



dyalog.WSEngine

Deployment of the Dyalog Interpreter as OLE-Server for interactive Use

(Microsoft Windows only)

Jürgen Wiedemann
DPC

Dyalog '24
17. September 2024

Agenda

- Introducing DPC and myself
- Starting point and Requirements
- Implementation
- Demo

About DPC (Dittrich & Partner Consulting)

- Founded in 1981
- Acquired from HBA Consulting in 2023
- 10 employees, together with HBA 50
- Located in Germany, customers in Germany, Austria and Switzerland
- Providing consulting services mainly for life insurance companies
- Distributing APL Software licences

About me

- Working for DPC since 1998
- Consultant with focus on APL
- Done a lot of APL migrations
- Managing director since 2019
(second job)

Starting point and Requirements

- Migration of several APL+Win applications
(with all the typical challenges)
- Requirement:
 - special functionality referred to as
“APL+Win as an ActiveX server”
- Possibility to
 - create an instance of APL
 - to use any workspace in it without special preparation

Use cases

- Regression test for lots of calculations in parallel on several local instances
- Integration of APL in other environments, e.g. in Excel (VBA)
- Operation in “Developer mode”:
 - fully functional IDE
 - visible
 - debugging and tracing

Solution

- OLEServer object
 - allows visibility and full access to the session
 - but deploys a specific namespace and only certain functions from it
 - an OLEServer object is generated from a namespace and the entry points to be provided are defined with its ExportedFns property
- Emulation for most of the APL+-Features

Solution - Template

■ Overview “APL+Win as an ActiveX server”

Methods

Call

Exec

SetOrphanTimeout

SysCall

SysCommand

UExec

Description

Call a function and return its value

Execute an expression and return its value

Sets the timeout period for the parent process

Call a system-function and return its value

Execute a system-command

Execute an APL expression in the Unicode-aware client and return a (Variant) value

Properties

SysVariable

Variable

Visible

Return or set the value of a system-variable

Return or set the value of a variable

Return or set the visibility of the APL session

Events

onComAction

onNotify

onSysNotify

Triggered in the APL server when a command is initiated in the client session

Triggered when the APL server's system object invokes the **Notify** method

Triggered when a specified action occurs on the APL server

Solution - Template

- Overview “APL+Win as an ActiveX server”

Methods

Call

Exec

SysCommand

Properties

SysVariable

Variable

Visible

Description

Call a function and return its value

Execute an expression and return its value

Execute a system-command

Return or set the value of a system-variable

Return or set the value of a variable

Return or set the visibility of the APL session

dyalog.WSEngine - Essentials

- Tiny namespace as an OLEServer
- Very few and general methods
- To overcome some limitations that apply to an OLEServer
- The OLEServer namespace only acts as an intermediate layer for dynamic use of any APL resources.

dyalog.WSEngine - Details

SysCommand **A** **□CY into OLE-Session**
objAPL.SysCommand **c** 'LOAD ' ,ws

SysVariable **A** **Get □WA value**
objAPL.SysVariable **c** 'WA'

Variable **A** **Return or set value of a variable**
x **←** objAPL.Variable **c** 'APLvar'
x **←** objAPL.Variable 'XX' (2 2 ρ 4)

Call **A** **Call niladic/monadic/dyadic fn**
x **←** objAPL.Call **c** 'fn_nil'
x **←** objAPL.Call 'fn_mon' 123
x **←** objAPL.Call 'fn_dya' 123 ⁻²

dyalog.WSEngine - Details

Exec

A Execute expressions, return value
`x←objAPL.Exec⊢'NC='myVar''`

Exec_async

(no result)

A Execute asynchr. (add to event queue
`objAPL.Exec_async⊢'OFF'`
`objAPL.Exec_async⊢'longCalc'`

setVisible

A Hide/Show APL-Session
`objAPL.setVisible 0`
`objAPL.setVisible 1`

getObjects

A Return names according to name class
`vars←objAPL.getObjects 2`

Update

A Update WSEngine namespace
`objAPL.Update⊢'E:\Fix.dws'`

Deployment

■ Registration

- with the OLERegister method invoked in `□L X`
- called with a special switch `/RegWSEngine`
- `dyalog.exe WSEngine.dws -MAXWS=256M
RegWSEngine=1 DYALOG_NOPOUPS=1`

■ Invocation

• VBA

```
Dim objAPL As Object  
Set objAPL = CreateObject("dyalog.WSEngine")
```

• Dyalog

```
'objAPL' □WC 'OLEClient' 'dyalog.WSEngine'
```

Examples

From APL

```
'APL' □ WC 'OLEClient' 'dyalog.WSEngine'  
APL.setVisible 1  
x←APL.SysCommand 'LOAD ',ws  
x←APL.Call 'init_appl'  
x←APL.Call 'do_calc' 123  
x←APL.Exec_async 'OFF'  
□ EX 'APL'
```

From Excel (VBA)

```
Set APL = CreateObject("dyalog.WSEngine")  
rc = APL.SysCommand("LOAD " & DWSfile)  
rc = APL.SysVariable("PATH", "#.UTILS")  
rc = APL.Call("init_appl")  
rc = APL.Call("do_calc",123)  
APL.Exec_async (apl_quad & "OFF")  
Set APL = Nothing
```

Demo

- Excel/VBA
 - `CreateObject ("dyalog.WSEngine")`

Demo: Excel/VBA ↔ dyalog.WSEngine

The screenshot displays a Microsoft Excel spreadsheet titled "WSEngine_Demo.xlsm". The spreadsheet content is as follows:

A	B	C	D	E	F	G	H
	dyalog.WSEngine						
	Codebase F:\tmp\WSEngine_Demo.dws						

The spreadsheet includes a "LOAD" button, an "OFF" checkbox, and a "Run Demo" button. A yellow box with the text "Coming soon ..." is also present. The Excel ribbon is visible at the top, and the Windows taskbar is at the bottom.

Demo: Excel/VBA ↔ dyalog.WSEngine

The screenshot displays a Windows desktop environment. On the left, a terminal window titled "Dyalog APL/W-64 Version 18" shows the following output:

```
Dyalog APL/W-64 Version 18
Serial number: 000091
Sat Sep 21 11:12:22 2024
E:\Ergo\WSEngine\WSEngine.

Load Workspace ...
prepare WA...

Starte CY F:\tmp\WSEng
CY beendet
LOAD(CY) erfolgreich du

welcome...
welcome...

application is ready

do some stuff...
```

Below the terminal is a "Debugger" window showing the command "Copying F:\tmp\WSEngine_Demo.dws" and "CurObj:". On the right, an Excel spreadsheet titled "WSEngine_Demo.xlsm" is open. The spreadsheet contains the following content:

- Cell B1: "dyalog.WSEngine"
- Cell B4: "Codebase F:\tmp\WSEngine_Demo.dws"
- Cell B5: "LOAD" button
- Cell B6: "OFF" button
- Cell B7: "Run Demo" button
- Cell B10: "Hello GLASGOW" (highlighted in a yellow box)

The Windows taskbar at the bottom shows the system tray with the time "11:12" and date "21.09.2024".

Demo

- Dyalog v19 using WSEngine v18.2:
 -]demo F:\tmp\WSE

Demo: APL ↔ dyalog.WSEngine

The screenshot displays the Dyalog APL/W-64 environment. The main window, titled 'CLEAR WS - Dyalog APL/W-64', shows a script being executed. The script content is as follows:

```
)clear
clear ws

]demo F:\tmp\WSE

Loaded script F:\tmp\WSE.dyalog
Script Initialized...
  A Demo: Dyalog APL OLE-Server << dyalog.WSEngine >>
]box on
Was ON
]rows on
Was OFF
'objAPL'⌊WC'OLEClient' 'dyalog.WSEngine'  A Create instance
```

The left-hand pane shows the output of the execution:

```
Dyalog APL/W-64 Version 18.2.50027
Serial number: 000091
Sat Sep 21 11:14:44 2024
E:\Ergo\WSEngine\WSEngine.DWS saved Thu
```

The bottom status bar indicates the current object is 'objAPL' and the workspace is 'on (Undefined)'. The system tray shows the date and time as 11:14 on 21.09.2024.

Demo: APL ↔ dyalog.WSEngine

The screenshot shows a Dyalog APL environment with the following components:

- Main Window (CLEAR WS - Dyalog APL/W-64):** Displays the script content:

```
Dyalog APL/W-64 Version 18.2.50027
Serial number: 000091
Sat Sep 21 11:14:44 2024
E:\Ergo\WSEngine\WSEngine.DWS saved Thu
prepare WA...

Starte CY F:\tmp\WSEngine_Demo.dws
CY beendet
LOAD(CY) erfolgreich durchgefuehrt

'objAPL' WC'OLEClient' 'dyalog.WSEngine' A Create instance
objAPL.NL ~2 A Properties
[ AutoBrowse ChildList ClassID ClassName Data Describe Event EventList Handle HelpFile Instanc
objAPL.NL ~3 A Methods
[ Call Exec Exec_async SysCommand SysVariable Update Variable getObject getVariable setVariab
objAPL.SysVariable='WA'
1000933024
objAPL.setVisible 0 A Hide
objAPL.setVisible 1 A Show
objAPL.Update='F:\tmp\Update_2024_1_3.dws' A Backdoor-Update
[-2 Fehler beim Update. Workspace konnte nicht importiert werden
objAPL.SysCommand='LOAD F:\tmp\WSEngine_Demo.dws' A Load/Copy code
[ 0 SysCommand: LOAD Parameter: F:\tmp\WSEngine_Demo.dws
```
- Debugger (bottom left):** Shows the current object and code:

```
Debugger
Copying F:\tmp\WSEngine_Demo.dws
CurObj:
```
- Debugger (bottom right):** Shows the current object and code:

```
Debugger
Ready...
CurObj: code (Undefined)
```

Demo: APL ↔ dyalog.WSEngine

The screenshot shows the Dyalog APL/W-64 interface with two windows. The left window displays the version and serial number information, and the right window shows the execution of a program that interacts with the WSEngine.

Left Window: CLEAR WS - Dyalog APL/W-64

```
Dyalog APL/W-64 Version 18.2.50027
Serial number: 000091
Sat Sep 21 11:14:44 2024
E:\Ergo\WSEngine\WSEngine.DWS saved Thu
prepare [WA...

Starte [CY F:\tmp\WSEngine_Demo.dws
[CY beendet
LOAD(CY) erfolgreich durchgeführt
```

Right Window: CLEAR WS - Dyalog APL/W-64

```
0 SysCommand: LOAD Parameter: F:\tmp\WSEngine_Demo.dws

objAPL.getObjects 2 A #.NL 2
┌ Describe aa bb cc ──┴─┘
objAPL.getObjects 3 A #.NL 3
┌ demo_WSEngine func_with_error plus test_dy test_mon test_nil ──┴─┘
[+x←objAPL.Variable<'aa' A get variable "aa"
ABC123
[+x←objAPL.Variable<'cc' A get variable "cc"

wert1 2222
wert2 1 2
      3 4
wert3 111 2.22 ~3.333
```

Debugger: Ready... CurObj: code (Undefined)

System Tray: 11:16 21.09.2024

Demo: APL ↔ dyalog.WSEngine

The screenshot displays the Dyalog APL/W-64 environment with two windows. The left window shows the startup sequence and execution of a demo script. The right window shows the execution of APL code with detailed function call logs and variable values.

Left Window (F:\tmp\WSEngine_Demo.dws):

```
Dyalog APL/W-64 Version 18.2.50027
Serial number: 000091
Sat Sep 21 11:14:44 2024
E:\Ergo\WSEngine\WSEngine.DWS saved Thu
prepare [WA...

Starte [CY F:\tmp\WSEngine_Demo.dws
[CY beendet
LOAD(CY) erfolgreich durchgeführt
```

Right Window (CLEAR WS - Dyalog APL/W-64):

```
[-2 Fehler beim Lesen der Variablen NotExistingVar:
  VALUE ERROR Variable[38] NotExistingVar

  [-x+objAPL.Call' test_nil'           A Execute and get result: niladic function
test_nil called

  [-x+objAPL.Call' test_mon' 123       A Execute and get result: monadic function

  test_mon called [argument: 123]

  [-x+objAPL.Call' test_dy' 123 ^2     A Execute and get result: dyadic function

  test_dy called [argumenst: ^2 123]

  [-x+objAPL.Call' plus' 123 0.456
123.456
  [-x+objAPL.Call' plus' (13)(0.1x13)
1.1 2.2 3.3
```

Debugger (Bottom Left):

```
Debugger
Ready...
CurObj:
```

Debugger (Bottom Right):

```
Debugger
Ready...
CurObj: function (Undefined)
```

System Tray (Bottom Right): 11:18, 21.09.2024

Demo: APL ↔ dyalog.WSEngine

The screenshot displays the Dyalog APL IDE interface. The main window, titled 'CLEAR WS - Dyalog APL/W-64', shows the execution of several APL functions. The output is as follows:

```
⚠VALUE ERROR Variable[38] NotExistingVar  
  
[]+x+objAPL.Call<'test_nil'  
test_nil called A Execute and get result: niladic function  
[]+x+objAPL.Call<'test_mon' 123 A Execute and get result: monadic function  
test_mon called argument: 123  
  
[]+x+objAPL.Call<'test_dy' 123 ^2 A Execute and get result: dyadic function  
test_dy called argumentst: ^2 123  
  
[]+x+objAPL.Call<'plus' 123 0.456  
123.456  
[]+x+objAPL.Call<'plus'(ι3)(0.1×ι3)  
1.1 2.2 3.3  
  
[]+x+objAPL.Call<'plus'(ι5)(0.1×ι3) A force an internal error --> Test debugge
```

The left sidebar shows the 'plus' function being edited in the 'Dyalog Serial' editor. The code in the editor is:

```
erg←larg plus arg  
erg←larg+arg
```

The bottom status bar shows 'CurObj: debugger (Undefined)' and system information including '8:1', 'DDQ:0', 'TRAP', 'SI:0', 'IO:1', and the date '21.09.2024'.

Demo: APL ↔ dyalog.WSEngine

The image shows a Windows desktop with two Dyalog APL/W-64 windows. The left window, titled 'F:\tmp\WSEngine_Demo.dws - Dyalog APL/W-64', displays the following text:

```
Dyalog APL/W-64 Version 18.2.50027
Serial number: 000091
Sat Sep 21 11:14:44 2024
E:\Ergo\WSEngine\WSEngine.DWS saved Thu
prepare WA...

Starte CY F:\tmp\WSEngine_Demo.dws
CY beendet
LOAD(CY) erfolgreich durchgeführt
LENGTH ERROR: Mismatched left and right
plus[2] erg+larg+arg
      ^
      arg=10000
+lc
2024 9 21 11 19 46 703
2024 9 21 11 19 50 930
```

The right window, titled 'CLEAR WS - Dyalog APL/W-64', displays the following APL code and comments:

```
123.456
[x+objAPL.Call'plus'(t3)(0.1x13)
1.1 2.2 3.3

[x+objAPL.Call'plus'(t5)(0.1x13)           A force an internal error --> Test debugge
10000.1 10000.2 10000.3
[x+objAPL.Exec='SE.Link.Import'#.Utils' 'F:\tmp\Utils'   A Import Code
Imported: #.Utils + F:\tmp\Utils
objAPL.getObjects 9

[ Utils | WSEngine

[x+objAPL.Call='#.Utils.Δuserid'           A Execute and get result: niladic function
jwied
[x+objAPL.Exec='dl 3 ◊ ''done''           A Execute synchronous, wait for result
done
[x+objAPL.Exec_async='TS ◊ dl 4 ◊ TS '   A Execute asynchronous, don't wait

[ 0 | Kommando in Event-Queue gestellt

[x+objAPL.Call'plus'(t3) 0.01           A next command, don't wait
1.01 2.01 3.01
```

At the bottom, a Windows taskbar is visible with the search bar, taskbar icons, and system tray showing the time 11:19 and date 21.09.2024.

Demo: APL ↔ dyalog.WSEngine

The screenshot shows a Windows desktop with a blue background. The taskbar at the bottom includes the search bar, task view, and several application icons. A window titled "CLEAR WS - Dyalog APL/W-64" is open, displaying the APL workspace interface. The workspace contains the following code and output:

```
⎕←x←objAPL.Exec←'⎕SE.Link.Import'#.Utils'' 'F:\tmp\Utils'' A Import Code
Imported: #.Utils ← F:\tmp\Utils
objAPL.getObjects 9

[ Utils | WSEngine ]

⎕←x←objAPL.Call←'#.Utils.Δuserid' A Execute and get result: niladic function
jwied
⎕←x←objAPL.Exec←'⎕dl 3 ⋄ 'done'' A Execute synchronous, wait for result
done
⎕←x←objAPL.Exec_async←'⎕TS ⋄ ⎕dl 4 ⋄ ⎕TS ' A Execute asynchronous, don't wait

[ 0 Kommando in Event-Queue gestellt ]
⎕←x←objAPL.Call'plus'(⊔3) 0.01 A next command, don't wait
1.01 2.01 3.01

⎕←x←objAPL.Exec_async←'⎕OFF' A close remote APL session

[ 0 Kommando in Event-Queue gestellt ]
⎕EX'objAPL' A expunge object
```

The workspace also shows a Debugger window at the bottom with the status "Ready..." and "CurObj: object (Undefined)".



dyalog.WSEngine

Thank you for listening

Jürgen Wiedemann
DPC

Dyalog '24
17. September 2024