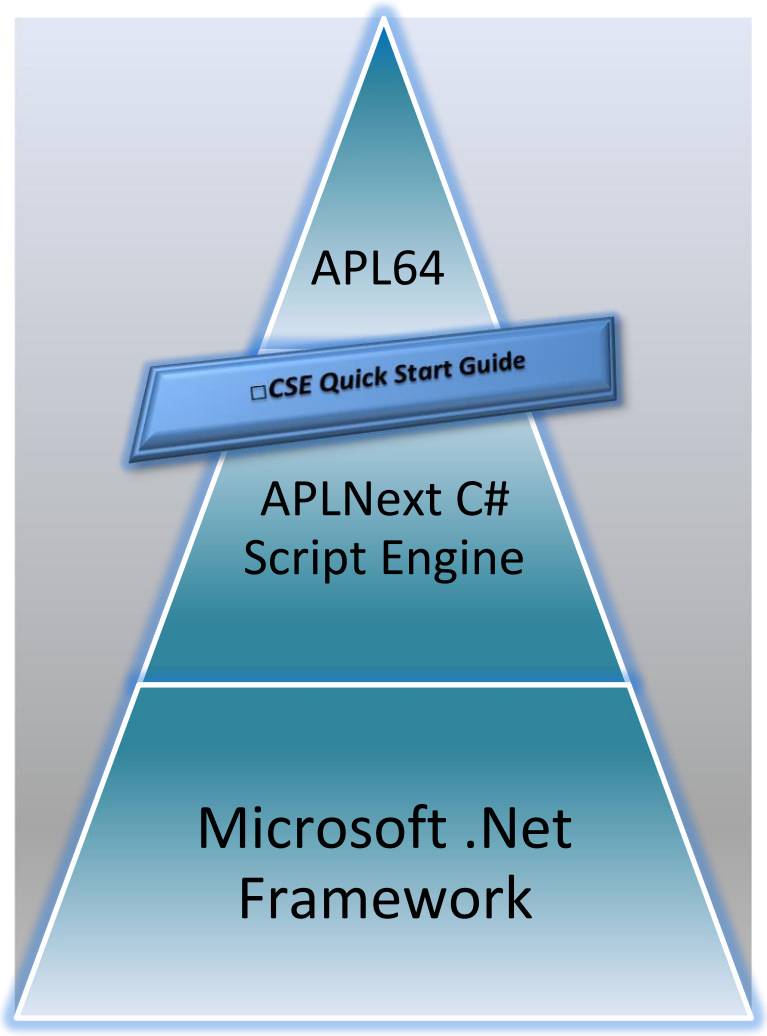

APL64
□CSE Quick Start Guide
APLNext C#
Script Engine
Microsoft .Net
Framework

Table of Contents

Overview – Get the Benefits of the .Net Toolkit in APL64

The Microsoft .Net Framework is an extensive, well-documented and easy-to-use application programming interface (API) toolkit which is incorporated at no additional cost in recent versions of the Microsoft Windows operating system for workstations and servers.

The APLNext C# Script Engine (CSE) empowers the APL64 programmer with the benefits of the Microsoft .Net Framework in an enhanced version of APL64 which maintains full compatibility with prior versions of APL64.

The CSE simplifies the use of the .Net Framework without the need to use Microsoft Visual Studio.

The Microsoft C# programming language is the premier tool used world-wide by millions of programmers to access and use the Microsoft .Net Framework. Ready-to-run, sample C# source code is available without cost from Microsoft and other sources on the Internet for virtually all application system development needs.

The C# statements necessary to accomplish a task in the .Net Framework are easy-to-understand, 'English language', object-oriented statements in contrast to the numerous special 'codes', 'structures' and 'buffers' that are necessary when the prior Win32 API is used.

APL64 incorporates the □cse system function as the interface to the CSE. The APL64 □cse system function provides a convenient syntax for using the CSE engine without imposing the .Net Framework on APL64.

Installing the CSE

The APLNext C# Script Engine is included with APL64 developer and runtime versions. No separate installation is required.

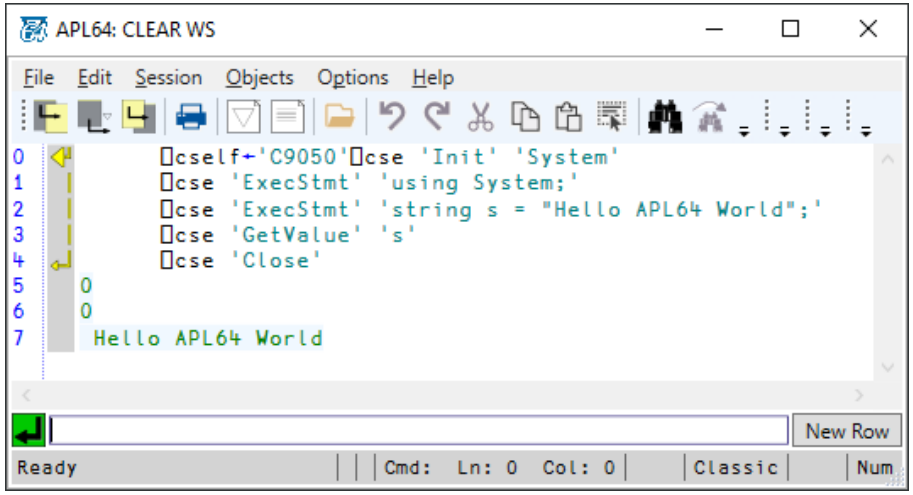
Examples

Hello World – Your First Program using APL64 and the CSE

The APL64 executable statements for this example:

```
□cself←'C9050'□cse 'Init' 'System'
□cse 'ExecStmt' 'using System;'
□cse 'ExecStmt' 'string s = "Hello APL64 World";'
□cse 'GetValue' 's'
```

What will happen in the APL64 session:

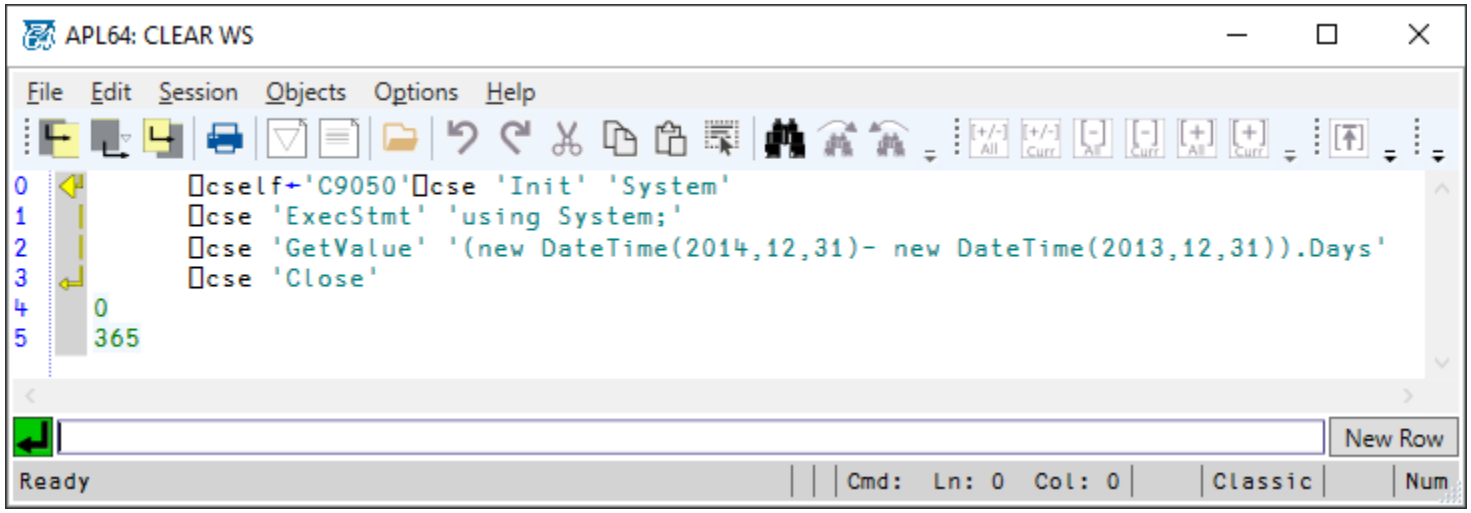


Date Time Calculation in APL64 using .Net

The APL64 executable statements for this example:

```
□cself←'C9050'□cse 'Init' 'System'
□cse 'ExecStmt' 'using System;'
□cse 'GetValue' '(new DateTime(2014,12,31)- new DateTime(2013,12,31)).Days'
□cse 'Close'
```

What will happen in the APL64 session:

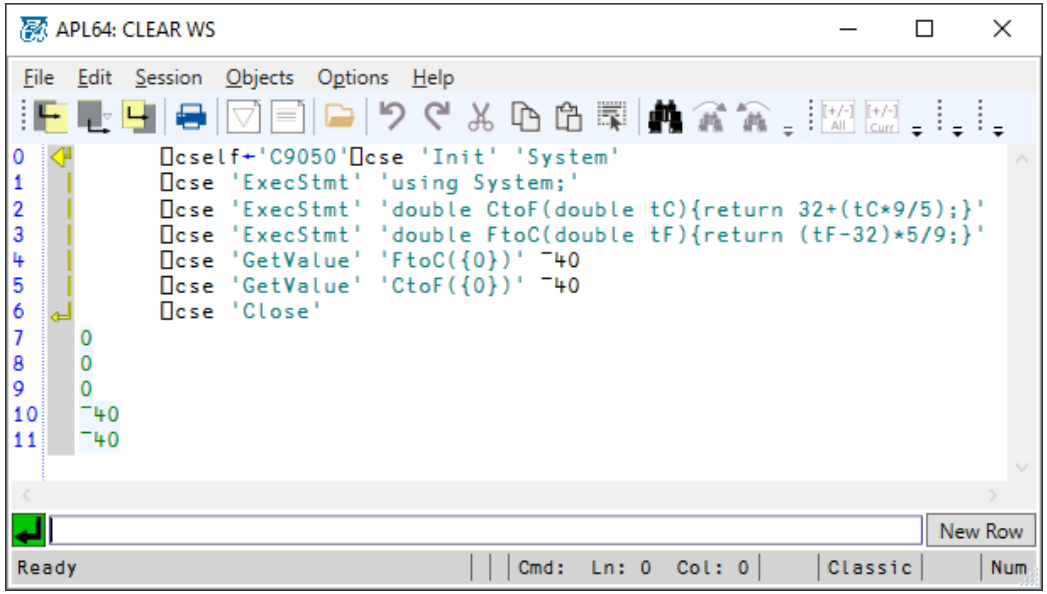


Create a .Net Temperature Converter Method and use it from APL64

The APL64 executable statements for this example:

```
⎕cself←'C9050'⎕cse 'Init' 'System'
⎕cse 'ExecStmt' 'using System;'
⎕cse 'ExecStmt' 'double CtoF(double tC){return 32+(tC*9/5);}'
⎕cse 'ExecStmt' 'double FtoC(double tF){return (tF-32)*5/9;}'
⎕cse 'GetValue' 'FtoC({0})' ¯40
⎕cse 'GetValue' 'CtoF({0})' ¯40
⎕cse 'Close'
```

What will happen in the APL64 session:



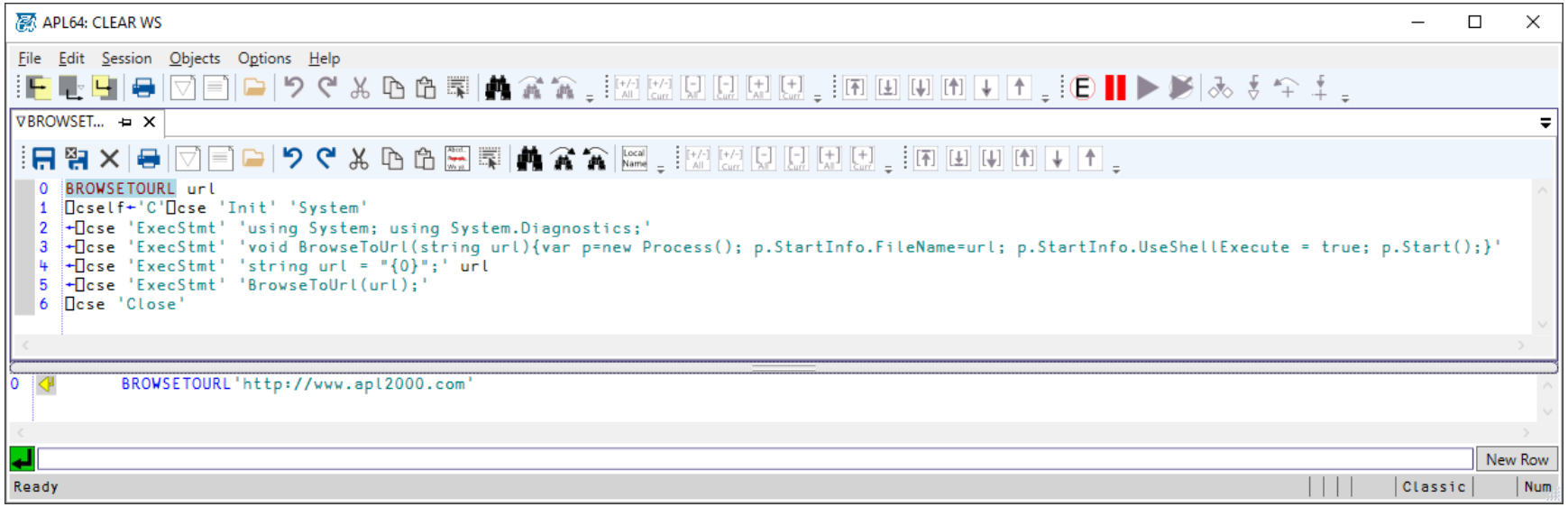
Browse to a Web Page in the default browser

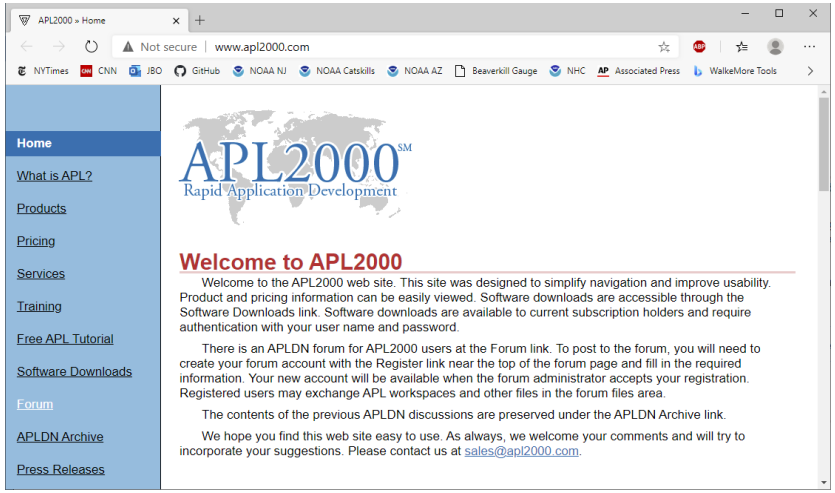
In the APL64 developer version, create the user-defined function 'BROWSETOURL' using this definition:

```
BROWSETOURL url
⎕cself←'C'⎕cse 'Init' 'System'
←⎕cse 'ExecStmt' 'using System; using System.Diagnostics;'
←⎕cse 'ExecStmt' 'void BrowseToUrl(string url){var p=new Process(); p.StartInfo.FileName=url; p.StartInfo.UseShellExecute = true; p.Start();}'
←⎕cse 'ExecStmt' 'string url = "{0}";' url
←⎕cse 'ExecStmt' 'BrowseToUrl(url);'
⎕cse 'Close'
```

Save the BROWSETOURL function definition to the current workspace.

Execute the BROWSETOURL function with the argument ‘http://www.apl2000.com’





I don't know C#. Do I need to learn it? How can I learn it?

This 'QuickStart' document shows a few very simple □cse examples. Refer to the complete □cse documentation installed with APL64 or on the APL2000 CSE Forum for more information and example workspaces. The .Net Framework and the APLNext C# Script Engine can be used by an APL64 programmer to create:

- Modern, intuitive graphical user interfaces
- High performance implementations of complex algorithms using industry-standard .Net tools
- Secure application data stores using databases such as Microsoft SQL Server
- Application systems which incorporate selected elements of the vast .Net Framework

If the CSE will be used to load and interact with previously-compiled .Net assemblies, simple and minimal C# skill is needed to benefit from using the CSE. If CSE scripts will be written using C# syntax to create .Net classes, additional knowledge of the C# .Net programming language is beneficial. The C# programming language and the Microsoft .Net framework are both well implemented and documented and have been extensively deployed to develop productive and amazing applications on the Windows workstation and server operating system platforms.

APL64 supports interfaces to many programming 'environments' and the CSE provides a robust interface to the C# and .Net environment for those who need it.

Many avenues are available for a programmer interested in learning C# and .Net including:

- Microsoft Developer Network
- MSDN Library
- MSDN Library C#
- Stackoverflow for C# Questions
- Apress Publications for C#
- APL2000 Consulting Services

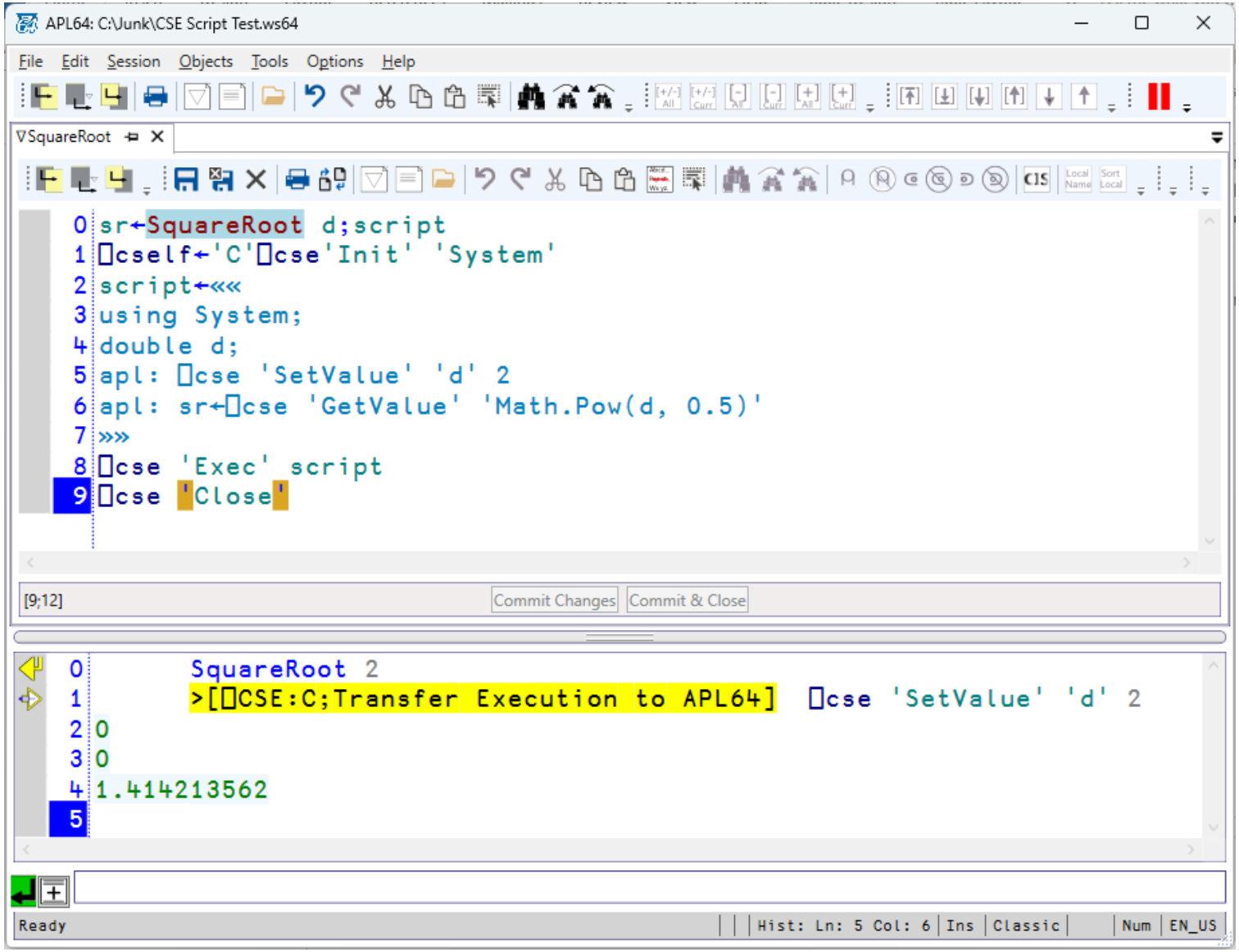
Familiar with C#? Start using CSE Scripts

A CSE script is an APL text variable created within an APL64 programmer-defined function. A CSE script can contain multiple C# and APL64 executable statements. A CSE script can be executed *en masse* using the CSE Exec method.

In this APL64 programmer defined function:

- A CSE instance named C is created
- The APL64 string line continuation structure (script←«« … »») is used to create a multi-line CSE script
    - The C# scalar variable, d, is created to contain the function right argument
    - The CSE SetValue method is used to set the variable d with the function right argument
    - The CSE GetValue method is used obtain the function result, sr

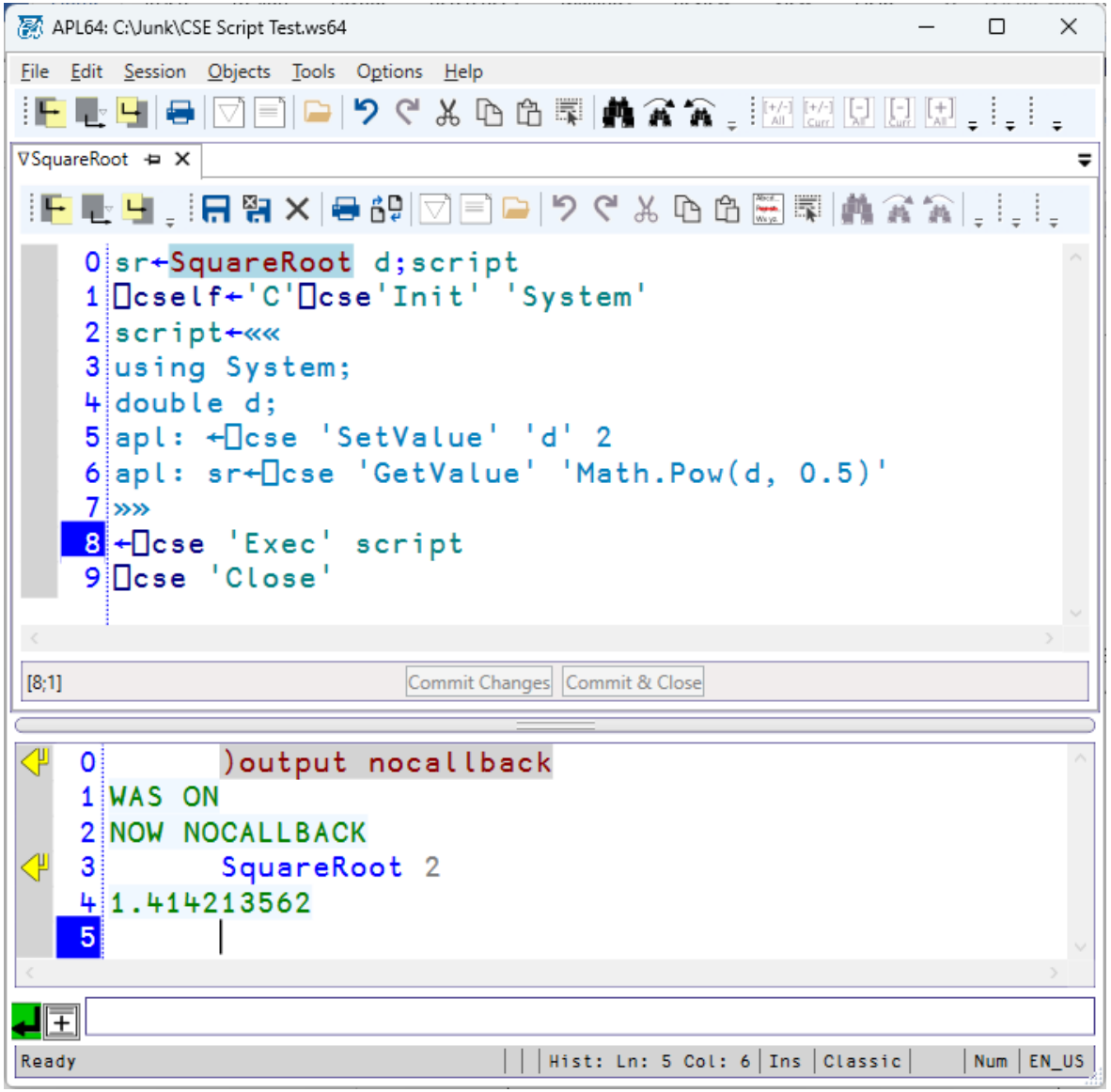
```
sr←SquareRoot d;script
⍝cself←'C'⍝cse'Init' 'System'
script←««
using System;
double d;
apl: ⍝cse 'SetValue' 'd' 2
apl: sr←⍝cse 'GetValue' 'Math.Pow(d, 0.5)'
»»
⍝cse 'Exec' script
⍝cse 'Close'
```


APL64: C:\Junk\CSE Script Test.ws64
File Edit Session Objects Tools Options Help
∇SquareRoot
0 sr←SquareRoot d;script
1 ⎕cself←'C'⎕cse'Init' 'System'
2 script←«««
3 using System;
4 double d;
5 apl: ⎕cse 'SetValue' 'd' 2
6 apl: sr←⎕cse 'GetValue' 'Math.Pow(d, 0.5)'
7 »»»
8 ⎕cse 'Exec' script
9 ⎕cse 'Close'
[9;12]
Commit Changes
Commit & Close
0 SquareRoot 2
1 >[⎕CSE:C;Transfer Execution to APL64] ⎕cse 'SetValue' 'd' 2
2 0
3 0
4 1.414213562
5
Ready
Hist: Ln: 5 Col: 6
Ins
Classic
Num
EN_US

When running the SquareRoot function, the non-essential output can be suppressed:

- Sink the output of the CSE Exec and SetValue methods
- Use the)output NOCALLBACK system command

```
sr←SquareRoot d;script
⍞cself←'C'⍞cse'Init' 'System'
script←««
using System;
double d;
apl: ←⍞cse 'SetValue' 'd' 2
apl: sr←⍞cse 'GetValue' 'Math.Pow(d, 0.5)'
»»
←⍞cse 'Exec' script
⍞cse 'Close'
```


APL64: C:\Junk\CSE Script Test.ws64
File Edit Session Objects Tools Options Help
SquareRoot
0 sr←SquareRoot d;script
1 ⎕cself←'C'⎕cse'Init' 'System'
2 script←«««
3 using System;
4 double d;
5 apl: ←⎕cse 'SetValue' 'd' 2
6 apl: sr←⎕cse 'GetValue' 'Math.Pow(d, 0.5)'
7 »»»
8 ←⎕cse 'Exec' script
9 ⎕cse 'Close'
[8;1]
Commit Changes
Commit & Close
0)output nocallback
1 WAS ON
2 NOW NOCALLBACK
3 SquareRoot 2
4 1.414213562
5
Ready
Hist: Ln: 5 Col: 6
Ins
Classic
Num
EN_US