DYALOG

The tool of thought for expert programming

Dyalog™ for Windows

Spice command bar

Version 1.0

Dyalog Limited

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

tel: +44 (0)1256 830030
fax: +44 (0)1256 830031
email: support@dyalog.com
http://www.dyalog.com

Spice command bar

APL has always made it easy to handle programs as data, and so for programs to create, read or modify other programs. Programmers commonly use this to extend the development environment by writing programs to help them manage other programs: filing, searching, analysing and replacing expressions. SALT (Simple APL Library Toolkit) is an example.

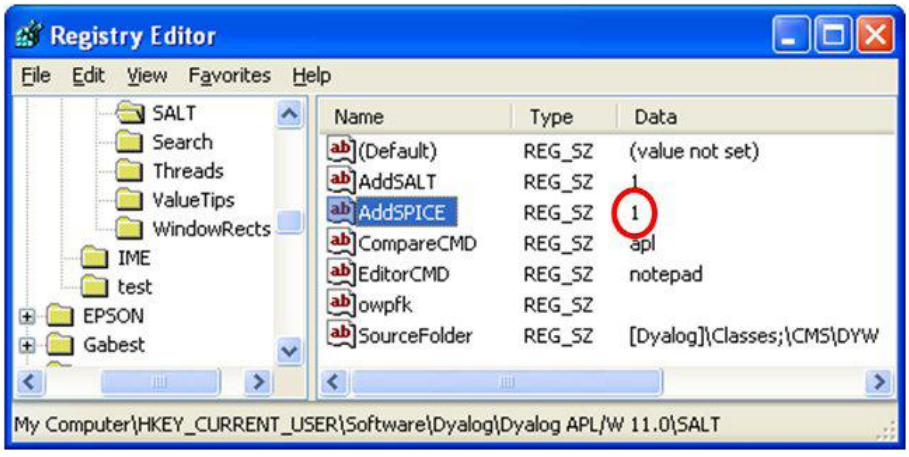
Spice extends your use of such development tools by providing a separate command line in the IDE. Now you can keep your development tools separate from your application environment.

Spice already has tools for SALT, SVN and other tasks plugged in to it. These might be all you need. Or you can plug your own development tools into Spice and invoke them from its command line.

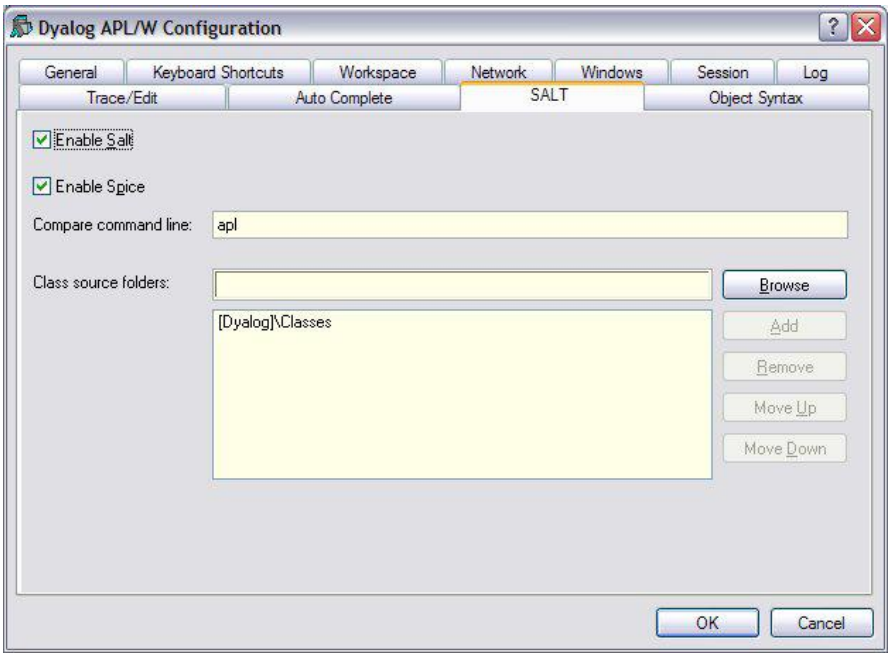
Turning Spice on

(To activate Spice, you must first activate SALT.)

In **Version 11**, edit the Registry to set the `AddSPICE` flag to 1.



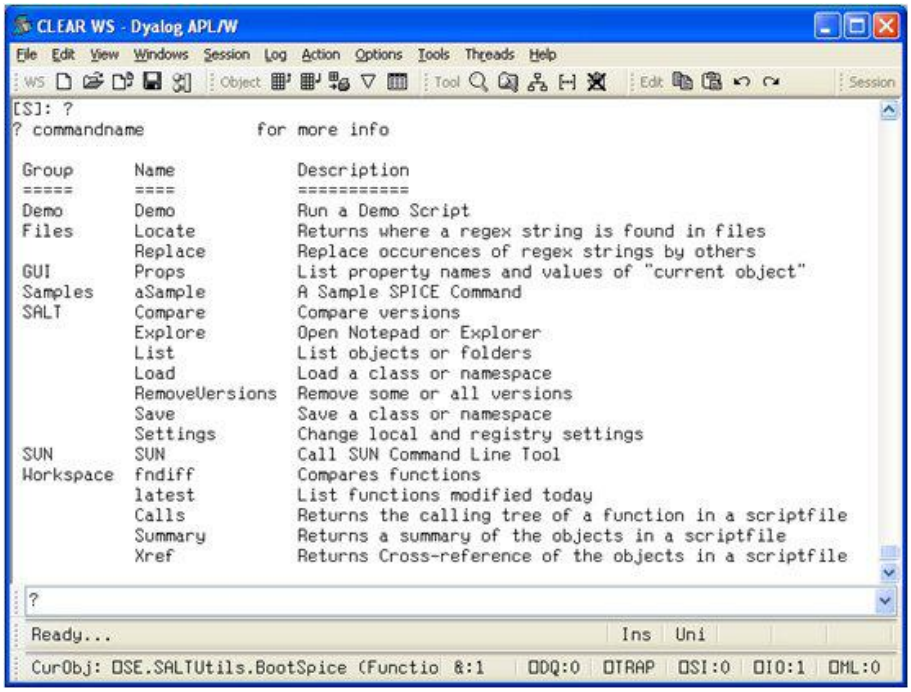
In **Version 12**, pick Configure from the Options menu. Select the SALT tab and check the Enable Spice box.



Using Spice

Click in the Spice command line at the bottom of the Dyalog window, or use **F2** to get there.

In the Spice command bar, type `?`. This will list the available commands in the session.



Typing `?` followed by a group name will list the commands in that group. For example,

```
? SALT
```

A `?` followed by a command name displays the help for that command. For example,

```
? Locate
```

Writes in the session log:

```
[S]: ? Locate
Help for command: Locate
Script location: C:\Program Files\Dyalog\Dyalog APL 12.0 …
Arg: regex string; shows where the string is found
```

So in the command line:

```
Locate blank
```

lists in the session log all the scripts in which the string `blank` occurs, and displays the script lines in which they are found.

```
Explore Spice\FnDiff
```

This opens the `Classes\Spice\FnDiff.dyalog` script in your preferred text editor.

```
Summary Utils\textUtils
```

This summarises the contents of the `textUtils` script.

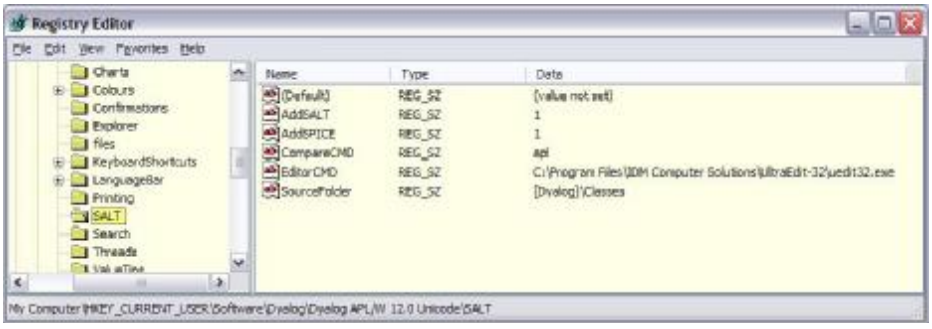
All the SALT functions are available as Spice commands, such as

```
save myclass file1
compare file1 -ver=4
list mine
settings
load myfile
removeversions filex -v=<9
```

Customising Spice

You can control the scope of the Spice commands from the SALT panel of the Configuration dialogue in the Options menu. Just edit the list of filepaths labelled Class source folders. (Close and restart Dyalog to activate your changes.)

The editor invoked by `Explore` is Notepad by default. You can specify an alternative by adding a string `EditorCMD` to the Registry.



Spice commands are defined in script files in the `Classes\Spice\` folder.

You can add your own scripts to this folder, using the script `Spice\aSample.dyalog` as a model. (On the Spice command line, try typing `Explore Spice\aSample.dyalog`.)

A Spice class script must contain at least the following public shared methods: `List`, `Run` and `Help`.

The `List` method takes no argument. It returns a list of namespaces corresponding to the commands supported by the class. Each namespace contains four character vectors, named as:

`Name` — the name of the command, a single word eg `Locate`

`Group` — the group it belongs to, a single word, eg `Files`

`Desc` — a sentence, such as `Find string in files`

`Parse` — the command syntax, eg `2 -exclude=`

Here is the `List` method from the `Spice\aSample` script.

```
∇ r←List
  :Access Shared Public
  r←⎕NS¨2⍴⊂'' ⍝ create 2 commands
  r.Name←'sampleA' 'sampleB' ⍝ their names
 ⍝ descriptions
  r[1].Desc←'A Sample SPICE Command without parsing'
  r[2].Desc←'A Sample SPICE Command with parsing'
  r.Group←⊂'Samples' ⍝ same group
 ⍝ Parsing rules for each:
 ⍝ 'sampleA' takes no rule
  r[1].Parse←''
 ⍝ 'sampleB' takes 1 argument
 ⍝ and a possible switch TZ which accepts a value
  r[2].Parse←'1 -TZ='
∇
```

The `Help` method takes one argument, the name of a supported command. Here is an example.

```
∇ r←Help Cmd
  :Access Shared Public
  :Select Cmd
  :Case 'sampleA'
      r←'This SPICE command "',Cmd
      r,←'" is for educational purposes'
  :Case 'sampleB'
      r←'This command is an example.'
      r,←' Its argument may only be "time" or "date".'
      r←↑r'It will display the time or the date'
  :EndSelect
∇
```

The `Run` method takes two arguments:

1. the name of the command, eg `'sampleB'`
2. a string or namespace, argument to the command

Here is an example.

```
∇ r←Run(Cmd Args);timezone;cmdparser
  :Access Shared Public
  :Select Cmd

  :Case 'sampleA'
      r←'This is an example of a command'
      r,←' that could do anything'

  :Case 'sampleB'
     ⍝ If parsing is required
     ⍝ (as per the 'Parse' variable above)
     ⍝ Spice will have performed the parsing
     ⍝ and the argument is a Parser instance.
     ⍝ See the Parser class for details.
     ⍝ Here we accept "time"
     ⍝ possibly followed by -TZ=a number

      :If 'time'≡1⊃Args.Arguments
         ⍝ default TZ to 'EST'
          timezone←'EST'Args.Switch'TZ'
          r←'The time is ',,'G<99:99>'⎕FMT 100⊥⎕TS[4 5]
          r,←', Time Zone=',timezone,' and all is well...'

      :ElseIf 'date'≡1⊃Args.Arguments
          r←'Today is ',⍕3↑⎕TS

      :Else
         ⍝ Here we could have other possibilities
          r←'You have entered an argument to ',Cmd
          r,←' which is not understood'

      :EndIf
  :EndSelect
∇
```

When Spice is started at the beginning of your Dyalog session, the `Setup` command is run.

This sets **F2** to switch focus to the Spice command line.

You can modify the Setup command by editing `Spice\Pers.dyalog`.