Using ⎕Profile

Contents

Summary

The APL64 ⎕Profile system function may be used to obtain a fine-grained performance profile of APL64 programmer-selected actions. Performance profile information is a detailed accounting of the APL64 interpreter processing of the APL64 programmer-selected actions.

Providing ⎕Profile information to the APL64 development team about an application will help the development team identify potential performance improvements in APL64.

The APL64 programmer-selected actions may be manually entered in the APL64 developer version or selected from a portion of an APL64 developer-created application system.

To obtain a performance profile of a selected portion of an APL64-based application system:

- Insert the APL64 statement: ⎕Profile 'On', in the application source code where profile recording should start.
- Insert the APL64 statement: ⎕Profile 'Save' targetFilename, in the application source code after the APL64 programmer-selected actions have been executed. 'targetFileName' is the APL64 programmer-selected file to save the profile. Use the file extension '.json' which is the default format for an APL64 profile record.
- Insert the APL64 statement: ⎕Profile 'off', in the application source code where profile recording should stop.
- Send the resulting file to the APL64 development team.

Performance profile information contains no proprietary information about the APL64 programmer-created application.

Performance profile information may be transmitted to support@apl2000.com for further analysis, which may yield potential performance improvements in the APL64 interpreter.

When performance profiling is ‘Off’, there is minimal impact on APL64 performance.

What Application Aspects should be profiled?

Suggested application profiling targets:

- ⎕LX function execution
- Significant portions of the application algorithms
- Application data access mechanisms
- Application output preparation

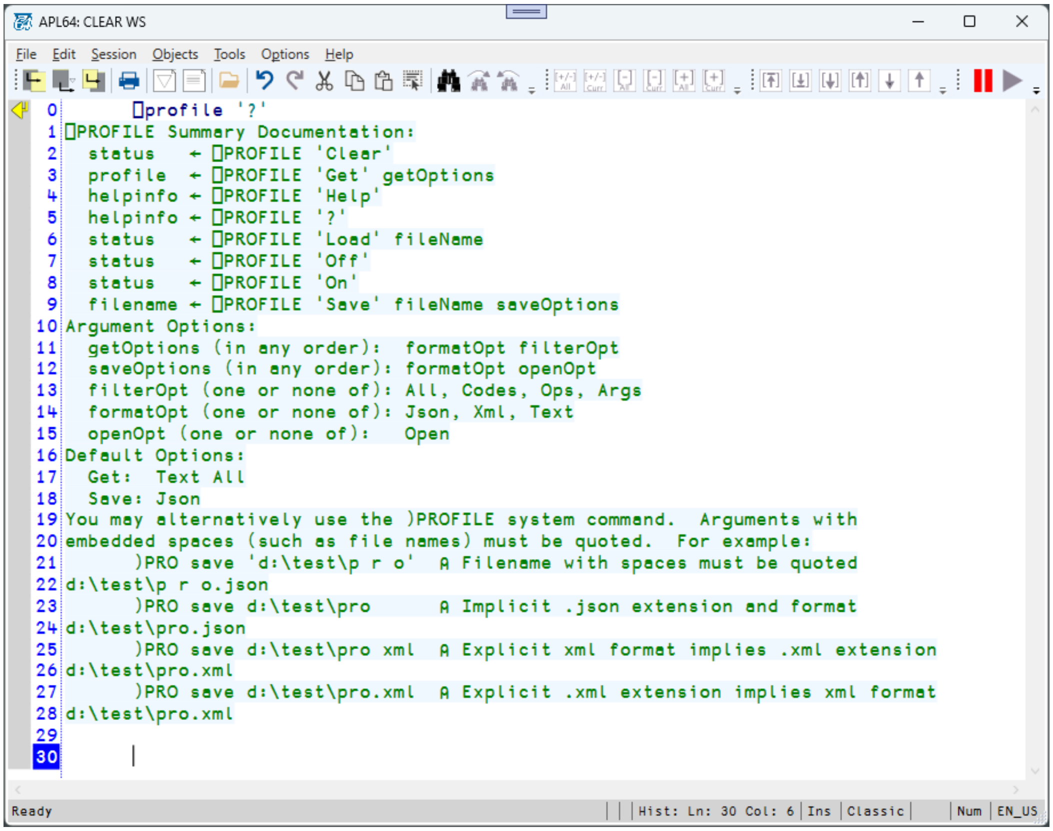
Multiple, separate profile records are more useful and easier to analyze.

Syntax

[result] ← ⎕Profile 'Action' [ActionArg1] [ActionArg2] ...

The ⎕Profile action names are case insensitive.

A summary of ⎕Profile is available at any time during an APL64 developer version instance:



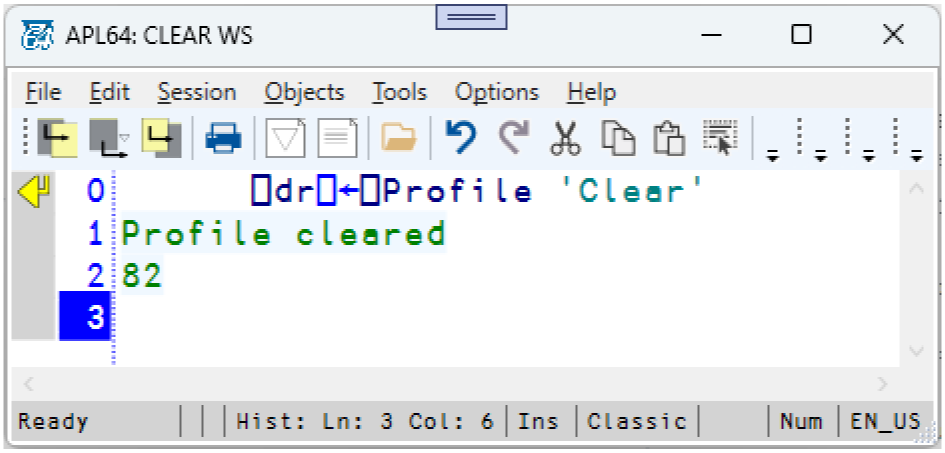
```
      ⎕profile '?'
⎕PROFILE Summary Documentation:
  status   ← ⎕PROFILE 'Clear'
  profile  ← ⎕PROFILE 'Get' getOptions
  helpinfo ← ⎕PROFILE 'Help'
  helpinfo ← ⎕PROFILE '?'
  status   ← ⎕PROFILE 'Load' fileName
  status   ← ⎕PROFILE 'Off'
  status   ← ⎕PROFILE 'On'
  filename ← ⎕PROFILE 'Save' fileName saveOptions
Argument Options:
  getOptions (in any order):  formatOpt filterOpt
  saveOptions (in any order): formatOpt openOpt
  filterOpt (one or none of): All, Codes, Ops, Args
  formatOpt (one or none of): Json, Xml, Text
  openOpt (one or none of):   Open
Default Options:
  Get:  Text All
  Save: Json
You may alternatively use the )PROFILE system command.  Arguments with
embedded spaces (such as file names) must be quoted.  For example:
      )PRO save 'd:\test\p r o'  ⍝ Filename with spaces must be quoted
d:\test\p r o.json
      )PRO save d:\test\pro      ⍝ Implicit .json extension and format
d:\test\pro.json
      )PRO save d:\test\pro xml  ⍝ Explicit xml format implies .xml extension
d:\test\pro.xml
      )PRO save d:\test\pro.xml  ⍝ Explicit .xml extension implies xml format
d:\test\pro.xml

```

Actions

Clear

Char[ ] ← ⎕Profile 'Clear'

The clear action initializes any performance profile information which may have been previously recorded during the APL64 instance.



```
      ⎕dr⎕←⎕Profile 'Clear'
Profile cleared
82
```

Get

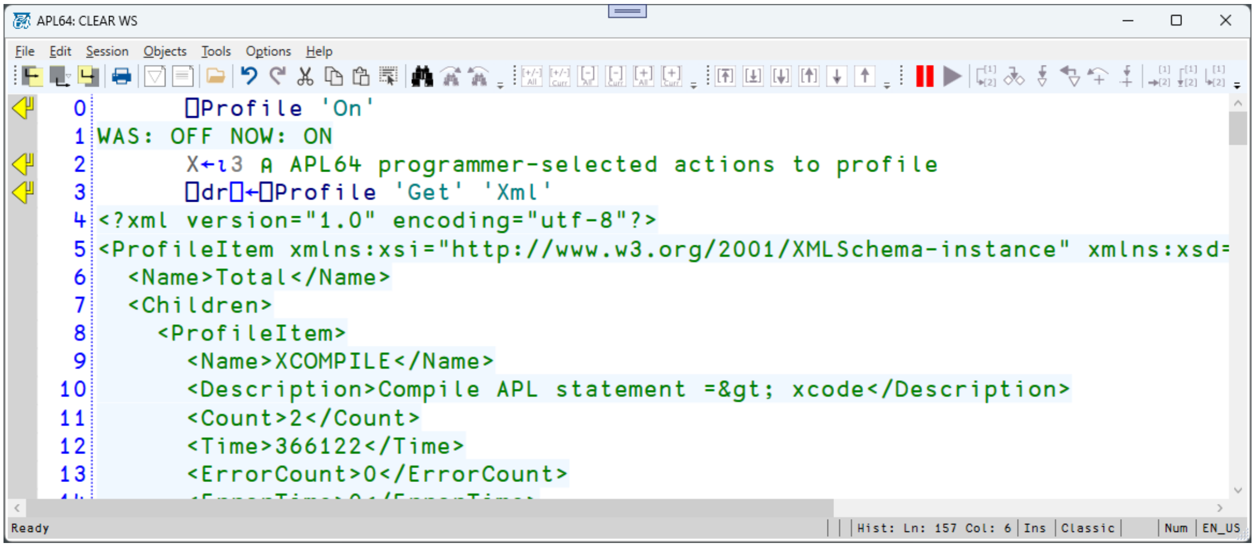
Status(Char[ ]) ← ⎕Profile 'Get' getOptions

getOptions (in any order) : formatOption, filterOption

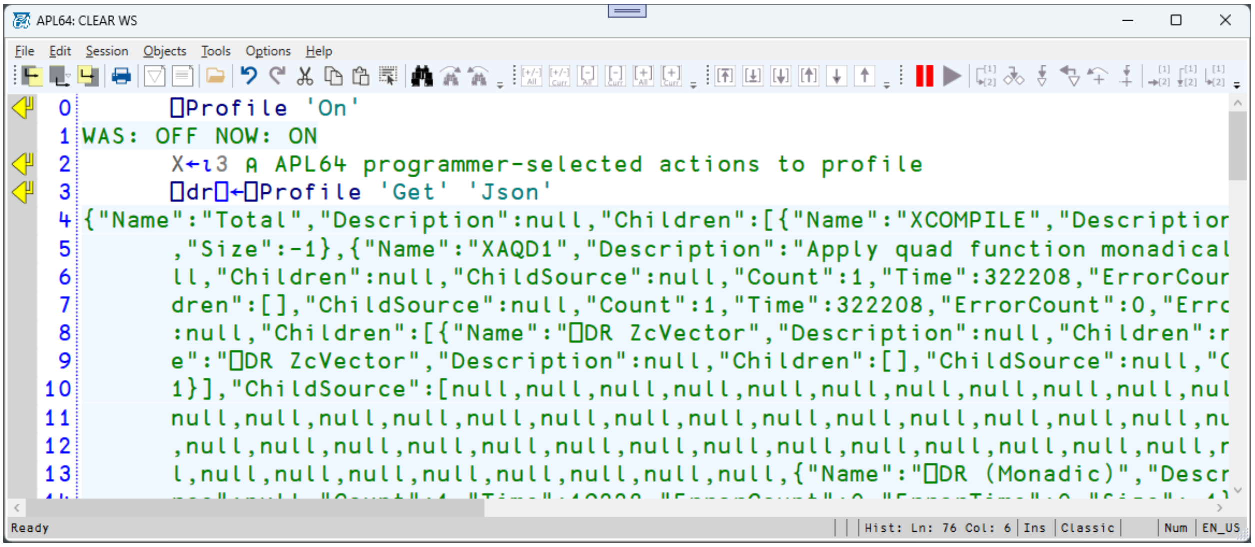
filterOptions : All, Codes, Ops, Args

formatOptions: Json(default), Xml, Text

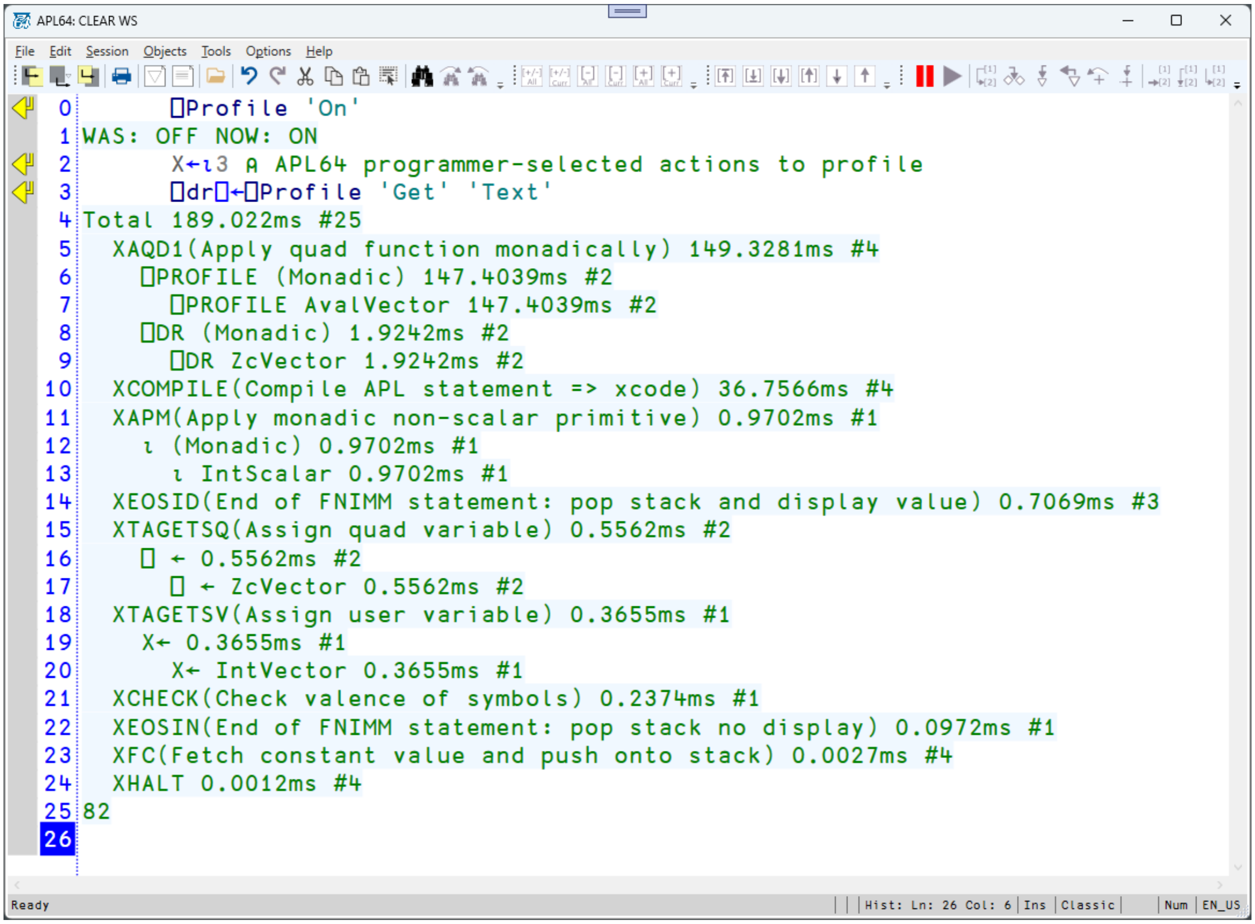
The Get action returns a character vector containing the json- or xml-serialization or text summary of the performance profile.



```
      ⎕Profile 'On'
WAS: OFF NOW: ON
      X←⍳3 ⍝ APL64 programmer-selected actions to profile
      ⎕dr⎕←⎕Profile 'Get' 'Xml'
<?xml version="1.0" encoding="utf-8"?>
<ProfileItem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
  <Name>Total</Name>
  <Children>
    <ProfileItem>
      <Name>XCOMPILE</Name>
      <Description>Compile APL statement =&gt; xcode</Description>
      <Count>2</Count>
      <Time>366122</Time>
      <ErrorCount>0</ErrorCount>
```


APL64: CLEAR WS
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 76 Col: 6 | Ins | Classic | Num | EN_US

```
      ⎕Profile 'On'
WAS: OFF NOW: ON
      X←⍳3 ⍝ APL64 programmer-selected actions to profile
      ⎕dr⎕←⎕Profile 'Get' 'Json'
{"Name":"Total","Description":null,"Children":[{"Name":"XCOMPILE","Descriptior
      ,"Size":-1},{"Name":"XAQD1","Description":"Apply quad function monadical
      ll,"Children":null,"ChildSource":null,"Count":1,"Time":322208,"ErrorCour
      dren":[],"ChildSource":null,"Count":1,"Time":322208,"ErrorCount":0,"Errc
      :null,"Children":[{"Name":"⎕DR ZcVector","Description":null,"Children":r
      e":"⎕DR ZcVector","Description":null,"Children":[],"ChildSource":null,"C
      1}],"ChildSource":[null,null,null,null,null,null,null,null,null,null,nul
      null,null,null,null,null,null,null,null,null,null,null,null,null,null,nu
      ,null,null,null,null,null,null,null,null,null,null,null,null,null,null,r
      l,null,null,null,null,null,null,null,null,{"Name":"⎕DR (Monadic)","Descr
```

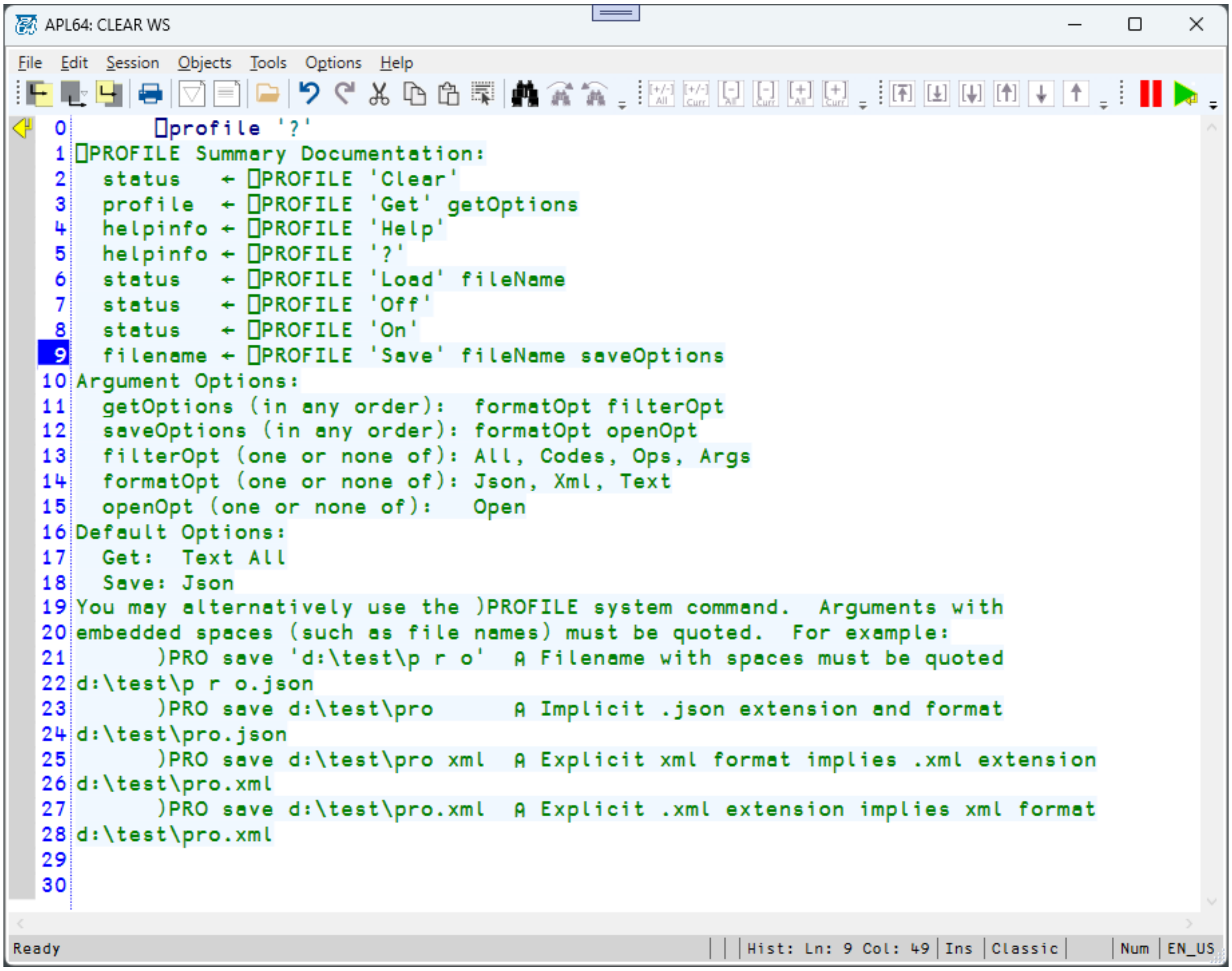

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 26 Col: 6 | Ins | Classic | Num | EN_US

```
      ⎕Profile 'On'
WAS: OFF NOW: ON
      X←⍳3 ⍝ APL64 programmer-selected actions to profile
      ⎕dr⎕←⎕Profile 'Get' 'Text'
Total 189.022ms #25
  XAQD1(Apply quad function monadically) 149.3281ms #4
    ⎕PROFILE (Monadic) 147.4039ms #2
      ⎕PROFILE AvalVector 147.4039ms #2
    ⎕DR (Monadic) 1.9242ms #2
      ⎕DR ZcVector 1.9242ms #2
  XCOMPILE(Compile APL statement => xcode) 36.7566ms #4
  XAPM(Apply monadic non-scalar primitive) 0.9702ms #1
    ⍳ (Monadic) 0.9702ms #1
      ⍳ IntScalar 0.9702ms #1
  XEOSID(End of FNIMM statement: pop stack and display value) 0.7069ms #3
  XTAGETSQ(Assign quad variable) 0.5562ms #2
    ⎕ ← 0.5562ms #2
      ⎕ ← ZcVector 0.5562ms #2
  XTAGETSV(Assign user variable) 0.3655ms #1
    X← 0.3655ms #1
      X← IntVector 0.3655ms #1
  XCHECK(Check valence of symbols) 0.2374ms #1
  XEOSIN(End of FNIMM statement: pop stack no display) 0.0972ms #1
  XFC(Fetch constant value and push onto stack) 0.0027ms #4
  XHALT 0.0012ms #4
82
```

Help

helpinfo ← ⎕Profile '?' or helpinfo ← ⎕Profile 'Help'

The Help action returns a character vector containing a summary of the ⎕Profile system function actions.



```
      ⎕profile '?'
⎕PROFILE Summary Documentation:
  status   ← ⎕PROFILE 'Clear'
  profile  ← ⎕PROFILE 'Get' getOptions
  helpinfo ← ⎕PROFILE 'Help'
  helpinfo ← ⎕PROFILE '?'
  status   ← ⎕PROFILE 'Load' fileName
  status   ← ⎕PROFILE 'Off'
  status   ← ⎕PROFILE 'On'
  filename ← ⎕PROFILE 'Save' fileName saveOptions
Argument Options:
  getOptions (in any order):  formatOpt filterOpt
  saveOptions (in any order): formatOpt openOpt
  filterOpt (one or none of): All, Codes, Ops, Args
  formatOpt (one or none of): Json, Xml, Text
  openOpt (one or none of):   Open
Default Options:
  Get:  Text All
  Save: Json
You may alternatively use the )PROFILE system command.  Arguments with
embedded spaces (such as file names) must be quoted.  For example:
      )PRO save 'd:\test\p r o'  ⍝ Filename with spaces must be quoted
d:\test\p r o.json
      )PRO save d:\test\pro      ⍝ Implicit .json extension and format
d:\test\pro.json
      )PRO save d:\test\pro xml  ⍝ Explicit xml format implies .xml extension
d:\test\pro.xml
      )PRO save d:\test\pro.xml  ⍝ Explicit .xml extension implies xml format
d:\test\pro.xml
```

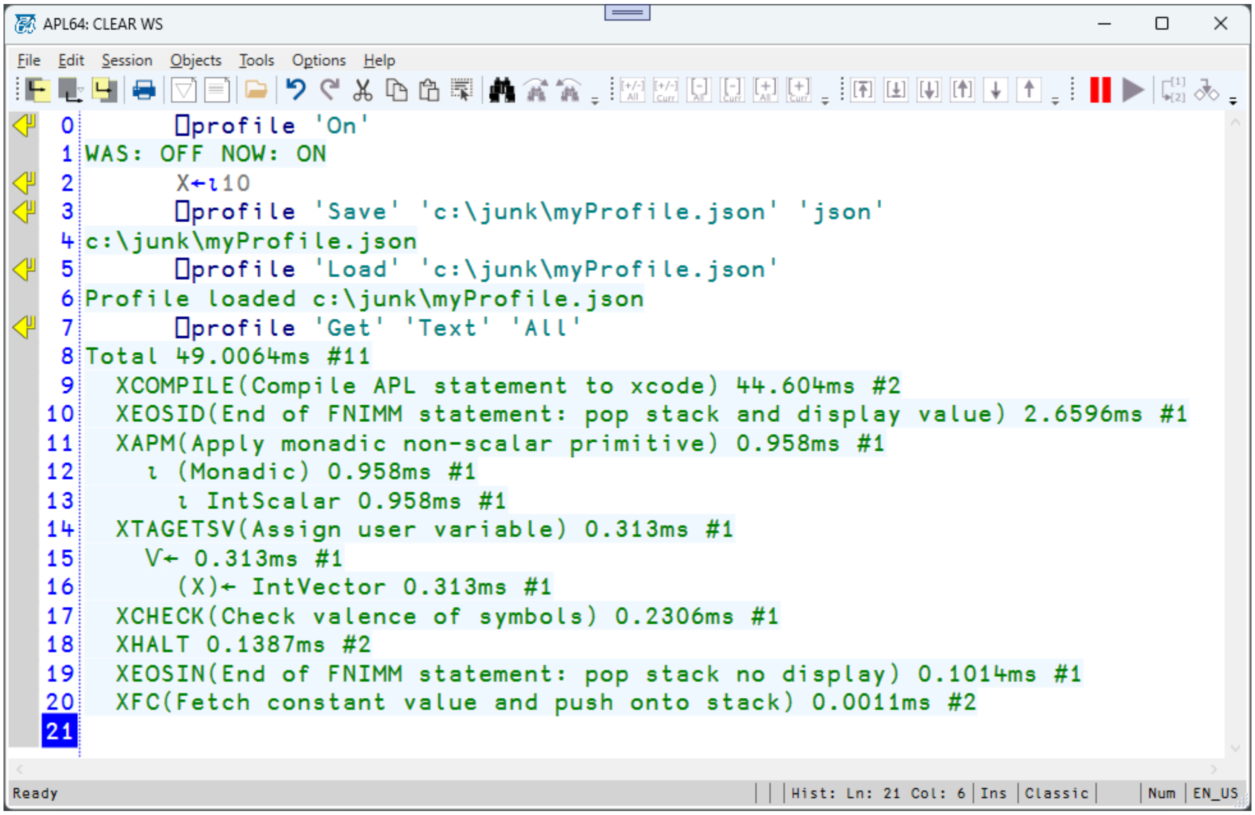
Load

status(Char[ ]) ← ⎕Profile 'Load' filename

fileName is the name of a previously-saved APL64 ⎕Profile record.

After loading a pre-existing ⎕Profile record:

- ⎕Profile 'Get' may be used to review the ⎕Profile record information
- ⎕Profile 'Off' is executed

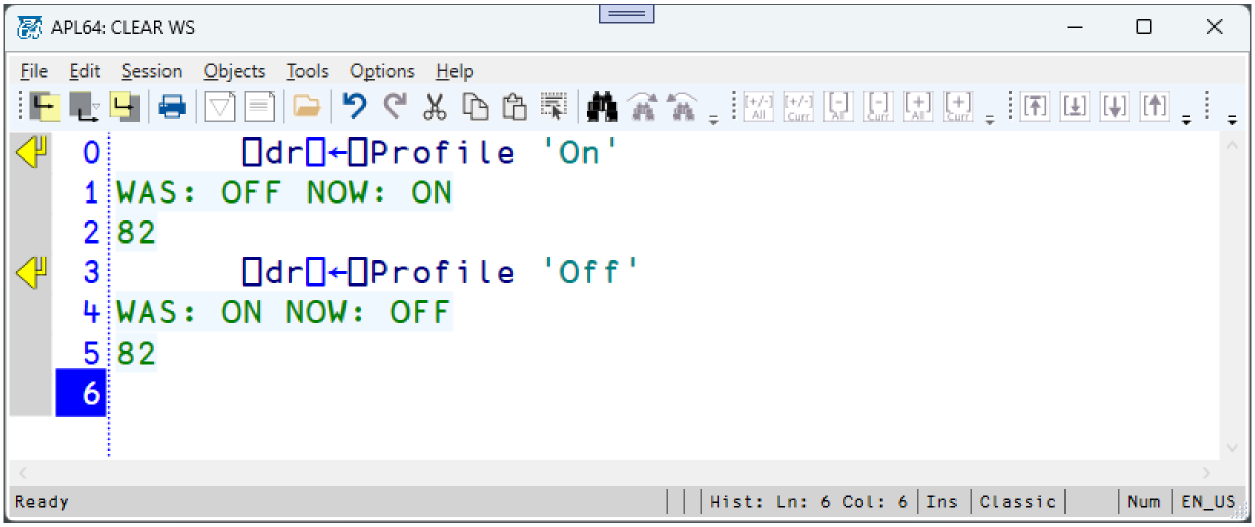


```
      ⎕profile 'On'
WAS: OFF NOW: ON
      X←⍳10
      ⎕profile 'Save' 'c:\junk\myProfile.json' 'json'
c:\junk\myProfile.json
      ⎕profile 'Load' 'c:\junk\myProfile.json'
Profile loaded c:\junk\myProfile.json
      ⎕profile 'Get' 'Text' 'All'
Total 49.0064ms #11
  XCOMPILE(Compile APL statement to xcode) 44.604ms #2
  XEOSID(End of FNIMM statement: pop stack and display value) 2.6596ms #1
  XAPM(Apply monadic non-scalar primitive) 0.958ms #1
    ⍳ (Monadic) 0.958ms #1
      ⍳ IntScalar 0.958ms #1
  XTAGETSV(Assign user variable) 0.313ms #1
    V← 0.313ms #1
      (X)← IntVector 0.313ms #1
  XCHECK(Check valence of symbols) 0.2306ms #1
  XHALT 0.1387ms #2
  XEOSIN(End of FNIMM statement: pop stack no display) 0.1014ms #1
  XFC(Fetch constant value and push onto stack) 0.0011ms #2
```

Off

Status(Char[ ]) ← ⎕Profile 'Off'

The Off action stops the recording of additional APL actions into the current performance profile.

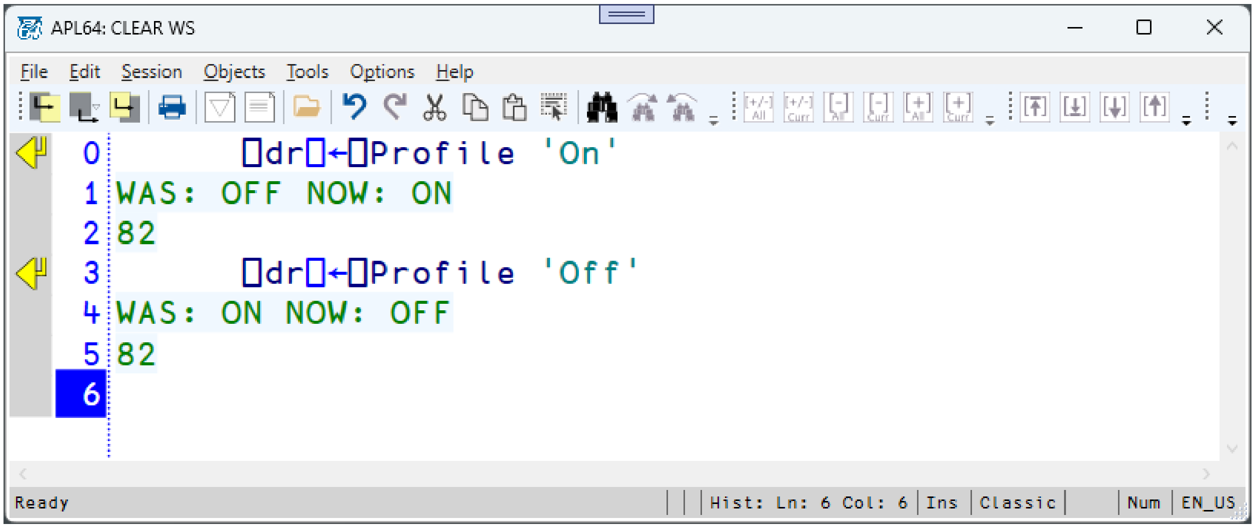


```
      ⎕dr⎕←⎕Profile 'On'
WAS: OFF NOW: ON
82
      ⎕dr⎕←⎕Profile 'Off'
WAS: ON NOW: OFF
82
```

On

Status(Char[ ]) ← ⎕Profile 'On'

The On action starts the recording of additional APL actions into the current performance profile.



```
      ⎕dr⎕←⎕Profile 'On'
WAS: OFF NOW: ON
82
      ⎕dr⎕←⎕Profile 'Off'
WAS: ON NOW: OFF
82
```

Save

filename(char[])←⎕Profile 'Save' targetFilename saveOptions

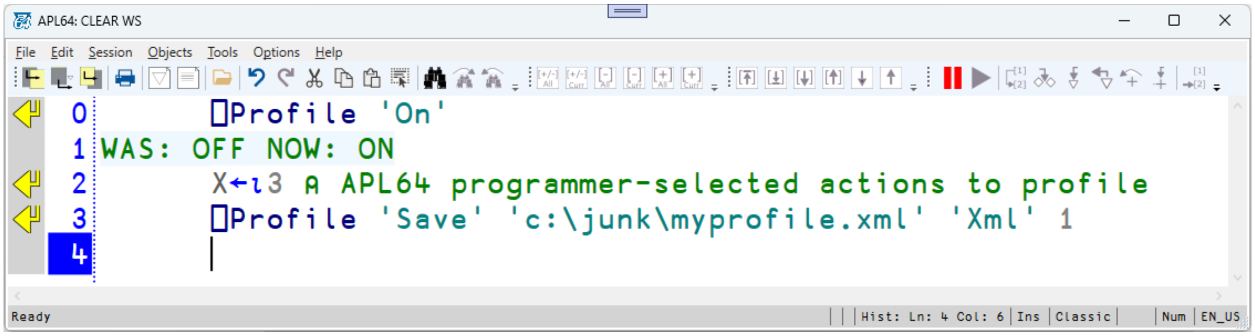
saveOptions (in any order): formatOpt, filterOpt

filterOptions : All, Codes, Ops, Args

formatOptions: Json(default), Xml, Text

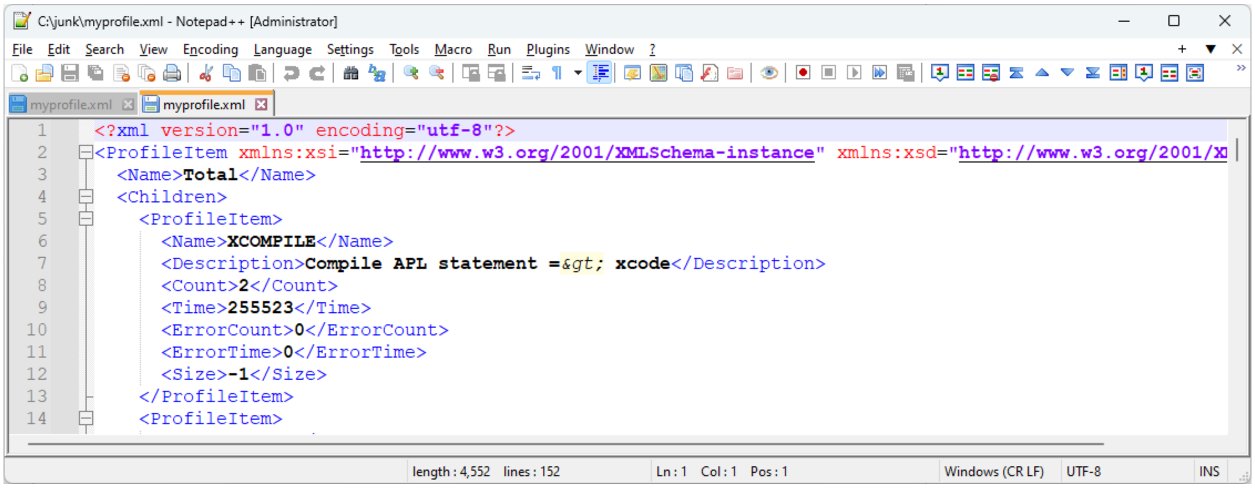
The Save action will prepare the the json- or xml-serialization or text summary of the performance profile, save it to the specified targetFilename, and optionally request the operating system to display the file content using the default viewer for the specified formatType.

formatType: Json, Text, Xml

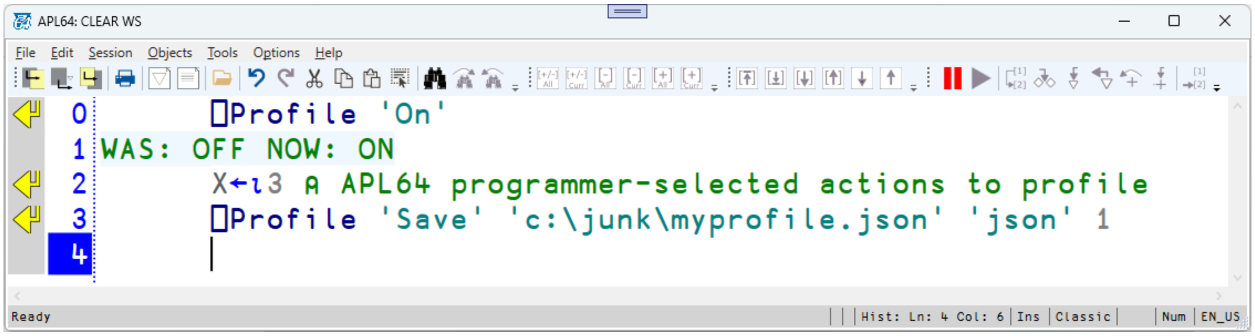


```
      ⎕Profile 'On'
WAS: OFF NOW: ON
      X←⍳3 ⍝ APL64 programmer-selected actions to profile
      ⎕Profile 'Save' 'c:\junk\myprofile.xml' 'Xml' 1
```

In this example the default Xml-format file viewer is Notepad++.

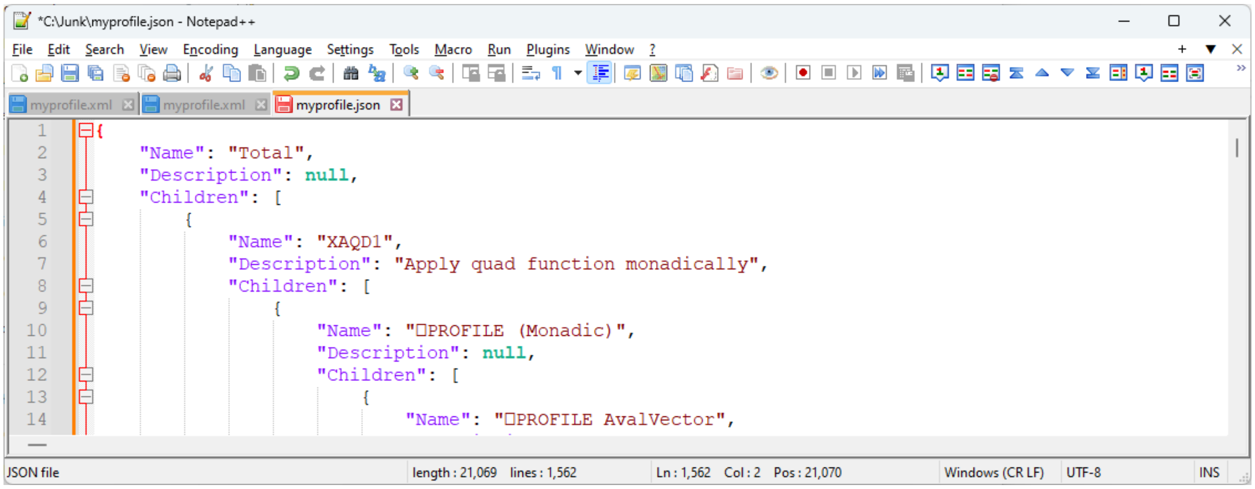

C:\junk\myprofile.xml - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
myprofile.xml
myprofile.xml
length : 4,552 lines : 152
Ln : 1 Col : 1 Pos : 1
Windows (CR LF)
UTF-8
INS

```
<?xml version="1.0" encoding="utf-8"?>
<ProfileItem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/X
  <Name>Total</Name>
  <Children>
    <ProfileItem>
      <Name>XCOMPILE</Name>
      <Description>Compile APL statement =&gt; xcode</Description>
      <Count>2</Count>
      <Time>255523</Time>
      <ErrorCount>0</ErrorCount>
      <ErrorTime>0</ErrorTime>
      <Size>-1</Size>
    </ProfileItem>
    <ProfileItem>
```

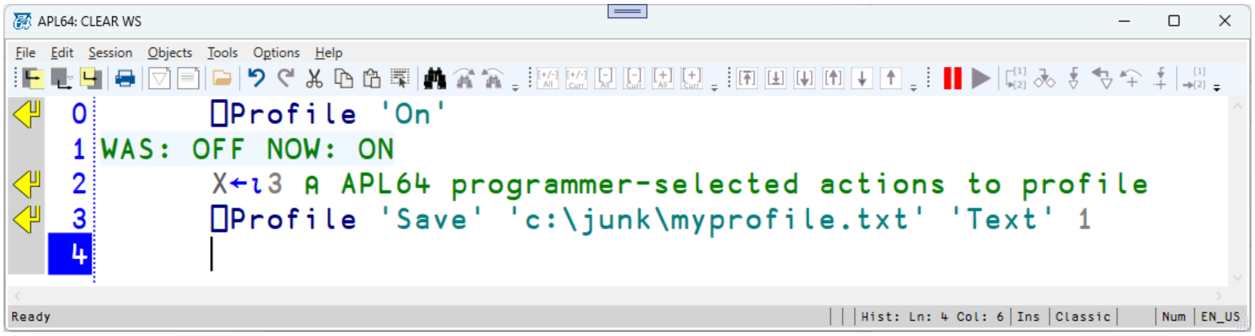


```
      ⎕Profile 'On'
WAS: OFF NOW: ON
      X←⍳3 ⍝ APL64 programmer-selected actions to profile
      ⎕Profile 'Save' 'c:\junk\myprofile.json' 'json' 1
```

In this example the default json-format file viewer is Notepad++.

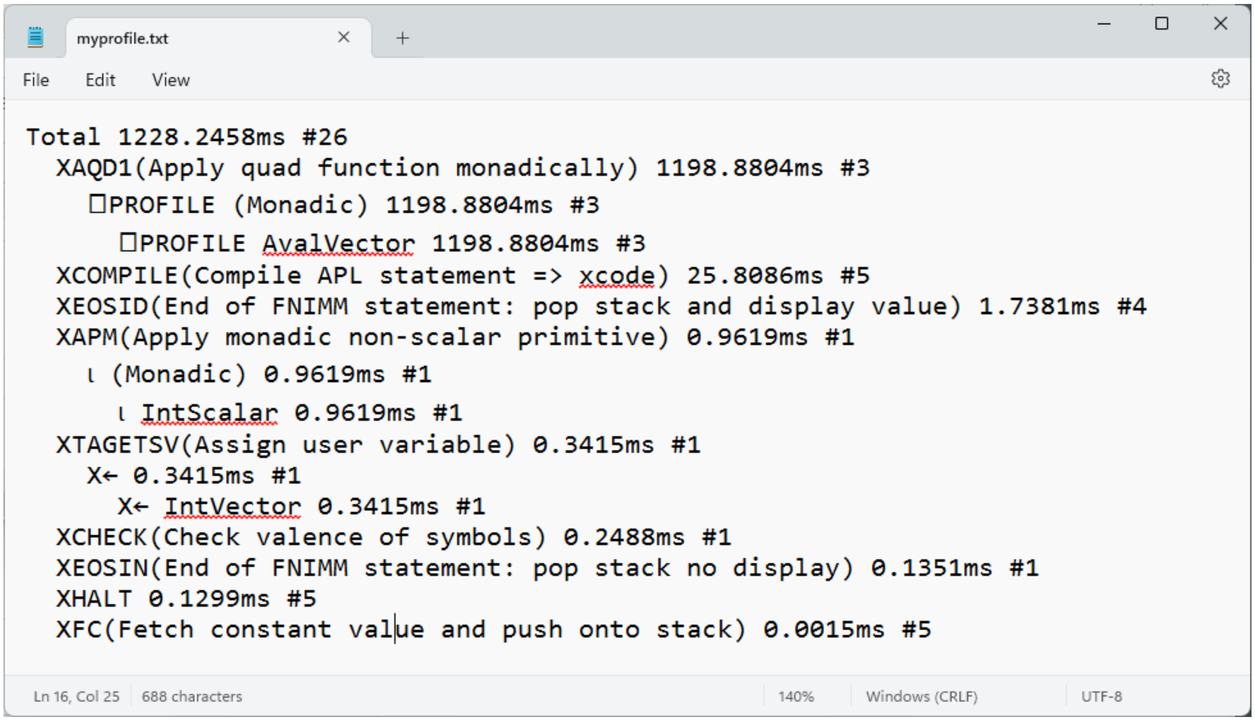

*C:\Junk\myprofile.json - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
myprofile.xml
myprofile.xml
myprofile.json
JSON file
length : 21,069 lines : 1,562
Ln : 1,562 Col : 2 Pos : 21,070
Windows (CR LF)
UTF-8
INS

```
{
    "Name": "Total",
    "Description": null,
    "Children": [
        {
            "Name": "XAQD1",
            "Description": "Apply quad function monadically",
            "Children": [
                {
                    "Name": "⎕PROFILE (Monadic)",
                    "Description": null,
                    "Children": [
                        {
                            "Name": "⎕PROFILE AvalVector",
```



```
      ⎕Profile 'On'
WAS: OFF NOW: ON
      X←⍳3 ⍝ APL64 programmer-selected actions to profile
      ⎕Profile 'Save' 'c:\junk\myprofile.txt' 'Text' 1
```

In this example the default text-format file viewer is Windows Notepad.



```
Total 1228.2458ms #26
  XAQD1(Apply quad function monadically) 1198.8804ms #3
    ⎕PROFILE (Monadic) 1198.8804ms #3
      ⎕PROFILE AvalVector 1198.8804ms #3
  XCOMPILE(Compile APL statement => xcode) 25.8086ms #5
  XEOSID(End of FNIMM statement: pop stack and display value) 1.7381ms #4
  XAPM(Apply monadic non-scalar primitive) 0.9619ms #1
    ⍳ (Monadic) 0.9619ms #1
      ⍳ IntScalar 0.9619ms #1
  XTAGETSV(Assign user variable) 0.3415ms #1
    X← 0.3415ms #1
      X← IntVector 0.3415ms #1
  XCHECK(Check valence of symbols) 0.2488ms #1
  XEOSIN(End of FNIMM statement: pop stack no display) 0.1351ms #1
  XHALT 0.1299ms #5
  XFC(Fetch constant value and push onto stack) 0.0015ms #5
```

⎕Profile Record Structure

A ⎕profile record class instance is a recursive structure containing these properties:

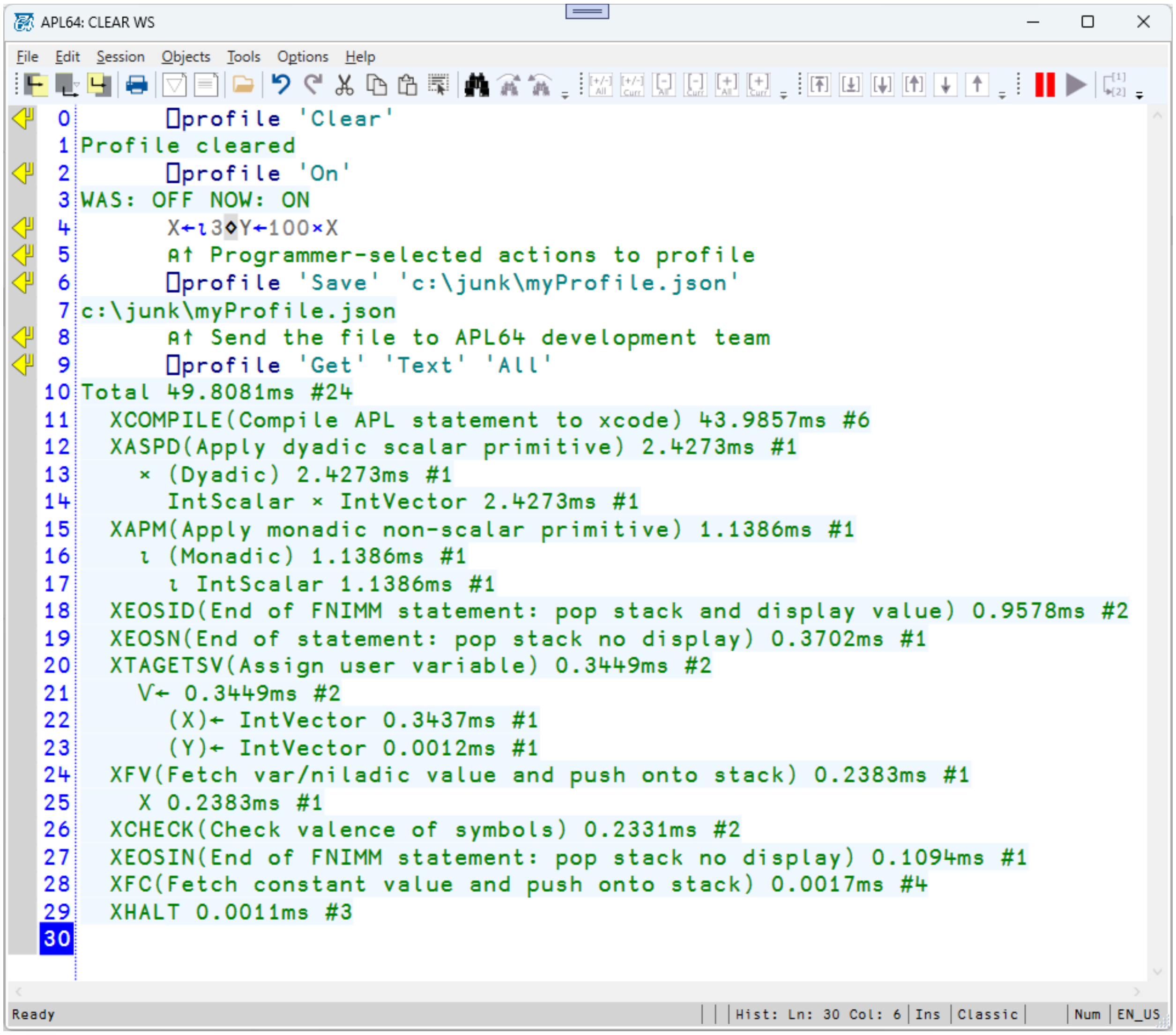
Property Name	Description
Name	Interpreter X-code action
Description	Description of X-code
Type	Interpreter type code
Count	#Actions of this type recorded
Time	Processing time (ms)
ScexCount	#Pre-compiled processing units
Scex Time	Scex Processing time (ms)
Error Count	#Processing exceptions
Error Time	Error processing time (ms)
Children	Subordinate ⎕profile records

Viewing a ⎕profile Record

Generally, a profile record is designed for analysis by members of the APL64 development team. Performance profile information may be transmitted to support@apl2000.com for further analysis, which may yield potential performance improvements in the APL64 interpreter.

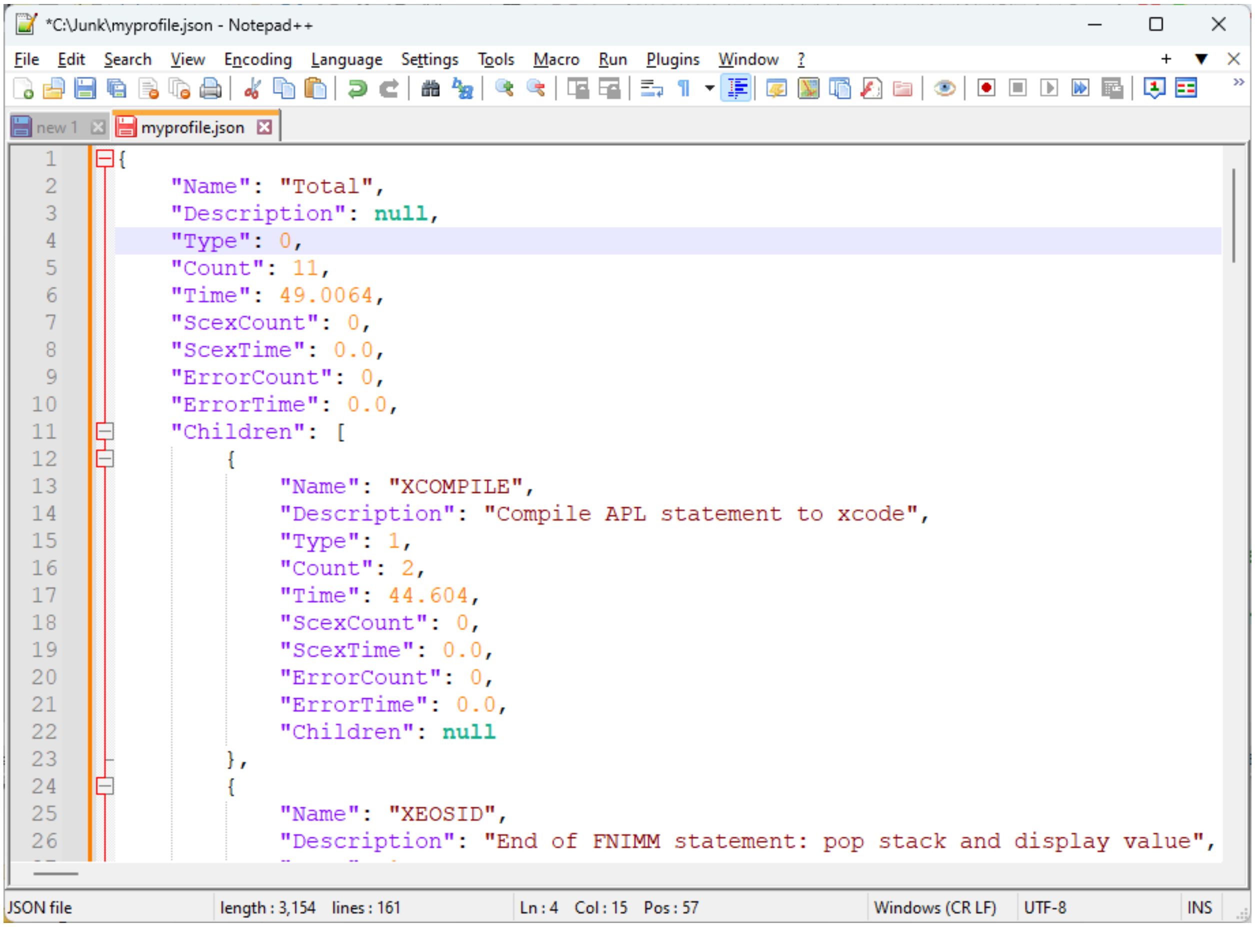
The 'Text' format option and the filter option (selected from: All, Codes, Ops, Args) is the most human-friendly format to view a ⎕profile record:

```
⎕profile 'Clear'
⎕profile 'On'
X←⍳3⋄Y←100×X
⍝↑ Programmer-selected actions to profile
⎕profile 'Save' 'c:\junk\myProfile.json'
⍝↑ Send the file to APL64 development team
⎕profile 'Get' 'Text' 'All'
```



```
      ⎕profile 'Clear'
Profile cleared
      ⎕profile 'On'
WAS: OFF NOW: ON
      X←⍳3⋄Y←100×X
      ⍝↑ Programmer-selected actions to profile
      ⎕profile 'Save' 'c:\junk\myProfile.json'
c:\junk\myProfile.json
      ⍝↑ Send the file to APL64 development team
      ⎕profile 'Get' 'Text' 'All'
Total 49.8081ms #24
  XCOMPILE(Compile APL statement to xcode) 43.9857ms #6
  XASPD(Apply dyadic scalar primitive) 2.4273ms #1
    × (Dyadic) 2.4273ms #1
      IntScalar × IntVector 2.4273ms #1
  XAPM(Apply monadic non-scalar primitive) 1.1386ms #1
    ⍳ (Monadic) 1.1386ms #1
      ⍳ IntScalar 1.1386ms #1
  XEOSID(End of FNIMM statement: pop stack and display value) 0.9578ms #2
  XEOSN(End of statement: pop stack no display) 0.3702ms #1
  XTAGETSV(Assign user variable) 0.3449ms #2
    V← 0.3449ms #2
      (X)← IntVector 0.3437ms #1
      (Y)← IntVector 0.0012ms #1
  XFV(Fetch var/niladic value and push onto stack) 0.2383ms #1
    X 0.2383ms #1
  XCHECK(Check valence of symbols) 0.2331ms #2
  XEOSIN(End of FNIMM statement: pop stack no display) 0.1094ms #1
  XFC(Fetch constant value and push onto stack) 0.0017ms #4
  XHALT 0.0011ms #3
```

To view a □profile record in serialized XML or Json file format the no-cost [Notepad++ Windows application]() is recommended. After installation of NotePad++ use the application's Plugins | Admin dialog to install the Json Viewer and XML Tools plug-ins. Both of these plug-ins have configuration options to expand the fields of a □profile record.



```
{
    "Name": "Total",
    "Description": null,
    "Type": 0,
    "Count": 11,
    "Time": 49.0064,
    "ScexCount": 0,
    "ScexTime": 0.0,
    "ErrorCount": 0,
    "ErrorTime": 0.0,
    "Children": [
        {
            "Name": "XCOMPILE",
            "Description": "Compile APL statement to xcode",
            "Type": 1,
            "Count": 2,
            "Time": 44.604,
            "ScexCount": 0,
            "ScexTime": 0.0,
            "ErrorCount": 0,
            "ErrorTime": 0.0,
            "Children": null
        },
        {
            "Name": "XEOSID",
            "Description": "End of FNIMM statement: pop stack and display value",
```