dyalog.WSEngine

Interaktive Nutzung des Dyalog Interpreters als OLE-Server

(Microsoft Windows)

Jürgen Wiedemann
DPC

APL Germany Frühjahrstagung 2025
28. März 2025

Agenda

- Ausgangssituation und Anforderungen
- Umsetzung
- Demo

Ausgangssituation und Anforderungen

- Migration von APL+Win nach Dyalog APL
  (mit allen tpyischen Herausforderungen)
- Anforderung
  - Bereitstellung einer Funktionalität wie in APL+Win:
    “APL+Win as an ActiveX server”
- Möglichkeit
  - zur Erzeugung einer neuen APL-Instanz
  - zum Aufruf aus APL und Excel heraus
  - zur Nutzung beliebiger Workspaces
    ohne zusätzliche Vorbereitung oder Bereitstellung

Einsatzgebiet / Fallbeispiele

- Regressionstest mit parallelen Berechnungen in mehreren lokalen APL-Sessions
- Integration von APL in andere Umgebungen, z.B. in Excel (VBA)
- Einsatz im “Entwicklermodus”:
  - komplette Funktionalität der IDE
  - sichtbar
  - Debugger

Herausforderung

- Dyalog bietet keine “fertige” Lösung für diese Anforderungen.
- Man kann zwar
  - Workspaces als Runtime, In- und Out-of-Process Server oder .Net Assembly exportieren.
  - Namespaces via Jarvis bereitstellen.
  - Aber stets mit einer fest definierten, festen Codebasis.
- IDE mit Editor und Debugger ?!?
  Genau dies aber war die Haupt-Anforderung

Lösung

- OLEServer Object (□WC)
  - Erlaubt Sichtbarkeit und vollen Zugriff auf die Session (überraschenderweise)
  - Stellt aber einen festen Namespace und nur bestimmte Funktionen daraus bereit
  - Exkurs:
    Ein OLEServer Object wird aus einem Namespace erzeugt und die bereitgestelten Einstiegspunkte werden über dessen Eigenschaft “ExportedFns” definiert.
- Emulation der meisten APL+-Features

Lösungsvorlage: APL+Win

- Overview “APL+Win as an ActiveX server”

Methods	Description
Call	Call a function and return its value
Exec	Execute an expression and return its value
SysCommand	Execute a system-command
Properties	
SysVariable	Return or set the value of a system-variable
Variable	Return or set the value of a variable
Visible	Return or set the visibility of the APL session
onComAction	Triggered in the APL server when a command is initiated in the client session
onNotify	Triggered when the APL server's system object invokes the Notify method
onSysNotify	Triggered when a specified action occurs on the APL server

dyalog.WSEngine - Überblick

- Sehr kleiner Namespace als OLEServer
- Sehr wenige und allgemeine Methoden
- Zweck: Umgehung von Beschränkungen des OLEServer Objekts
- Der OLEServer Namespace fungiert nur als Zwischenschicht zur “dynamischen” Nutzung beliebiger APL Quellen.

dyalog.WSEngine - Details

SysCommand

```
⍝ ⎕CY into OLE-Session
objAPL.SysCommand⊂'LOAD ',ws
```

SysVariable

```
⍝ Get ⎕WA value
objAPL.SysVariable⊂'WA'
```

Variable

```
⍝ Return or set value of a variable
x←objAPL.Variable⊂'APLvar'
x←objAPL.Variable 'XX'(2 2⍴⍳4)
```

Call

```
⍝ Call niladic/monadic/dyadic fn
x←objAPL.Call⊂'fn_nil'
x←objAPL.Call 'fn_mon' 123
x←objAPL.Call 'fn_dya' 123 ¯2
```

dyalog.WSEngine - Details

Exec

```
⍝ Execute expressions, return value
x←objAPL.Exec⊂'⎕NC⊂''myVar'''
```

Exec_async
(no result)

```
⍝ Execute asynchr. (add to event queue
objAPL.Exec_async⊂'⎕OFF'
objAPL.Exec_async⊂'longCalc'
```

setVisible

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

getObjects

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

Update

```
⍝ Update WSEngine namespace
objAPL.Update⊂'E:\Fix.dws'
```

Deployment

- Registrierung
  - Mit der OLERegister Methode, in ⎕LX
  - Aufruf mit Option /RegWSEngine
  - **`dyalog.exe WSEngine.dws -MAXWS=256M RegWSEngine=1 DYALOG_NOPOPUPS=1`**
- Aufruf
  - VBA

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

  - Dyalog

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

Beispiele

Aus 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'
```

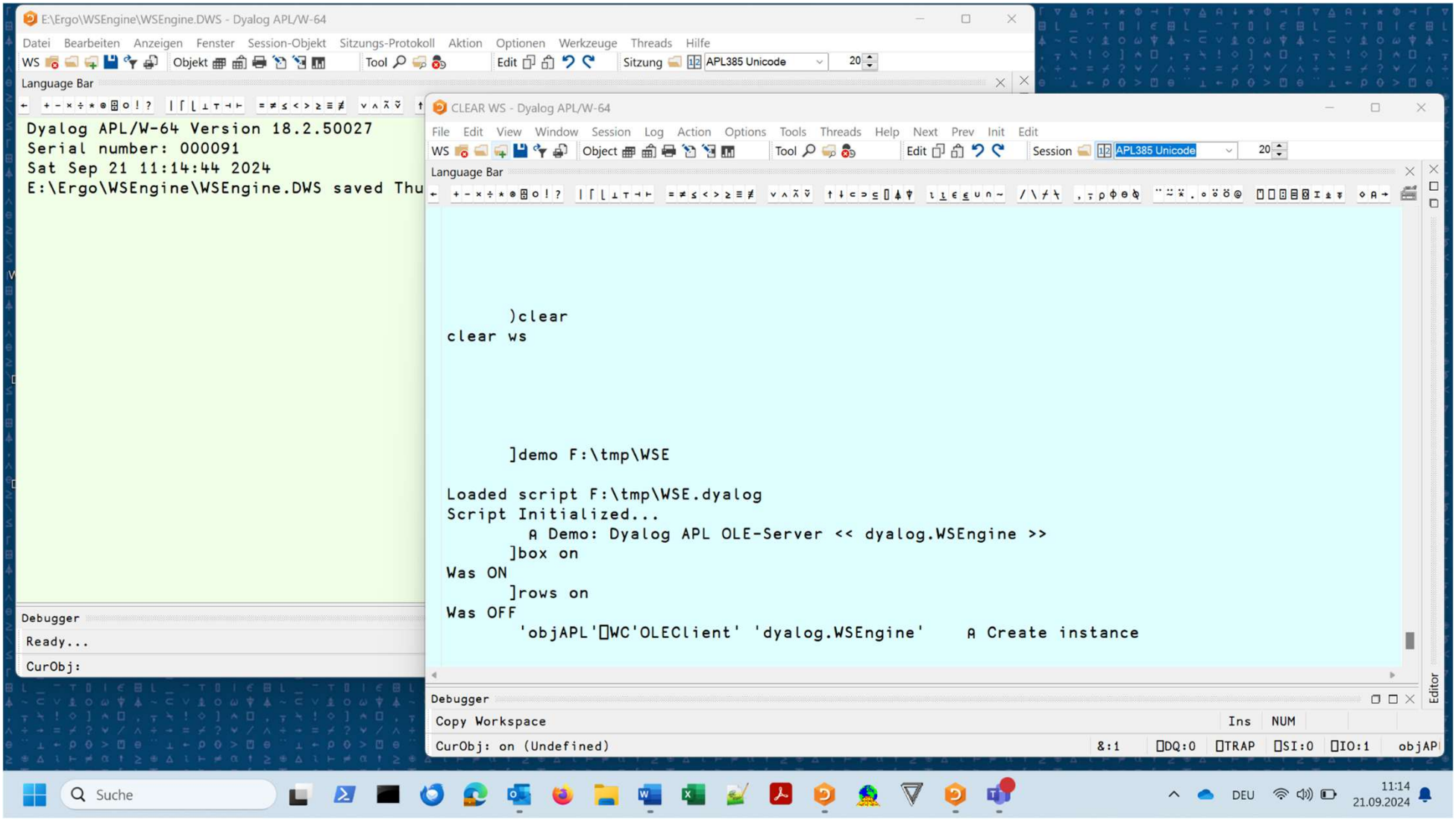
Aus Excel (VBA)

```
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
```

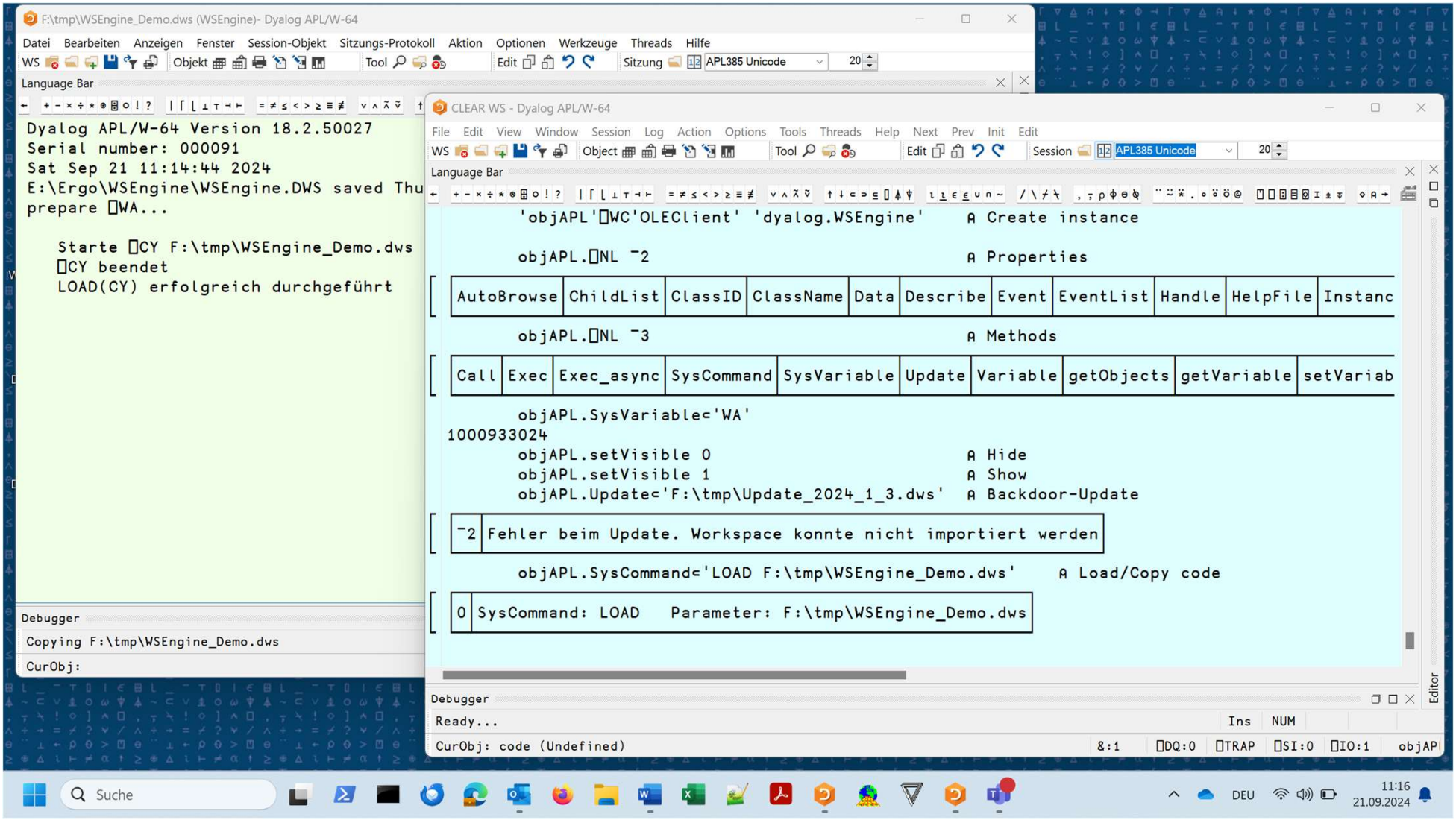
Demo

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

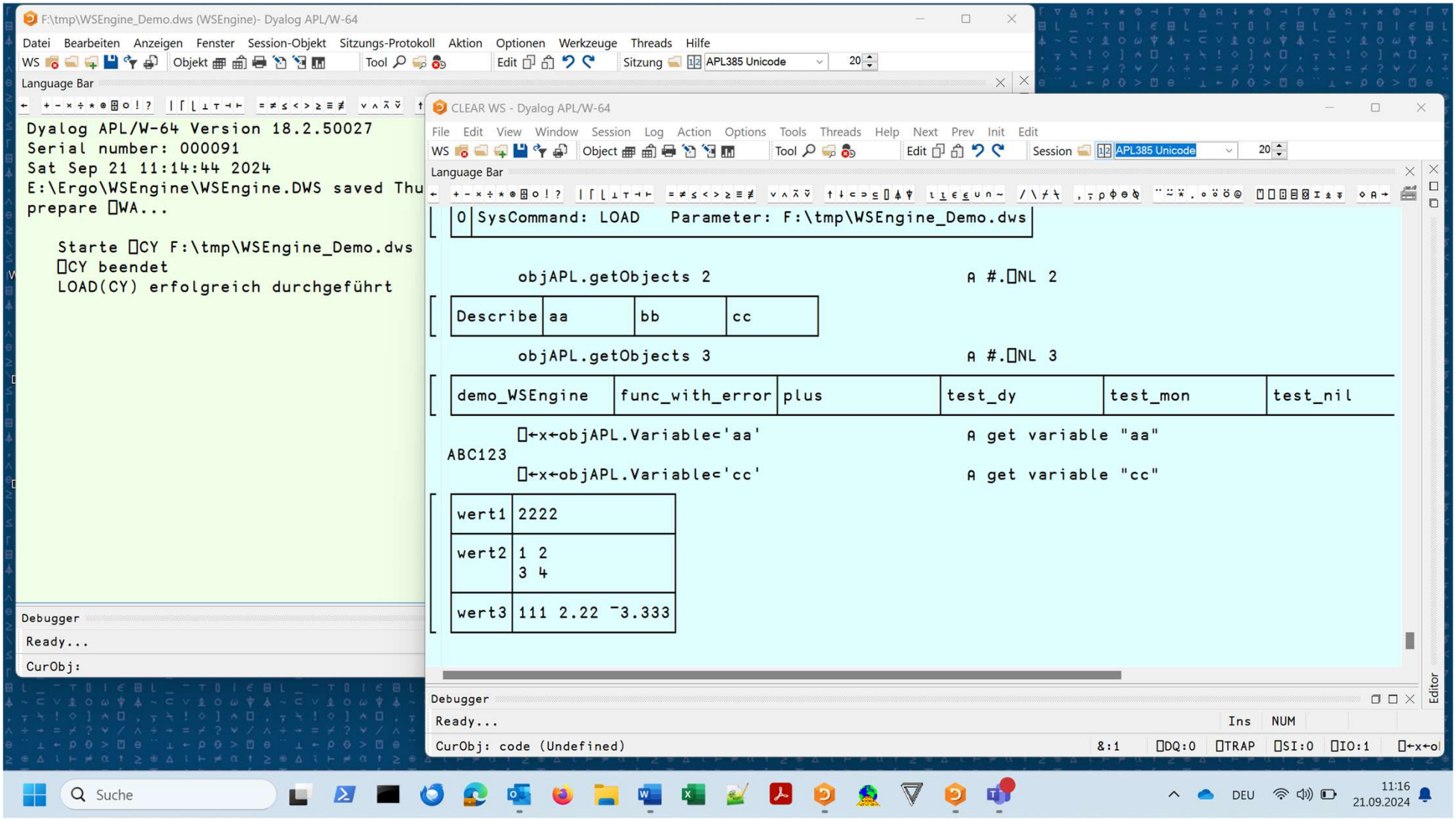
Demo: APL ↔ dyalog.WSEngine



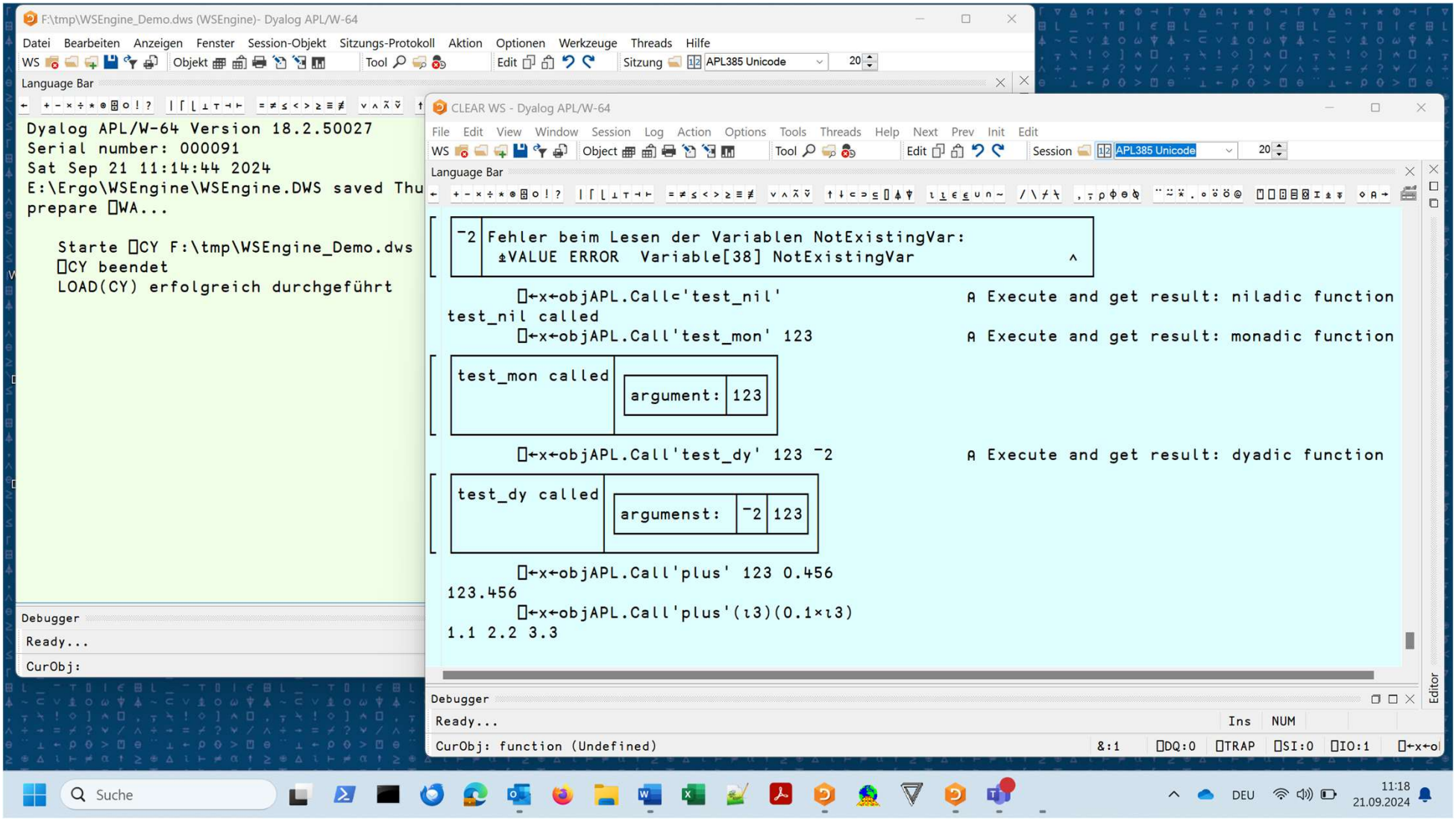
Demo: APL ↔ dyalog.WSEngine



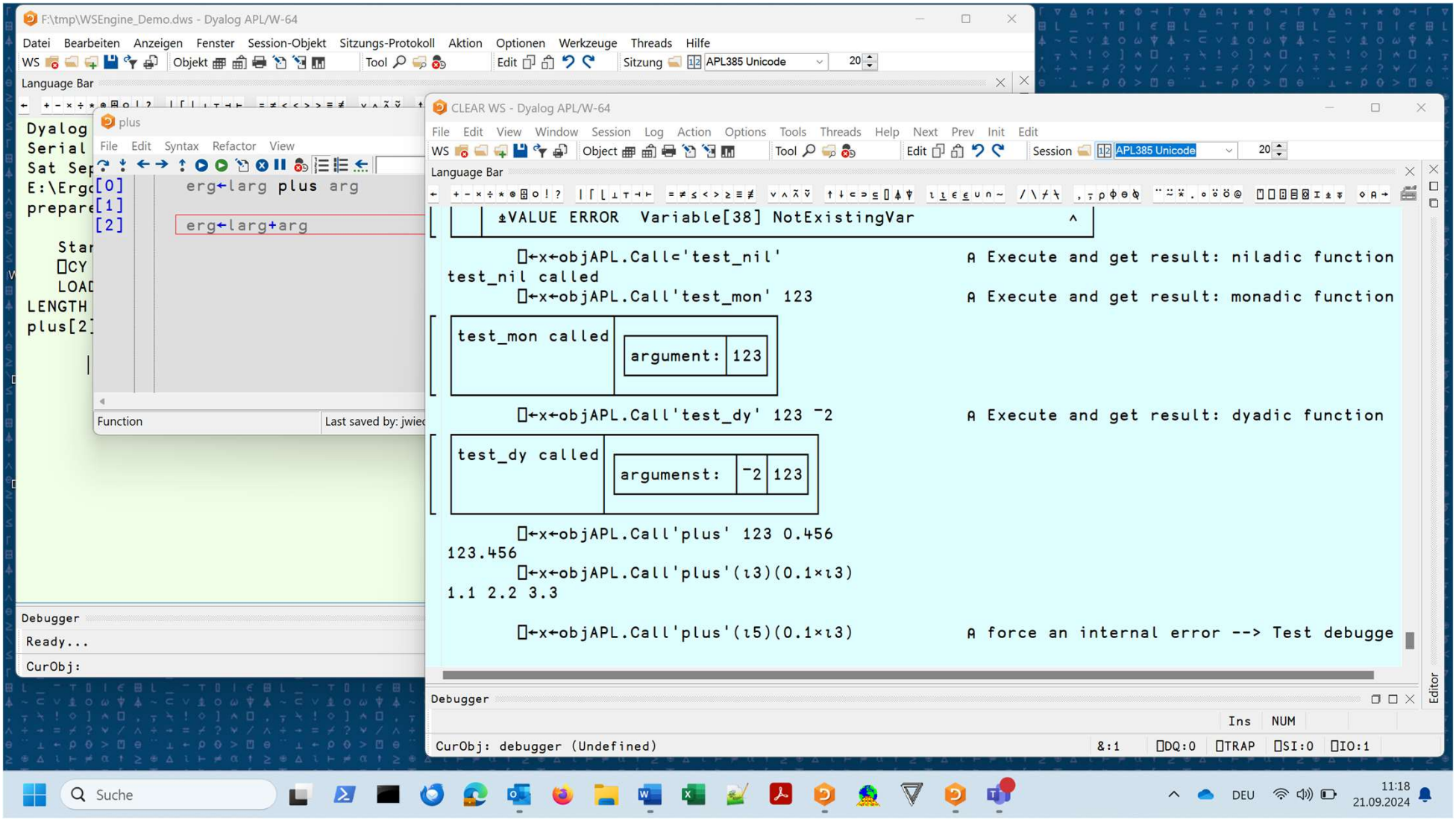
Demo: APL ↔ dyalog.WSEngine



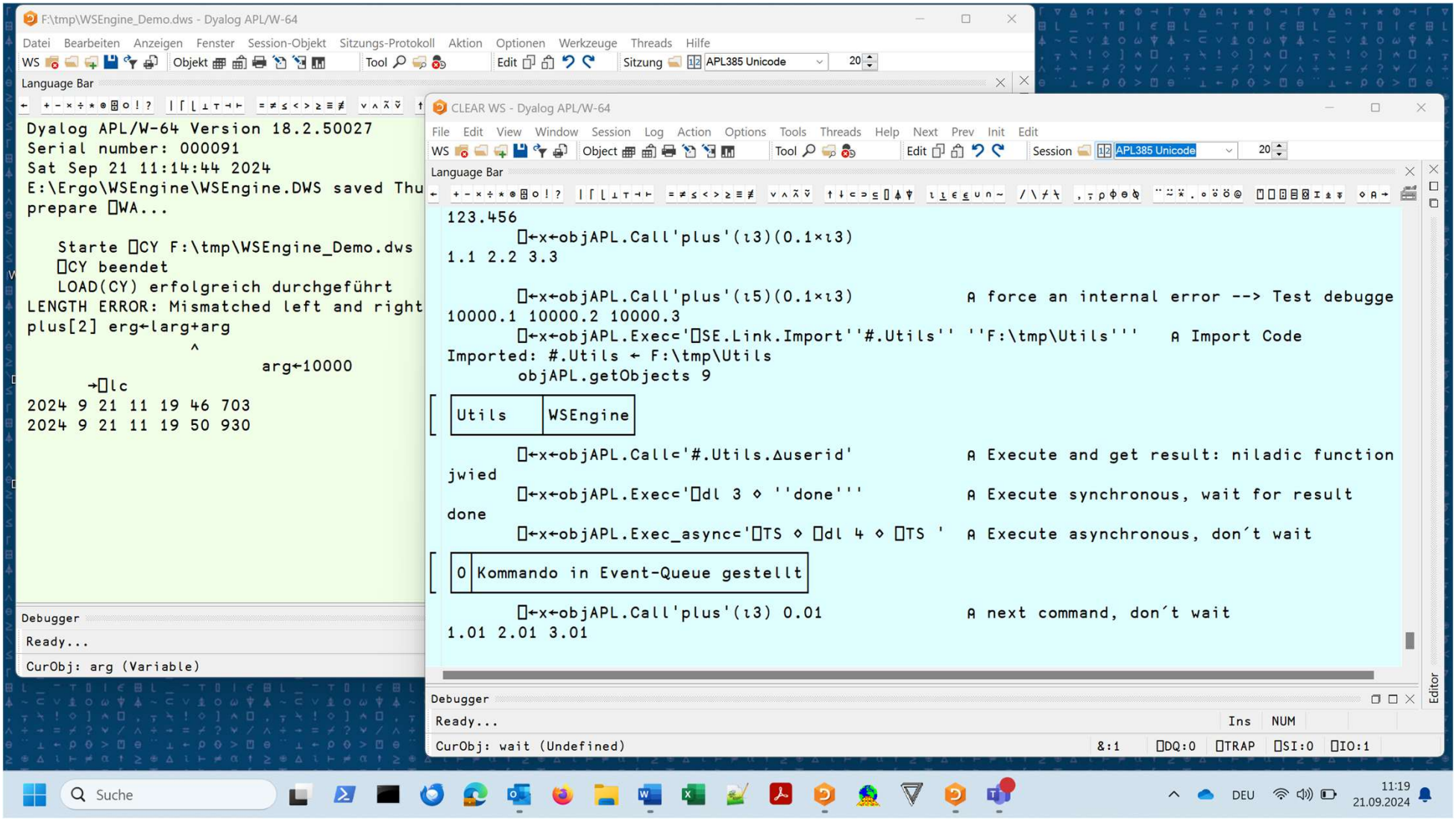
Demo: APL ↔ dyalog.WSEngine



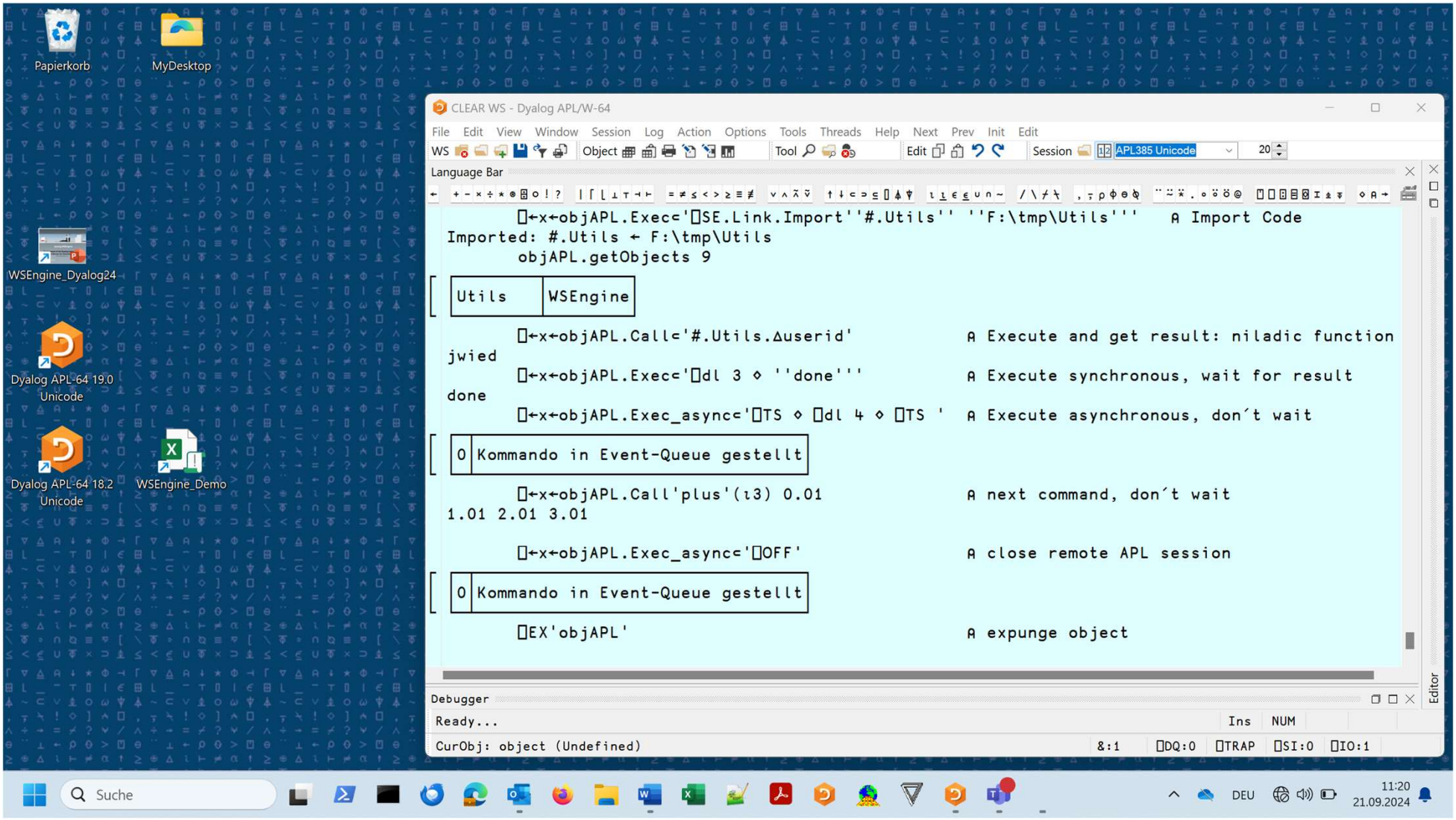
Demo: APL ↔ dyalog.WSEngine



Demo: APL ↔ dyalog.WSEngine



Demo: APL ↔ dyalog.WSEngine



dyalog.WSEngine

Fragen… ?

Jürgen Wiedemann
DPC

APL Germany Frühjahrstagung 2025
28. März 2025