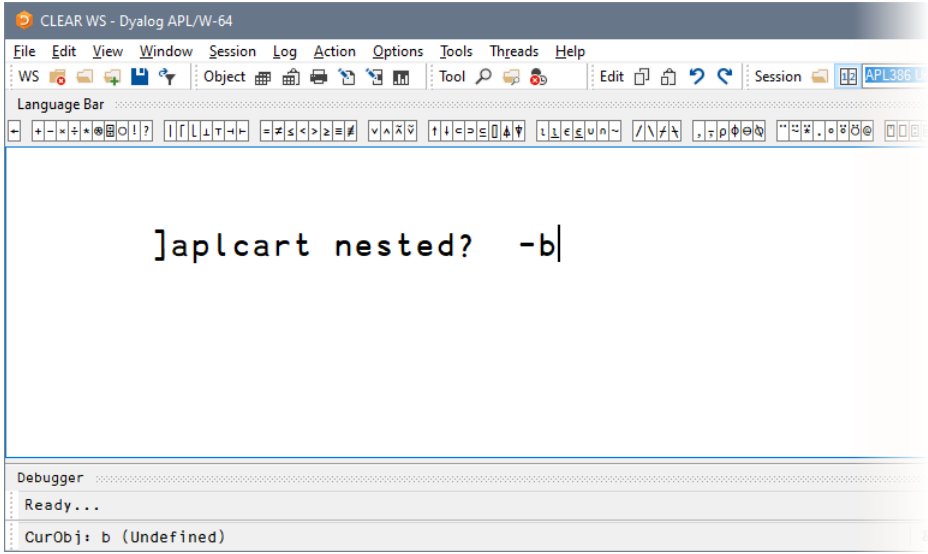
# ]TOOLS.APLCart

- Display in session



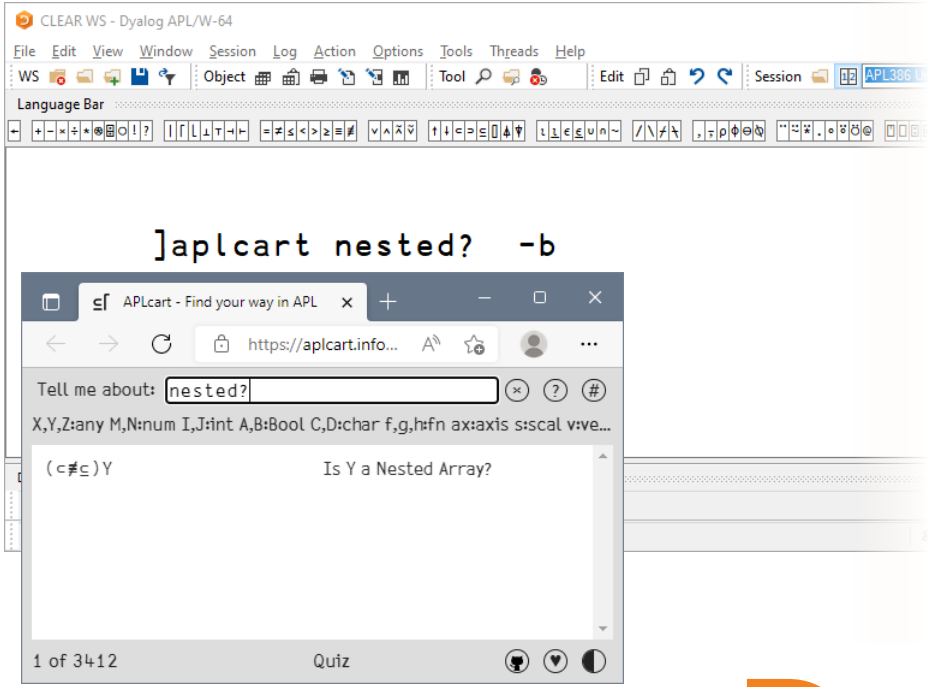
# ]TOOLS.APLCart

- Display in session
- Display in browser



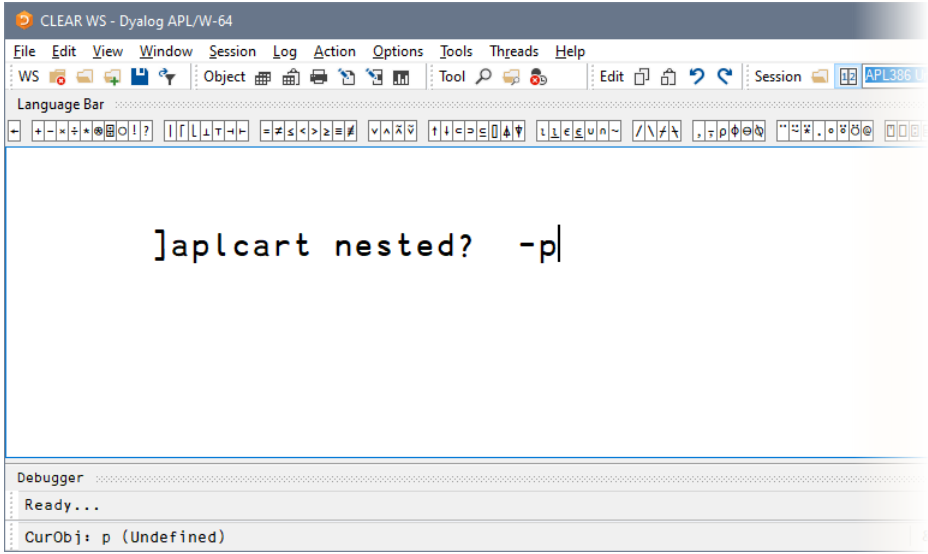
# ]TOOLS.APLCart

- Display in session
- Display in browser



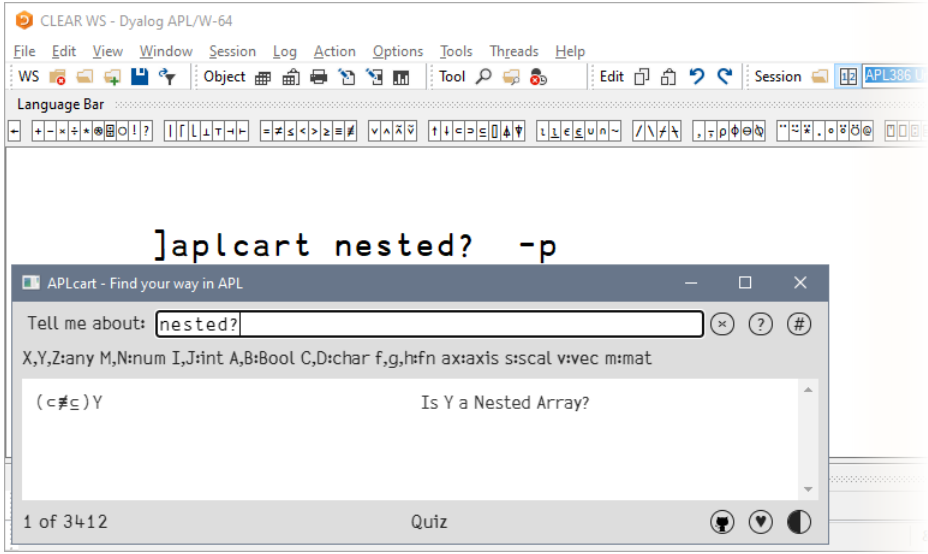
# ]TOOLS.APLCart

- Display in session
- Display in browser
- Display in popup



# ]TOOLS.APLCart

- Display in session
- Display in browser
- Display in popup



```
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
```

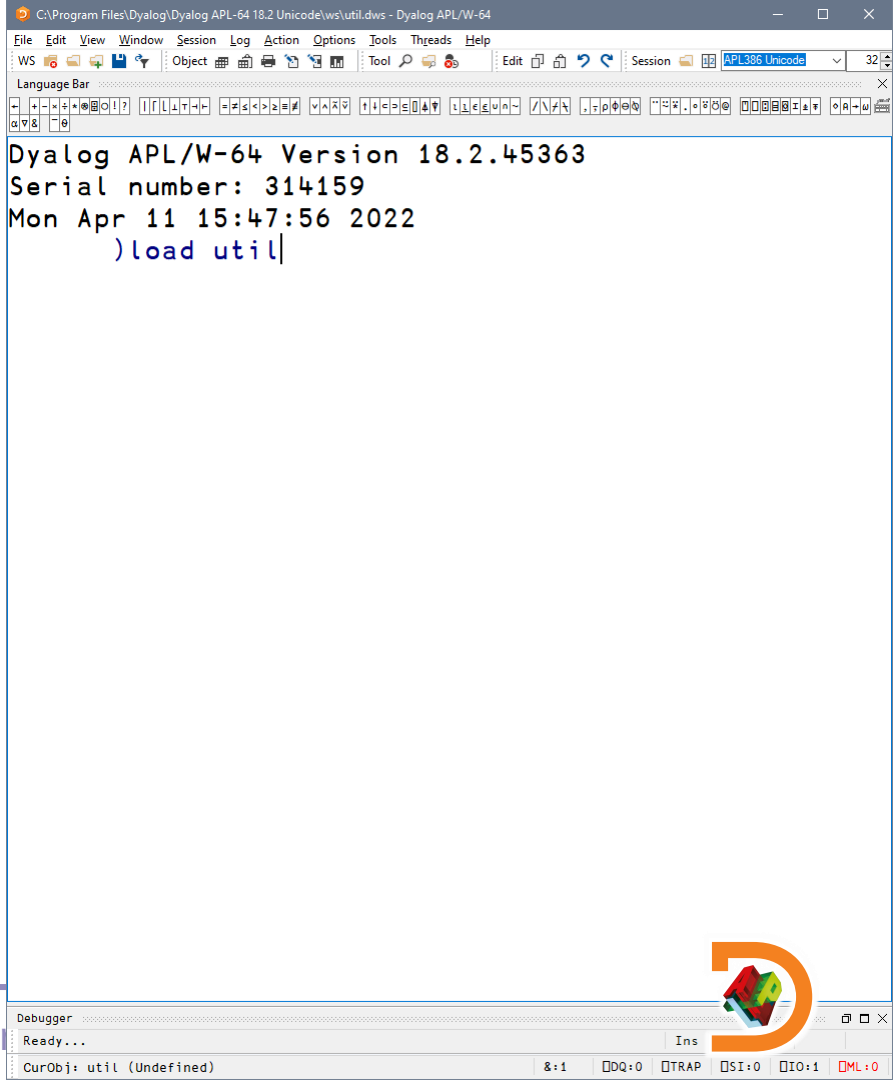
+ Ideas for the Future



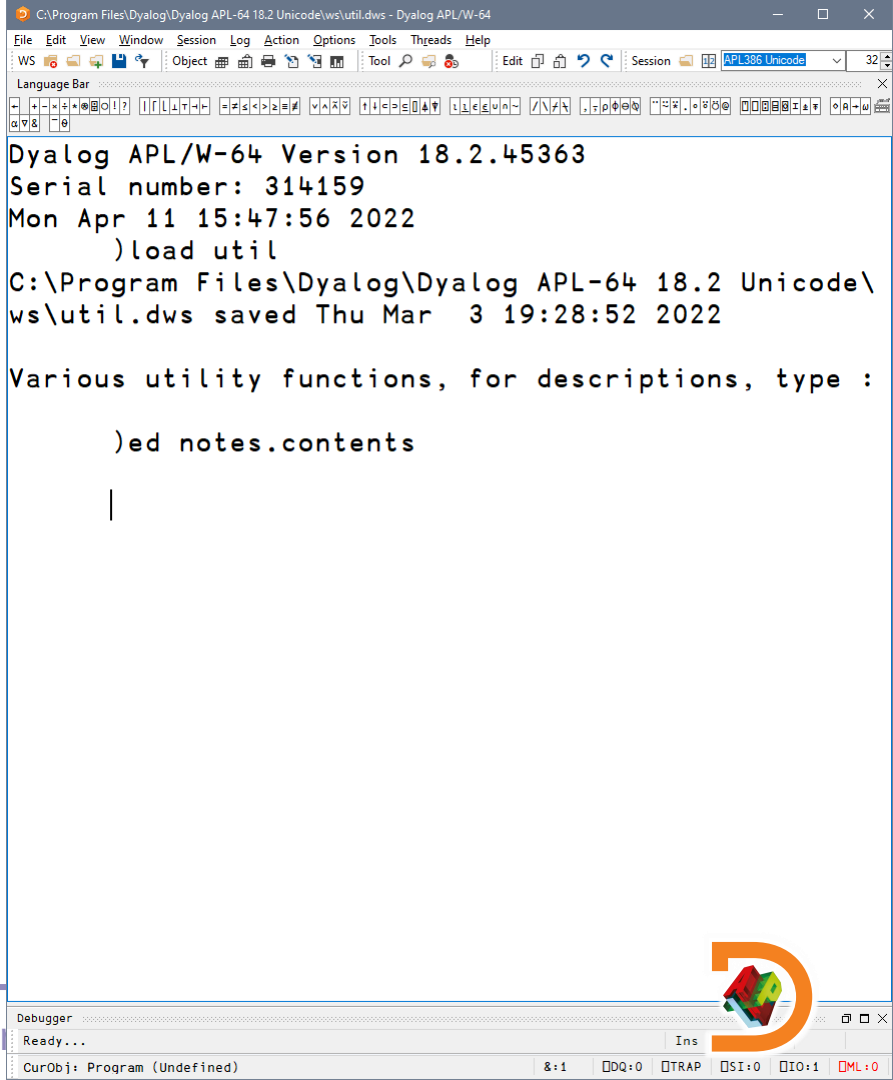
# ]WS.Names



# ]WS.Names

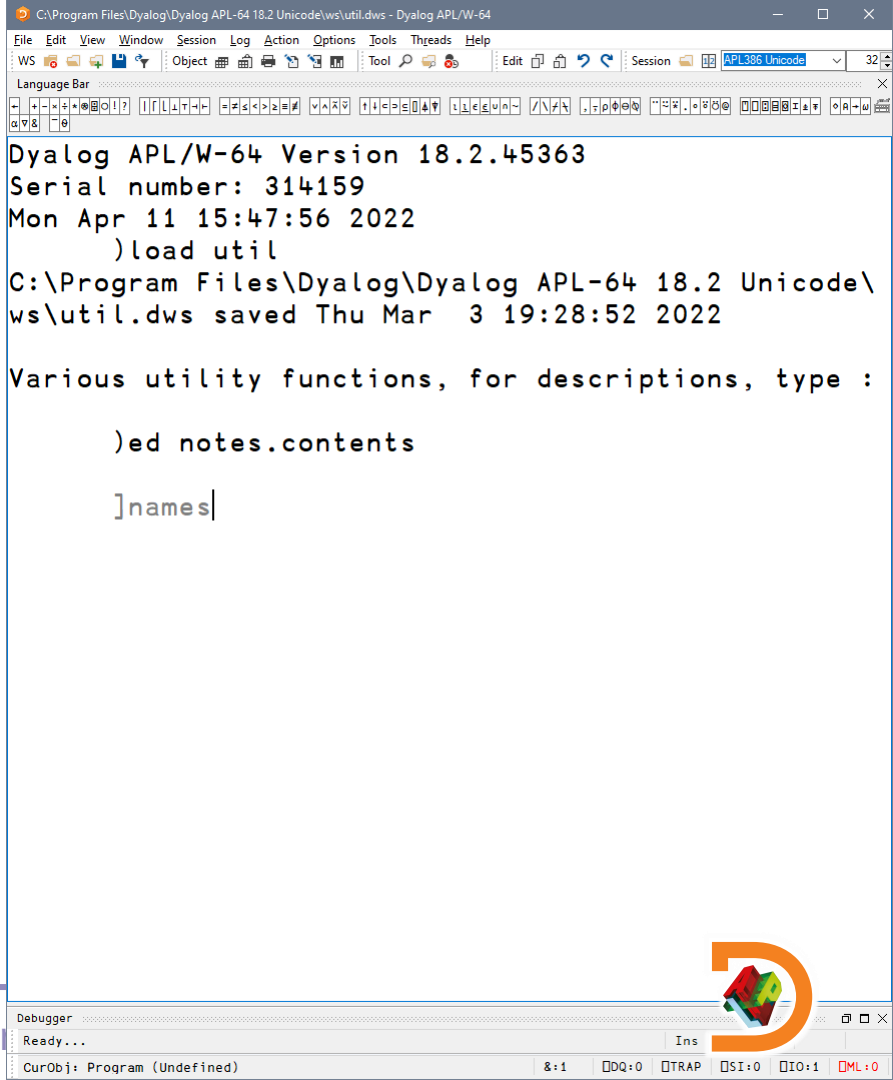

C:\Program Files\Dyalog\Dyalog APL-64 18.2 Unicode\ws\util.dws - Dyalog APL/W-64
File Edit View Window Session Log Action Options Tools Threads Help
Language Bar
Dyalog APL/W-64 Version 18.2.45363
Serial number: 314159
Mon Apr 11 15:47:56 2022
)load util
Debugger
Ready...
Ins
CurObj: util (Undefined)

# ]WS.Names


C:\Program Files\Dyalog\Dyalog APL-64 18.2 Unicode\ws\util.dws - Dyalog APL/W-64
File Edit View Window Session Log Action Options Tools Threads Help
Language Bar
Dyalog APL/W-64 Version 18.2.45363
Serial number: 314159
Mon Apr 11 15:47:56 2022
)load util
C:\Program Files\Dyalog\Dyalog APL-64 18.2 Unicode\ws\util.dws saved Thu Mar 3 19:28:52 2022
Various utility functions, for descriptions, type :
)ed notes.contents
Debugger
Ready...
Ins
CurObj: Program (Undefined)

# ]WS.Names

- List all



# ]WS.Names

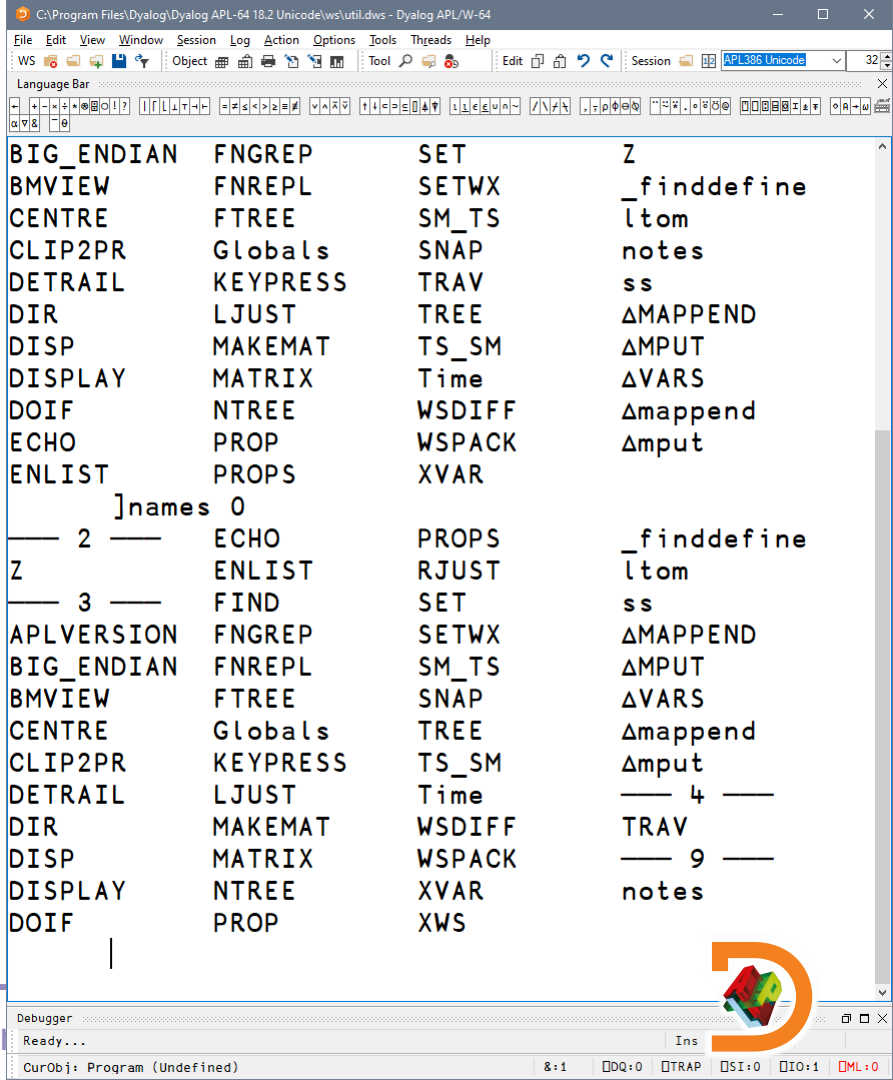
- List all

# ]WS.Names

- List all
- Group by nameclass

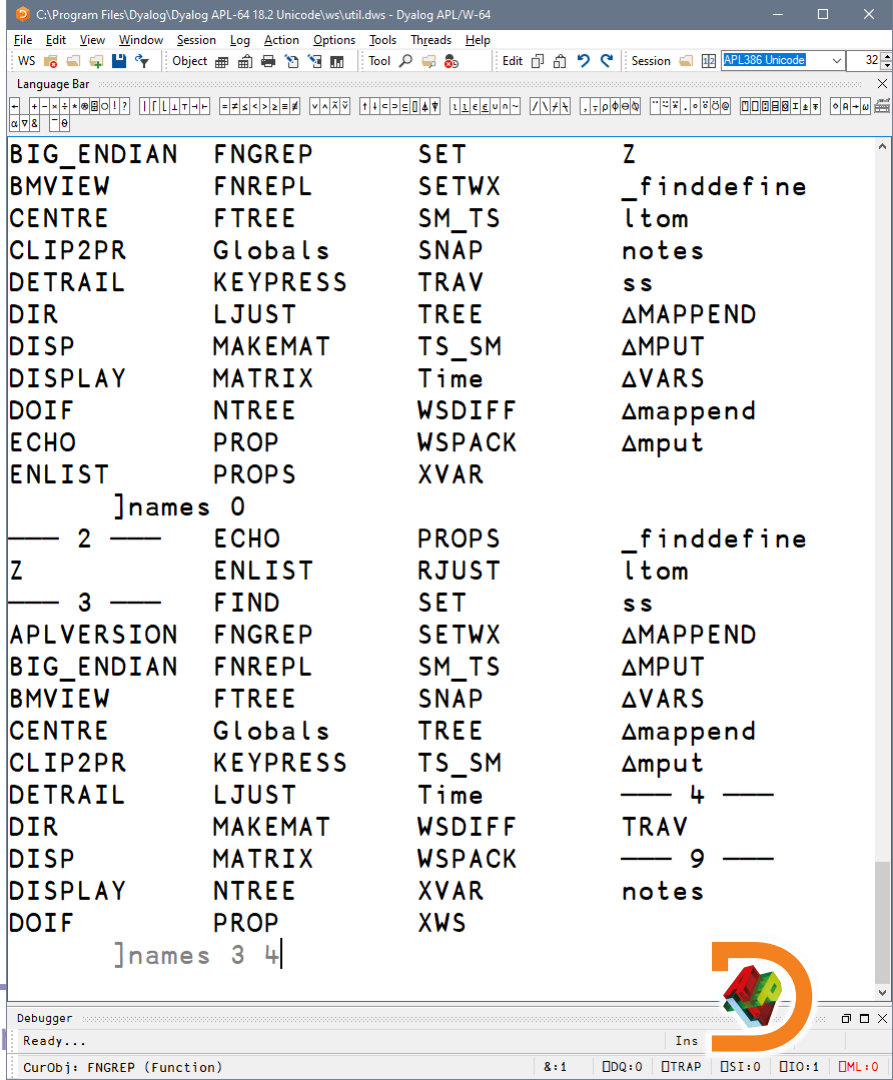
# ]WS.Names

- List all
- Group by nameclass



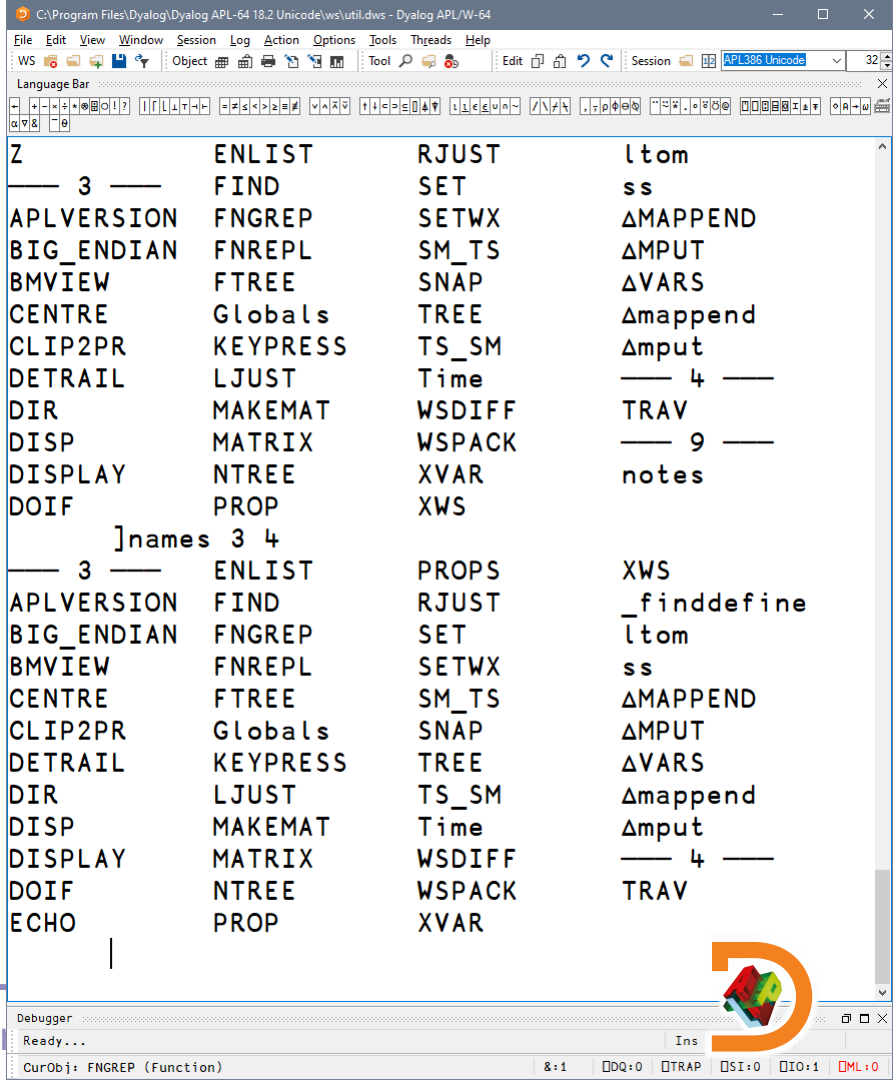
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass



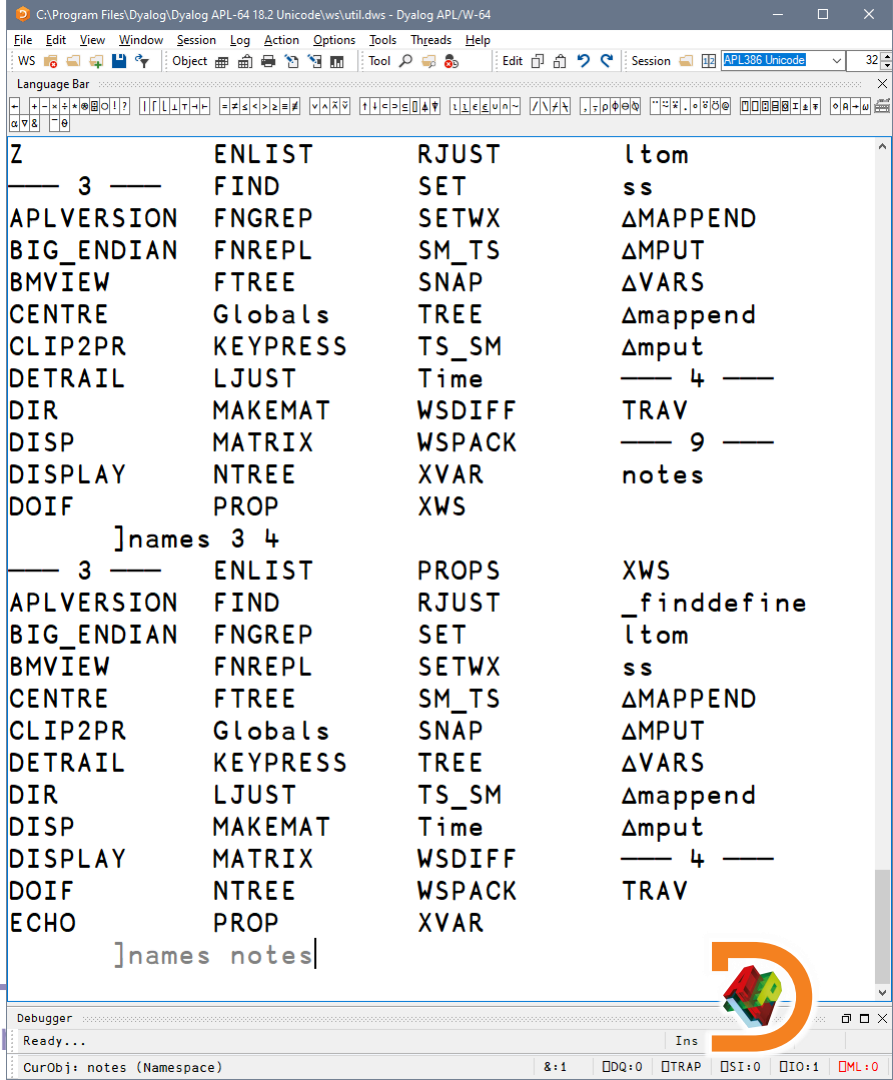
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass



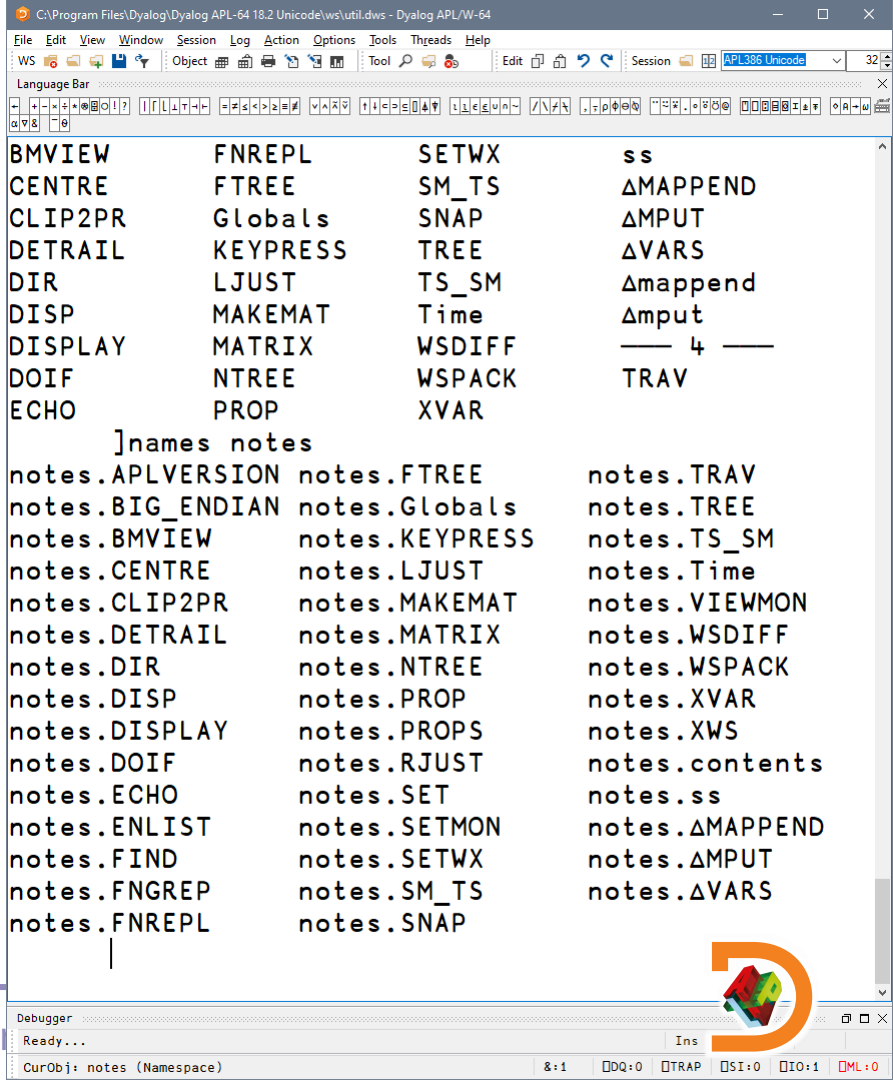
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace



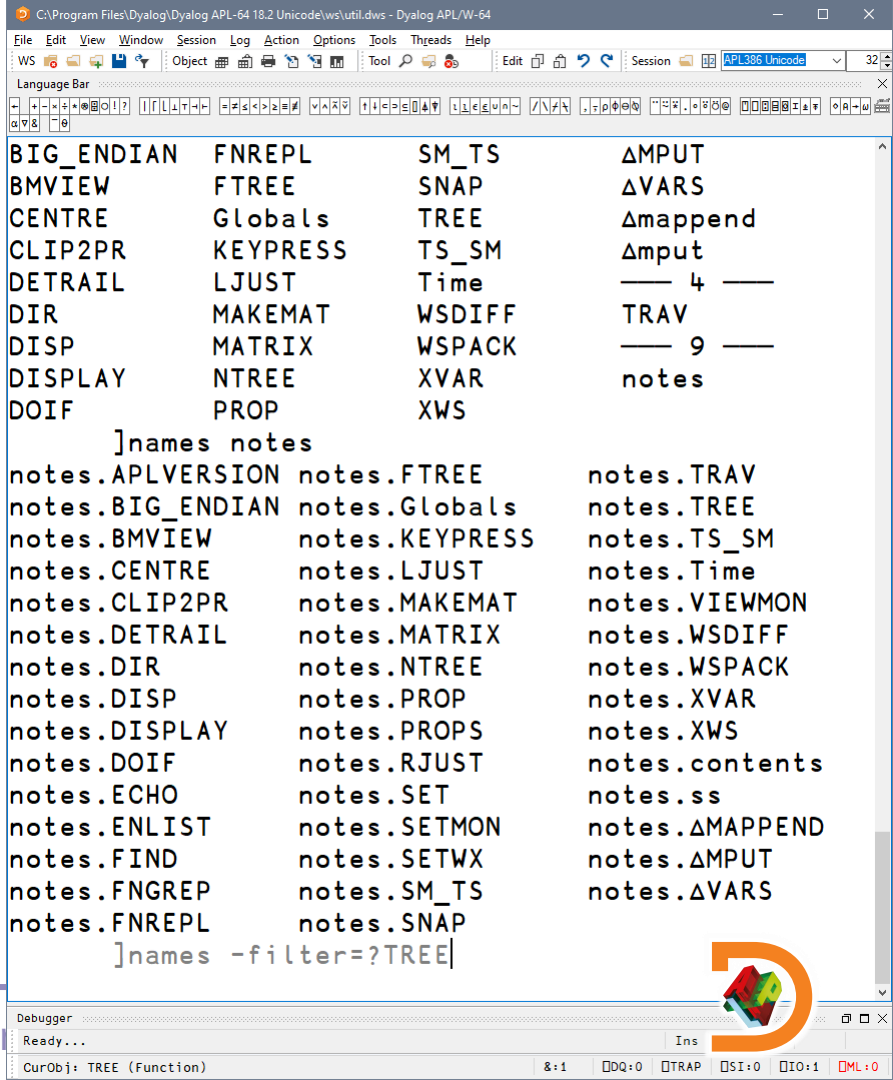
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace



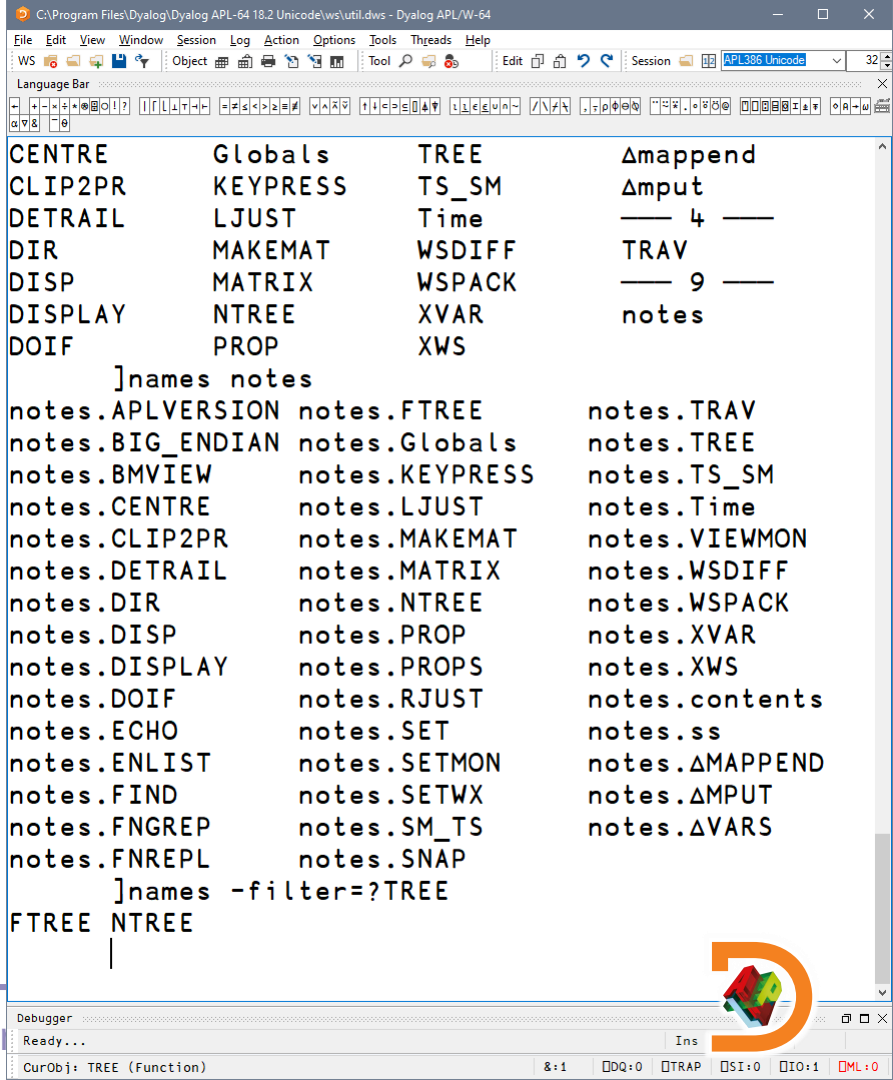
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards



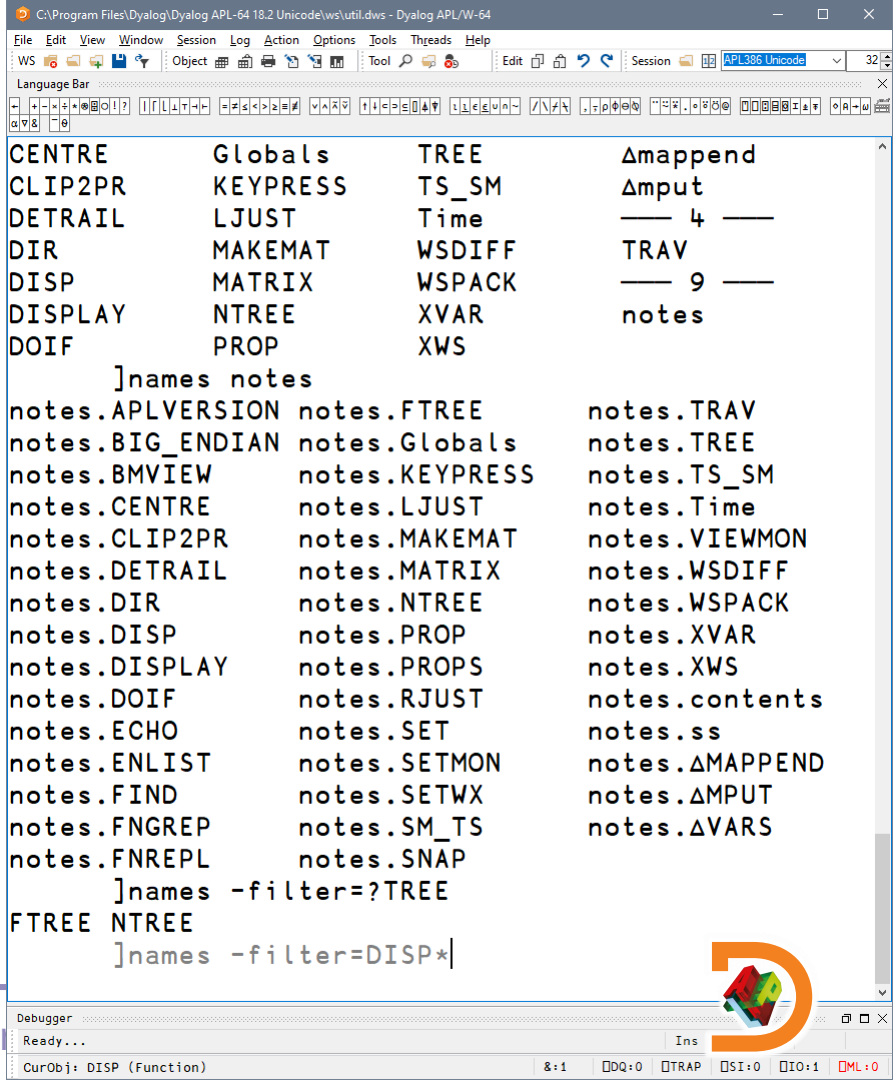
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards



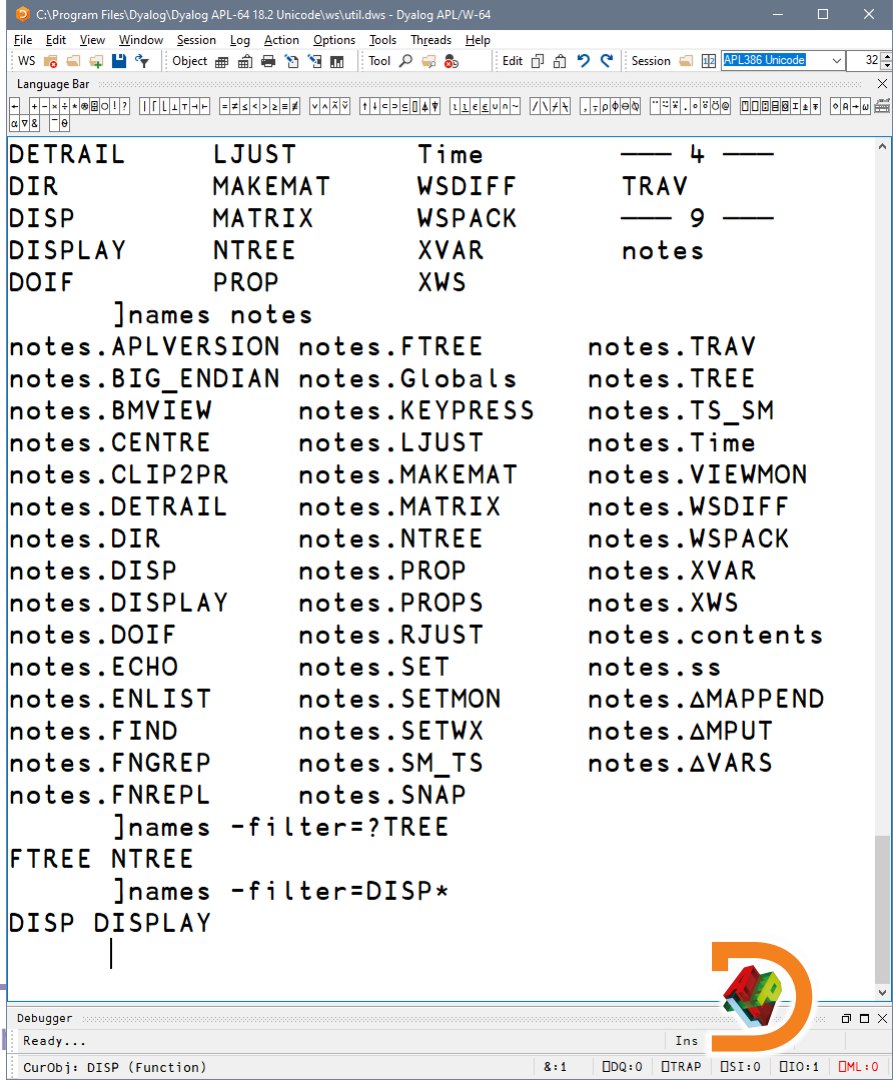
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards



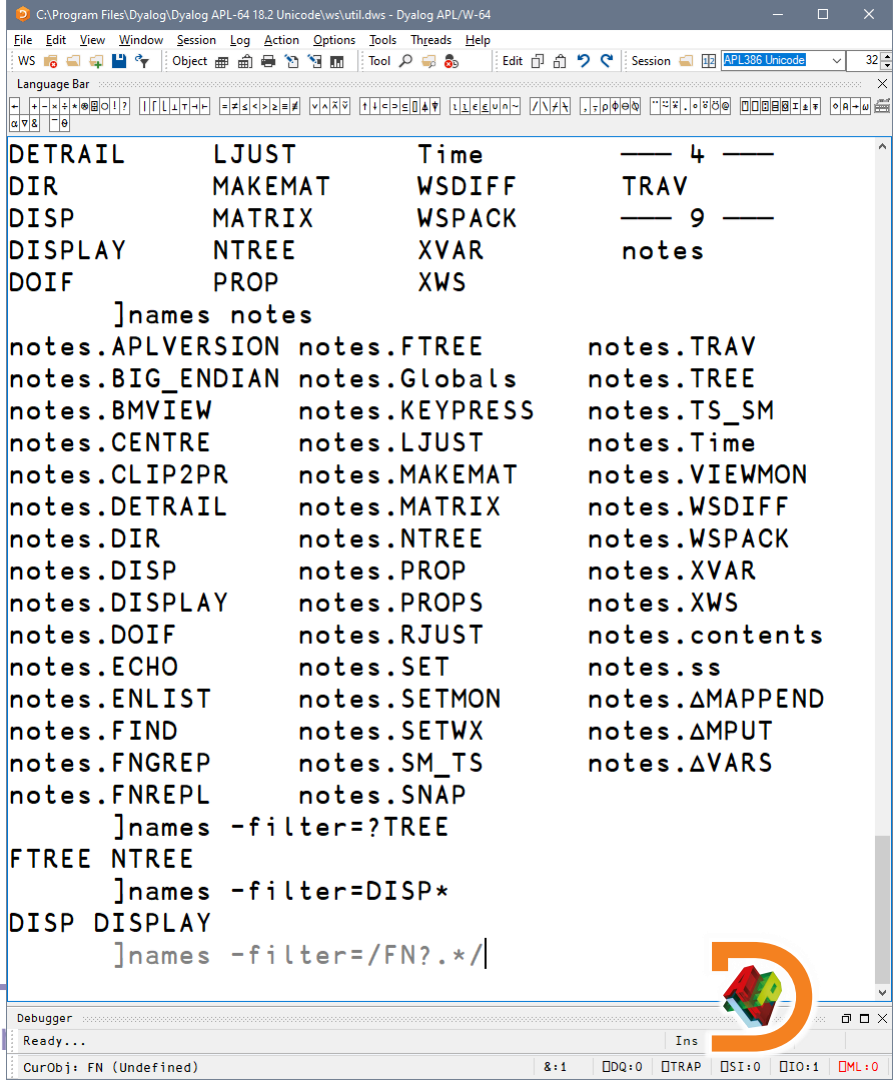
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards



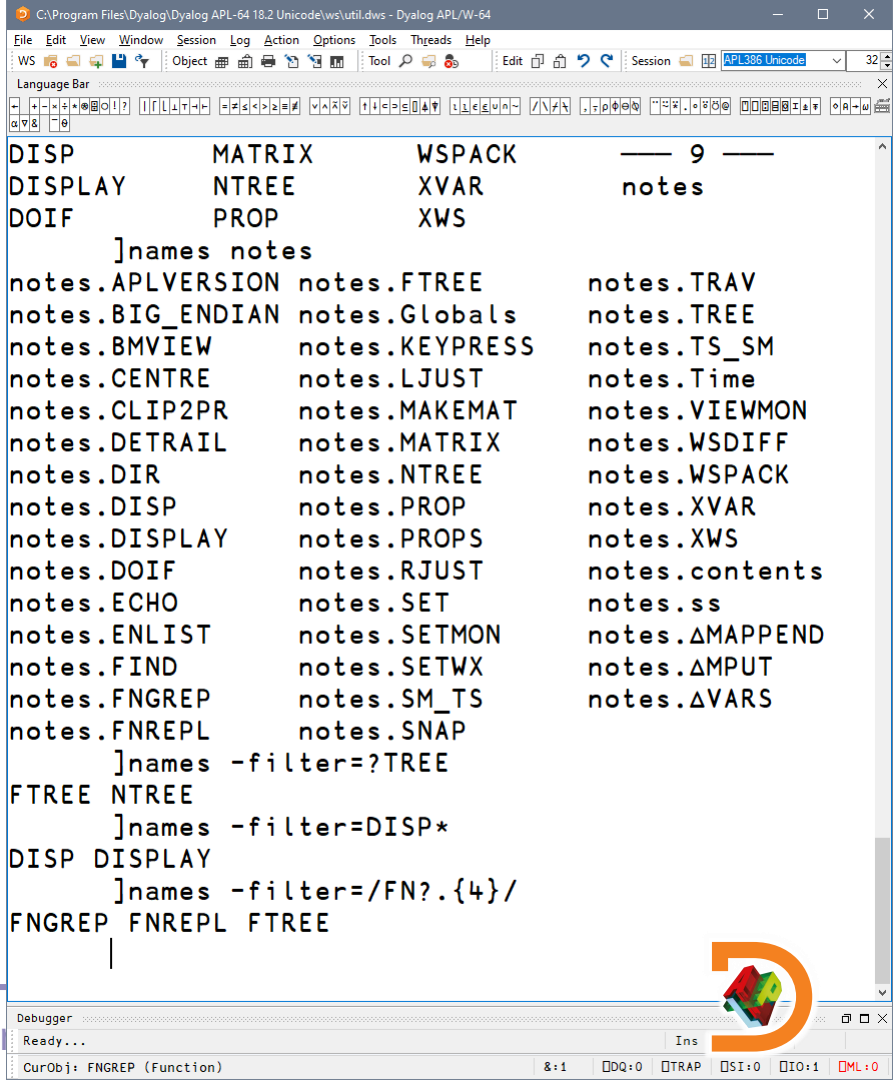
# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards
- Filter using regex



# ]WS.Names

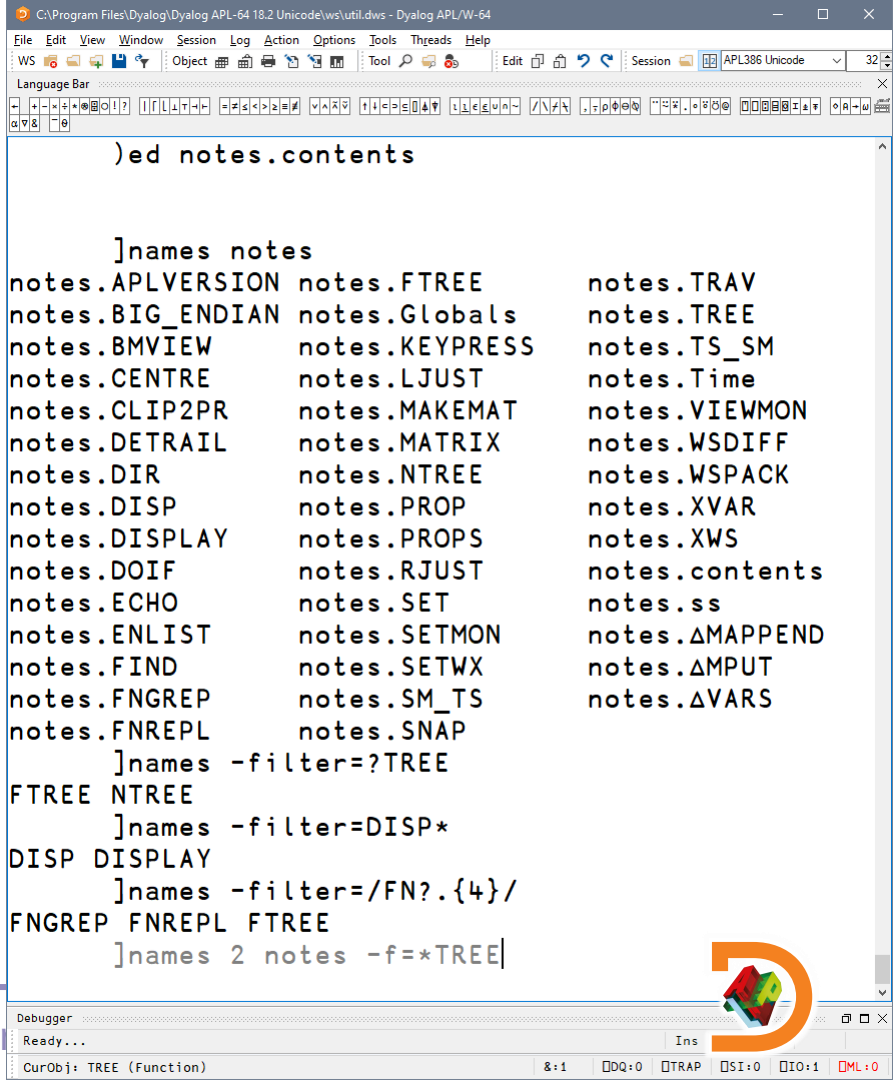
- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards
- Filter using regex



# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards
- Filter using regex

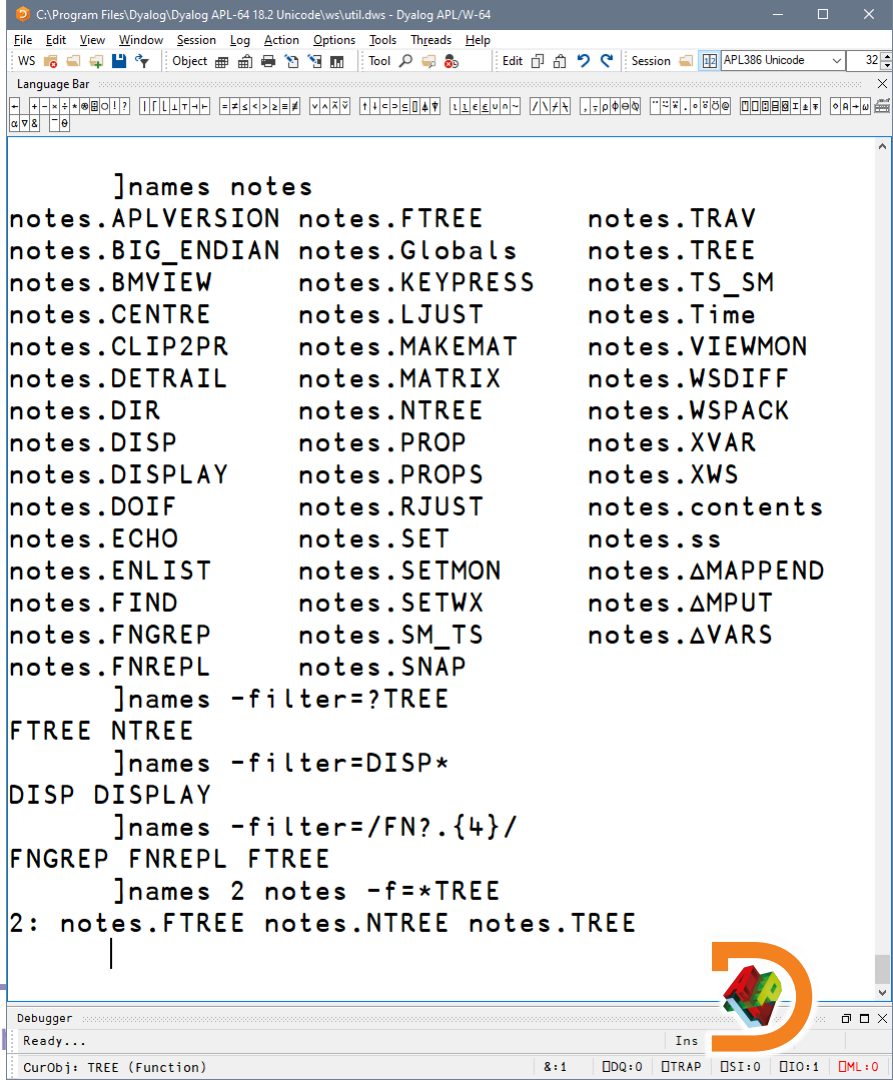
Combine all of these!



# ]WS.Names

- List all
- Group by nameclass
- Filter by nameclass
- Specific namespace
- Filter using wildcards
- Filter using regex

Combine all of these!



```
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
```

+ Ideas for the Future



# ]OUTPUT.Repr

]repr *expression*



# ]OUTPUT.Repr

`]repr` *expression* `-format=APL` single-line APL expression



# ]OUTPUT.Repr

]repr *expression* -format=APL — single-line APL expression

APLAN — APL Array Notation



# ]OUTPUT.Repr

]repr *expression* -format=APL — single-line APL expression

APLAN — APL Array Notation

JS — JavaScript for inclusion in code



# ]OUTPUT.Repr

]repr *expression* -format=APL — single-line APL expression

APLAN — APL Array Notation

JS — JavaScript for inclusion in code

JSON — JavaScript Object Notation



# ]OUTPUT.Repr

]repr *expression* -format=

| | |
|---|---|
| APL | single-line APL expression |
| APLAN | APL Array Notation |
| JS | JavaScript for inclusion in code |
| JSON | JavaScript Object Notation |
| XML | Extensible Markup Language |



# ]OUTPUT.Repr

```
]repr expression -format=APL
                         APLAN
                         JS
                         JSON
                         XML
                         CSV
```

| | |
|---|---|
| APL | single-line APL expression |
| APLAN | APL Array Notation |
| JS | JavaScript for inclusion in code |
| JSON | JavaScript Object Notation |
| XML | Extensible Markup Language |
| CSV | Comma-Separated Values |

,

# ]OUTPUT.Repr

```
]repr expression -format=APL
                         APLAN
                         JS
                         JSON
                         XML
                         CSV
                         SSV
```

| | |
|---|---|
| single-line APL expression | |
| APL Array Notation | |
| JavaScript for inclusion in code | |
| JavaScript Object Notation | |
| Extensible Markup Language | |
| Comma-Separated Values | , |
| Semicolon-Separated Values | ; |



# ]OUTPUT.Repr

]repr *expression* -format=

| | |
|---|---|
| APL | single-line APL expression |
| APLAN | APL Array Notation |
| JS | JavaScript for inclusion in code |
| JSON | JavaScript Object Notation |
| XML | Extensible Markup Language |
| CSV | Comma-Separated Values , |
| SSV | Semicolon-Separated Values ; |
| PSV | Pipe-Separated Values \| |



# ]OUTPUT.Repr

]repr *expression* -format=

| | | |
|---|---|---|
| APL | single-line APL expression | |
| APLAN | APL Array Notation | |
| JS | JavaScript for inclusion in code | |
| JSON | JavaScript Object Notation | |
| XML | Extensible Markup Language | |
| CSV | Comma-Separated Values | , |
| SSV | Semicolon-Separated Values | ; |
| PSV | Pipe-Separated Values | \| |
| TSV | Tab-Separated Values | ⇆ |

# ]OUTPUT.Repr

]repr *expression* -format=*format*



# ]OUTPUT.Repr

]repr *expression* -format=*format* -outfile=*filename*



# ]OUTPUT.Repr

```
      ]repr ⍳1 2
```



# ]OUTPUT.Repr

```
      ]repr ⍳1 2
(1 2⍴(2⍴1) (1 2))
```



# ]OUTPUT.Repr

```
      ]repr ⍳1 2
(1 2⍴(2⍴1) (1 2))

      ]repr +⌿÷1⌈≢
```



# ]OUTPUT.Repr

```
      ]repr ⍳1 2
(1 2⍴(2⍴1) (1 2))

      ]repr +⌿÷1⌈≢
(+⌿)÷(1⌈≢)
```



# ]OUTPUT.Repr

```
      ]repr ⍳1 2
(1 2⍴(2⍴1) (1 2))

      ]repr +⌿÷1⌈≢
(+⌿)÷(1⌈≢)

      Merge←⍳⍤≢⍤⊣⌷⊂⍤⍳⍤0⌷⍤⍳0 99
```



# ]OUTPUT.Repr

```
      ]repr ⍳1 2
(1 2⍴(2⍴1) (1 2))

      ]repr +⌿÷1⌈≢
(+⌿)÷(1⌈≢)

      Merge←⍳⍤≢⍤⊣⌷⊂⍤⍳⍤0⌷⍤0 99
      ]repr Merge
```



# ]OUTPUT.Repr

```
      ]repr ⍳1 2
(1 2⍴(2⍴1) (1 2))

      ]repr +⌿÷1⌈≢
(+⌿)÷(1⌈≢)

      Merge←⍳⍤≢⍤⊣⍤⍬∘⊃⍤0⍬⍤0 99
      ]repr Merge
 Merge←((⍳⍤≢)⍤⊣)((⍬∘⊃)⍤0)(⍬⍤0 99)
```



# ]OUTPUT.Repr

```
]repr ⍳1 2 -f=aplan
```



# ]OUTPUT.Repr

```
      ]repr ⍳1 2 -f=aplan
[
 (
  1 1
  1 2
 )
]
```



# ]OUTPUT.Repr

```
Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍺⍺ ⍵}
```



# ]OUTPUT.Repr

```
Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍺⍺ ⍵}
]repr Hook
```



# ]OUTPUT.Repr

```
      Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍺⍺ ⍵}
      ]repr Hook
⎕FX(,⊂' Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍺⍺ ⍵}')
```



# ]OUTPUT.Repr

```
      Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍨⍺⍺ ⍵}
      ]repr Hook
⎕FX(,⊂' Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍨⍺⍺ ⍵}')
      ]repr Hook -f=aplan
```



# ]OUTPUT.Repr

```
      Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍣⍺⍺ ⍵}
      ]repr Hook
⎕FX(,⊂' Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍣⍺⍺ ⍵}')
      ]repr Hook -f=aplan
 Hook←{⍺←⍵ ⋄ ⍵ ⍵⍵⍨⍣⍺⍺ ⍵}
```



```
]OUTPUT.Repr

      'ns.sub'⍞NS⍬
```



# ]OUTPUT.Repr

```
      'ns.sub'⎕NS⍬
      ns.sub.val←,42
```



# ]OUTPUT.Repr

```
'ns.sub'⎕NS⍬
ns.sub.val←,42
]repr ns -f=aplan
```



# ]OUTPUT.Repr

```
      'ns.sub'⎕NS⍬
      ns.sub.val←,42
      ]repr ns -f=aplan
(
 sub:(
  val:(42⋄)
 )
)
```



# ]OUTPUT.Repr

```
      'ns.sub'⎕NS⍬
      ns.sub.val←,42
      ]repr ns -f=js
```



# ]OUTPUT.Repr

```
      'ns.sub'⎕NS⍬
      ns.sub.val←,42
      ]repr ns -f=js
{
 sub: {
  val: [42],
 },
}
```



# ]OUTPUT.Repr

```
      'ns.sub'⎕NS⍬
      ns.sub.val←,42
      ]repr ns -f=js                    ]repr ns -f=json
{
 sub: {
  val: [42],
 },
}
```



# ]OUTPUT.Repr

```
      'ns.sub'⎕NS⍬
      ns.sub.val←,42
      ]repr ns -f=js
{
 sub: {
  val: [42],
 },
}
```

```
      ]repr ns -f=json
{
 "sub": {
  "val": [42]
 }
}
```



# ]OUTPUT.Repr

```
      'ns.sub'⎕NS⍬
      ns.sub.val←,42
      ]repr ns -f=js
{
 sub: {
  val: [42],
 },
}
```

```
      ]repr ns -f=json
{
 "sub": {
  "val": [42]
 }
}
```

# ]OUTPUT.Repr

```
      ]repr 3 3⍴⍳9 -f=xml
<array>
  <shape>3 3</shape>
  <list>1 2 3 4 5 6 7 8 9</list>
</array>
```



# ]OUTPUT.Repr

```
      ]repr 3 3⍴⍳9 -f=csv
1,2,3
4,5,6
7,8,9
```



# ]OUTPUT.Repr

```
      ]repr 3 3⍴⍳9 -f=ssv
1;2;3
4;5;6
7;8;9
```



```
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
```

+ Ideas for the Future



# ]EXPERIMENTAL.Get

*“get pretty much anything from pretty much anywhere”*



# ]EXPERIMENTAL.Get

***"get pretty much anything from pretty much anywhere"***

- One-stop utility for quickly getting bringing resources in.



# ]EXPERIMENTAL.Get

***"get pretty much anything from pretty much anywhere"***

- One-stop utility for quickly getting bringing resources in.
- Do not use at run time, as exact results may vary.



# ]EXPERIMENTAL.Get

**Syntax**

```
]get something
```



# ]EXPERIMENTAL.Get

**Syntax**

```
      ]get something
#.something
```



# ]EXPERIMENTAL.Get

## Plain name heuristics



# ]EXPERIMENTAL.Get

**Plain name heuristics**

```
      ]get dfns                    workspace from WS path
#.dfns
```



# ]EXPERIMENTAL.Get

**Plain name heuristics**

```
      ]get dfns              workspace from WS path
#.dfns
      ]get HttpCommand       source file from SALT path
#.HttpCommand
```



# ]EXPERIMENTAL.Get

**Plain name heuristics**

```
      ]get dfns              workspace from WS path
#.dfns
      ]get HttpCommand       source file from SALT path
#.HttpCommand
      ]get tools             directory relative to current
#.tools
```



# ]EXPERIMENTAL.Get

## File extension detection



# ]EXPERIMENTAL.Get

## File extension detection

```
      ]get MyFn.aplf      Link-type source files
#.MyFn
```



# ]EXPERIMENTAL.Get

## File extension detection

```
      ]get MyFn.aplf      Link-type source files
#.MyFn
      ]get myVar.apla     APL Array Notation files
#.myVar
```



# ]EXPERIMENTAL.Get

## File extension detection

```
      ]get MyFn.aplf      Link-type source files
#.MyFn
      ]get myVar.apla     APL Array Notation files
#.myVar
      ]get text.charmat   Acre-type source files
#.text
```



# ]EXPERIMENTAL.Get

## Data format conversion



# ]EXPERIMENTAL.Get

## Data format conversion

```
      ]get obj.json          JS, JSON, JSON5, config files
#.obj
```



# ]EXPERIMENTAL.Get

**Data format conversion**

```
      ]get obj.json        JS, JSON, JSON5, config files
#.obj
      ]get table.csv       Comma Separated Values
#.table
```



# ]EXPERIMENTAL.Get

**Data format conversion**

```
      ]get obj.json        JS, JSON, JSON5, config files
#.obj
      ]get table.csv       Comma Separated Values
#.table
      ]get data.xml        eXtensible Markup Language
#.data
```

# ]EXPERIMENTAL.Get

## Retrieval



# ]EXPERIMENTAL.Get

## Retrieval

```
      ]get https://example.com/index.html
#.index
```



# ]EXPERIMENTAL.Get

## Retrieval

```
      ]get https://example.com/index.html
#.index
      ]get https://github.com/abrudz/Kbd
#.Kbd
```



# ]EXPERIMENTAL.Get

## Protocols



# ]EXPERIMENTAL.Get

## Protocols

```
      ]get https://github.com/abrudz/Kbd
#.Kbd
```



# ]EXPERIMENTAL.Get

## Protocols

```
      ]get https://github.com/abrudz/Kbd
#.Kbd
      ]get ftp://ftp.software.ibm.com/software
#.foo
```



# ]EXPERIMENTAL.Get

## Protocols

```
      ]get https://github.com/abrudz/Kbd
#.Kbd
      ]get ftp://ftp.software.ibm.com/software
#.foo
      ]get github.com/abrudz/Kbd
#.Kbd
```

# ]EXPERIMENTAL.Get

## Unpacking



# ]EXPERIMENTAL.Get

## Unpacking

```
      ]get main.zip
#.main
```



# ]EXPERIMENTAL.Get

## Unpacking

```
      ]get main.zip
#.main
      ]get github.com/user/repo/tree/main/sub
#.sub
```



# ]EXPERIMENTAL.Get

## Unpacking



# ]EXPERIMENTAL.Get

## Unpacking

```
      ]get dfns -only=dxb,dab
#.dfns
```



# ]EXPERIMENTAL.Get

## Unpacking

```
      ]get dfns -only=dxb,dab
#.dfns
      ]map
#
·    dfns
·    ·    ∇ dab dxb
```



# ]EXPERIMENTAL.Get

## Unpacking



# ]EXPERIMENTAL.Get

**Unpacking**

```
      )clear
clear ws
```



# ]EXPERIMENTAL.Get

## Unpacking

```
      )clear
clear ws
      ]get dfns -only=dxb,dab -unpack
#.dxb #.dab
```



# ]EXPERIMENTAL.Get

## Unpacking

```
      )clear
clear ws
      ]get dfns -only=dxb,dab -unpack
#.dxb #.dab
      ]map
#
·    ∇ dab dxb
```



# ]EXPERIMENTAL.Get

## Options



# ]EXPERIMENTAL.Get

**Options**

```
      ]get obj.json:txt
#.obj
```



# ]EXPERIMENTAL.Get

**Options**

```
      ]get obj.json:txt
#.obj
      10↑obj
{"myKey":1
```



# ]EXPERIMENTAL.Get

**Options**

```
      ]get obj.json:txt
#.obj
      10↑obj
{"myKey":1
      ]get thing -target=⎕SE.there
```



# ]EXPERIMENTAL.Get

**Options**

```
      ]get obj.json:txt
#.obj
      10↑obj
{"myKey":1
      ]get thing -target=⎕SE.there
#.SE.there.thing
```



# ]EXPERIMENTAL.Get

## Options



# ]EXPERIMENTAL.Get

**Options**

```
      ]get /tmp/repo -sync
#.repo
```



# ]EXPERIMENTAL.Get

**Options**

```
      ]get /tmp/repo -sync
#.repo
      ]link.status
```



# ]EXPERIMENTAL.Get

**Options**

```
      ]get /tmp/repo -sync
#.repo
      ]link.status
 Namespace       Directory    Files
 #.repo      ↔   c:/tmp/repo      14
```



# Overview

```
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
```

+ Ideas for the Future



# Ideas for the Future

Current issues:



# Ideas for the Future

Current issues:

- SALT dependency



# Ideas for the Future

Current issues:

- SALT dependency
- Performance



# Ideas for the Future

Current issues:

- SALT dependency
- Performance
- Cache file invalidation



# Ideas for the Future

Current issues:

- SALT dependency
- Performance
- Cache file invalidation
- Abuse for utility functions



# Ideas for the Future

Current issues:

- SALT dependency
- Performance
- Cache file invalidation
- Abuse for utility functions
- Inflexibility



# Ideas for the Future

Current issues:

- SALT dependency
- Performance
- Cache file invalidation
- Abuse for utility functions
- Inflexibility
- Old philosophy

# Ideas for the Future

My ideas:



# Ideas for the Future

My ideas:

- Based on SessionStartup



# Ideas for the Future

My ideas:

- Based on SessionStartup
- Directories ⇔ namespaces ⇔ groups



# Ideas for the Future

My ideas:

- Based on SessionStartup
- Directories ⇔ namespaces ⇔ groups
- Any function can declare itself as a user command



# Ideas for the Future

My ideas:

- Based on SessionStartup
- Directories ⇔ namespaces ⇔ groups
- Any function can declare itself as a user command
- Standard format



# Ideas for the Future

My ideas:

- Based on SessionStartup
- Directories ⇔ namespaces ⇔ groups
- Any function can declare itself as a user command
- Standard format

Left argument:
namespace of options including utilities
Right argument:
argument raw/list

# Ideas for the Future

My ideas:

- Based on SessionStartup
- Directories ⇔ namespaces ⇔ groups
- Any function can declare itself as a user command
- Standard format
- Framework searches ⎕SE

Left argument:
namespace of options including utilities
Right argument:
argument raw/list

# Ideas for the Future

Benefits:



# Ideas for the Future

Benefits:

- One API function for every user command



# Ideas for the Future

Benefits:

- One API function for every user command
- Easy to package, distribute, install



# Ideas for the Future

Benefits:

- One API function for every user command
- Easy to package, distribute, install
- Can be part of a larger package



# Ideas for the Future

Benefits:

- One API function for every user command
- Easy to package, distribute, install
- Can be part of a larger package
- Can co-exist with utilities and dependencies



# Ideas for the Future

Benefits:

- One API function for every user command
- Easy to package, distribute, install
- Can be part of a larger package
- Can co-exist with utilities and dependencies
- Stability: Group name required under program control



# Ideas for the Future

Benefits:

- One API function for every user command
- Easy to package, distribute, install
- Can be part of a larger package
- Can co-exist with utilities and dependencies
- Stability: Group name required under program control
- Customisability: Change group by renaming a directory

# Questions?

```
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
```

**+ Ideas for the Future**

*Adám Brudzewsky*
*adam@dyalog.com*