

AutoLISP® Strategies for CAD Managers

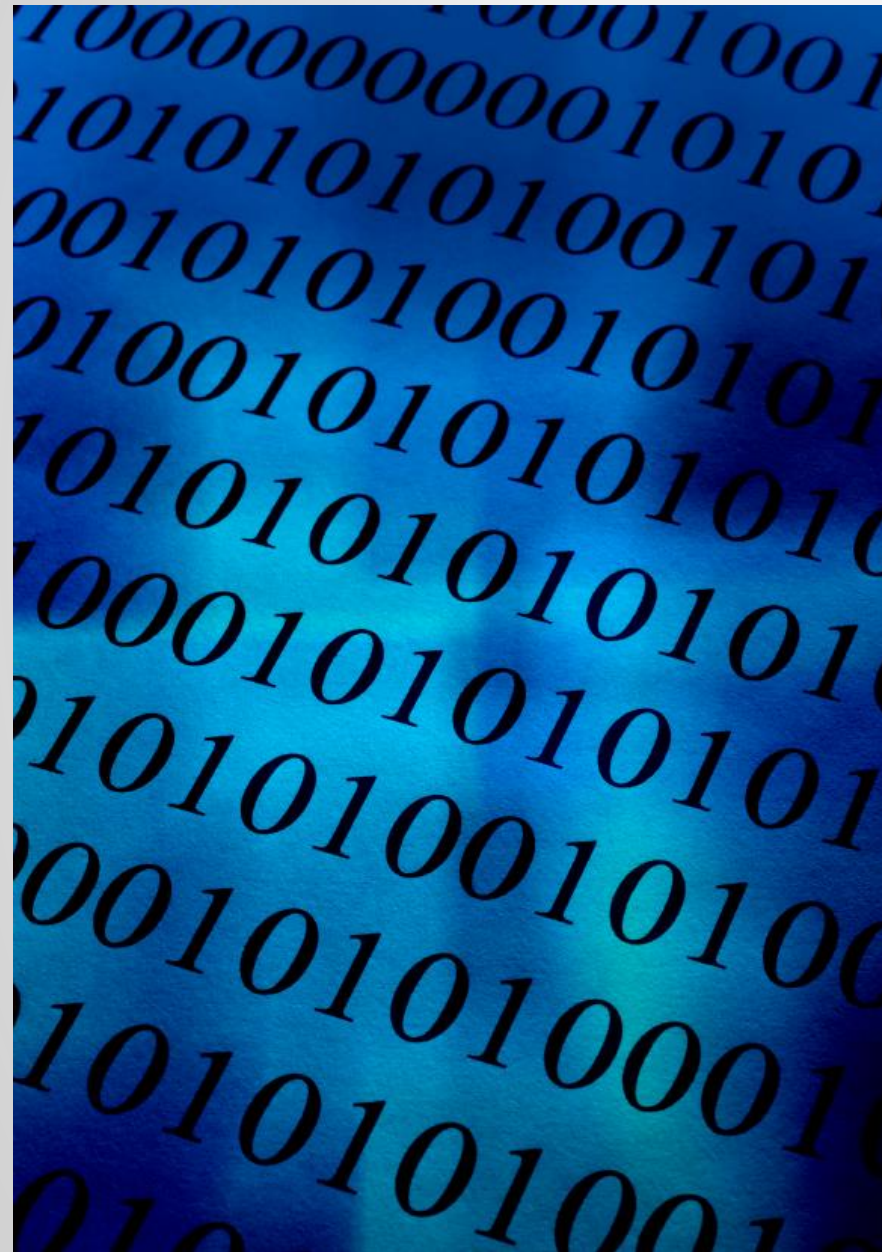


Robert Green

Robert Green Consulting

RGreen@GreenConsulting.com

Robert Green quick biography



Mechanical engineer turned CAD manager

Private consultant since 1991

Focus: CAD standards and management

Cadalyst Magazine author & contributing editor

21 year AU speaker

Facebook: **CAD Managers Unite!**

Twitter: **@RGCADMan**

Web site: **www.CAD-Manager.com**

Course Philosophy

You can manage AutoCAD based tools much more easily with this language. Every CAD manager should know it!

We'll start out simple while explaining some theory, speed up quickly, and finish strong. Follow along then refer to the course guide to find your own learning level then move on from there.

All examples are in the course guide.



Key Files and Variables

Defining some standard “rules” for
the purposes of our class.

Key files and why they matter ...

- **LSP**
- **Load in order**
- **What to mess with**
- **What not to!**



All those wacky file names ...

- **ACAD20XX.LSP** (system file – XX is version)
- **ACAD.LSP** (This is your global file)
- **ACAD20XXDOC.LSP** (system file - XX is version)
- **ACADDOC.LSP** (This is your per drawing file)
- **CUINAME.MNL** (loads with CUI of the same name)
- Residing in support folders.
- Load order is as listed above.

- So what can we conclude ...

Suggested approach ...

- Use ACADDOC.LSP to load your code
- Put in in the SUPPORT folder and start hacking
- Keep the file backed up
- If you mess up too bad, just delete!
- Later, after debugging, deploy to the network ...

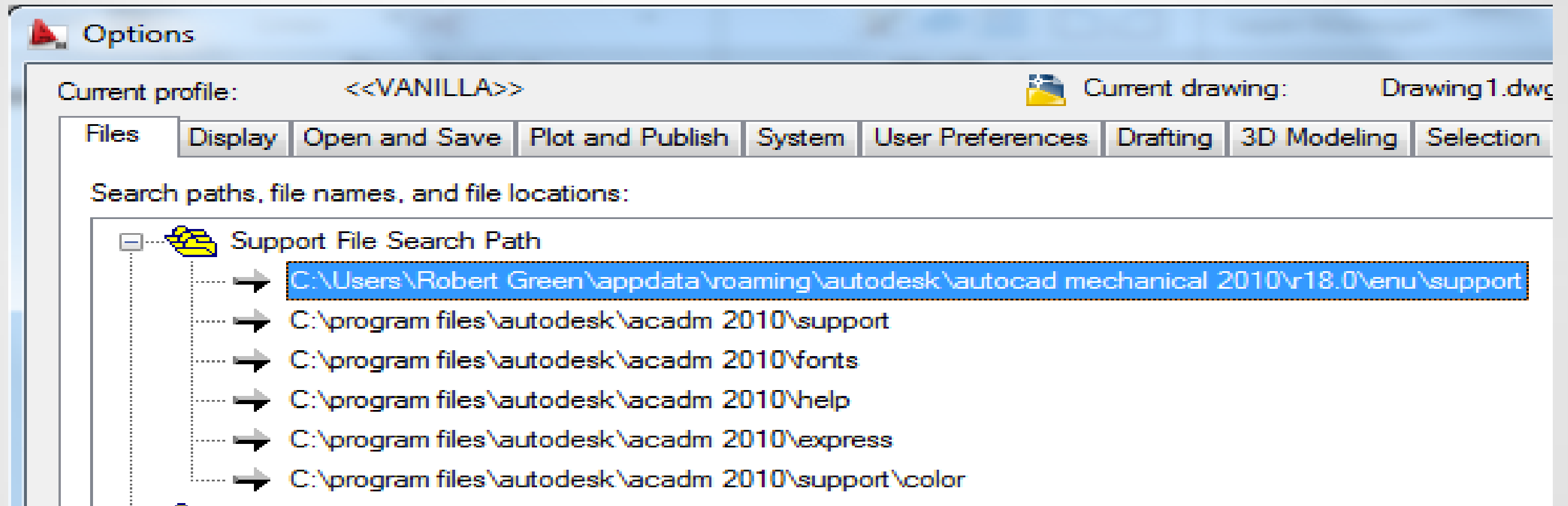
Make sure it works ...

- Create ACADDOC.LSP
- Load from network?
- Verify operation



Find the support folder ...

- Use OPTIONS to find the folders ...



Please Note

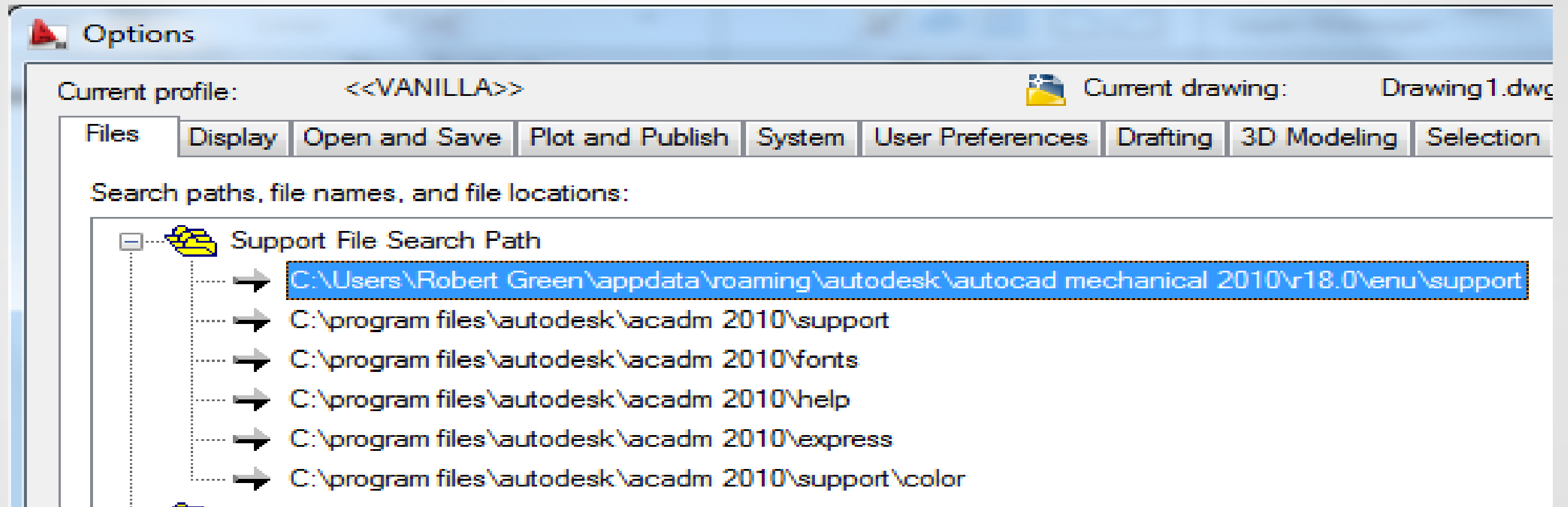
Create a New Profile!

You can use it for testing without disturbing the default (just in case you mess up!)



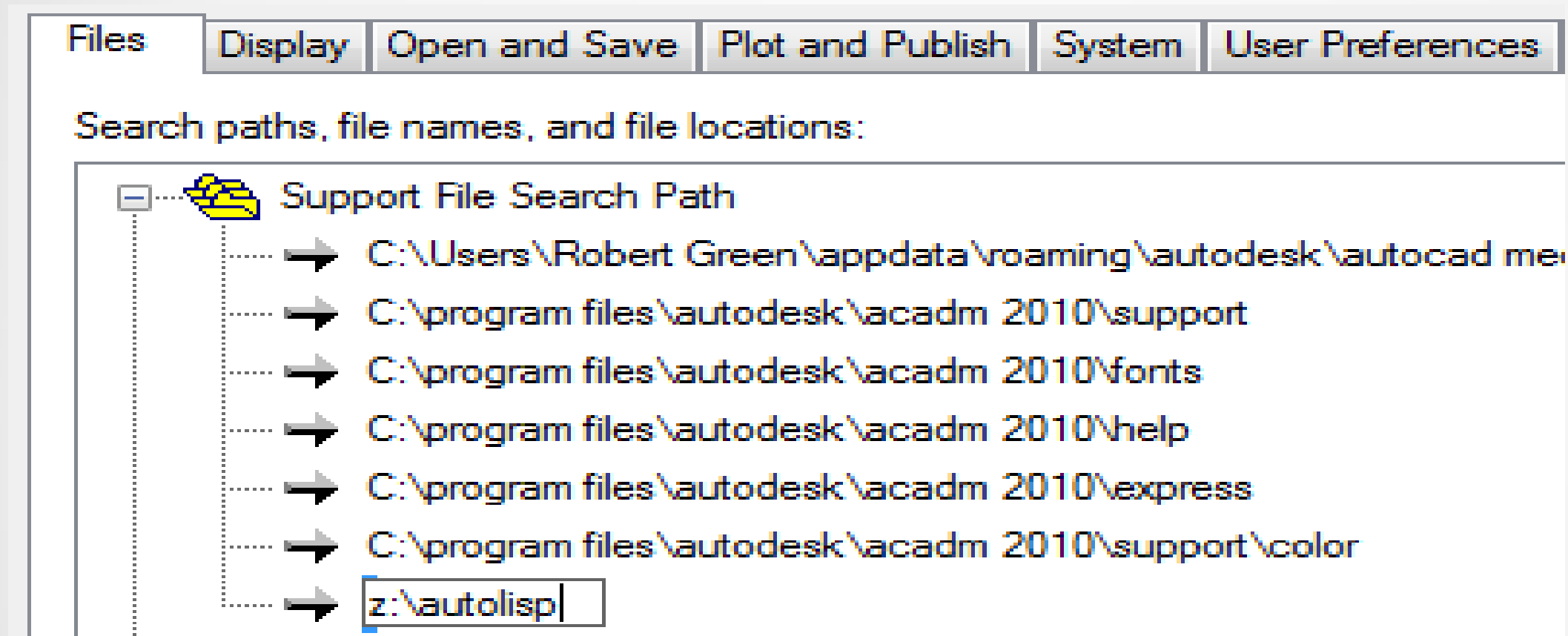
Find the support folder ...

- Use OPTIONS to find the folders ...



Add a network folder?

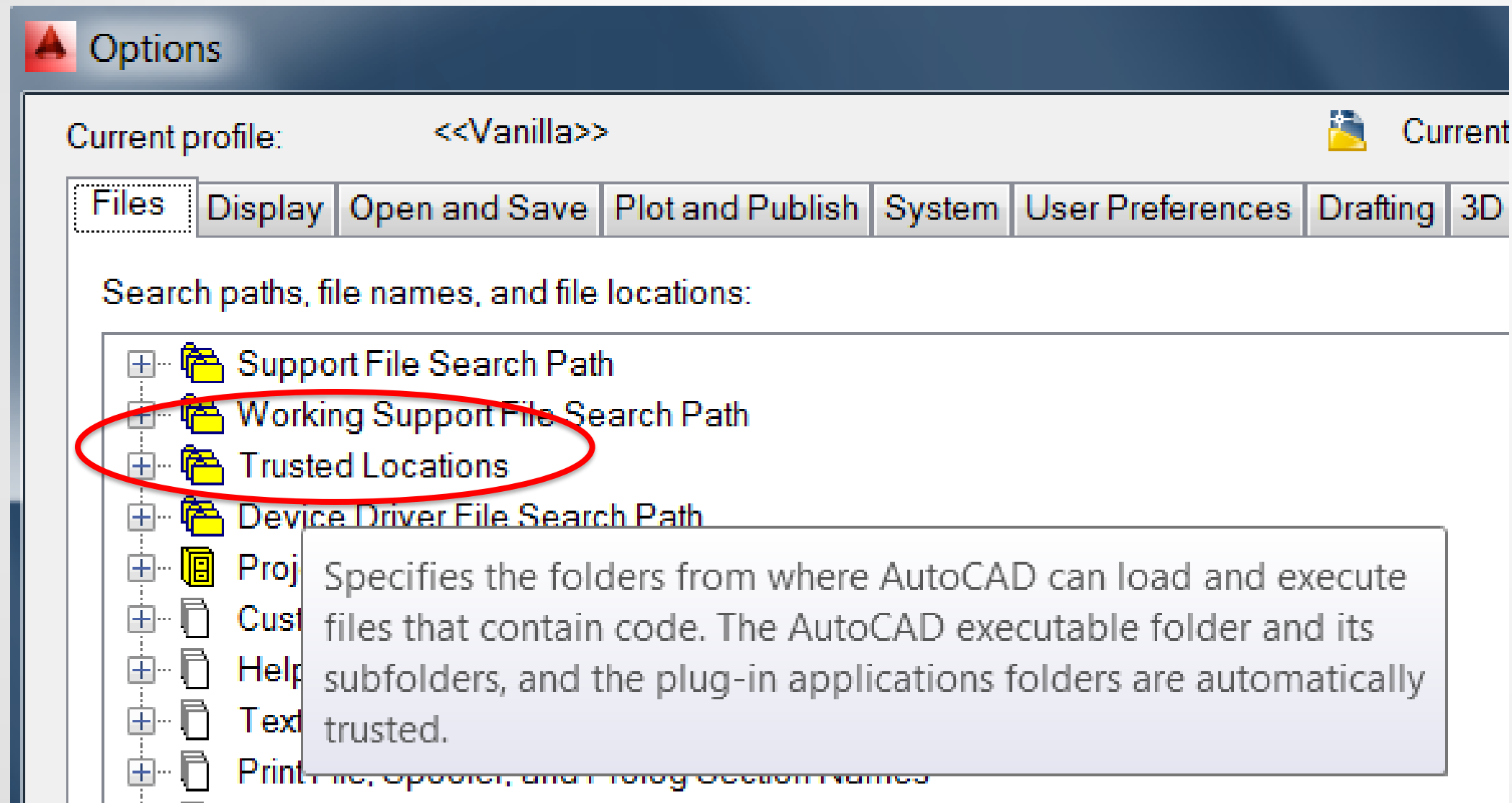
- Use **OPTIONS** to add it ...



- Push the network path to the top of the list

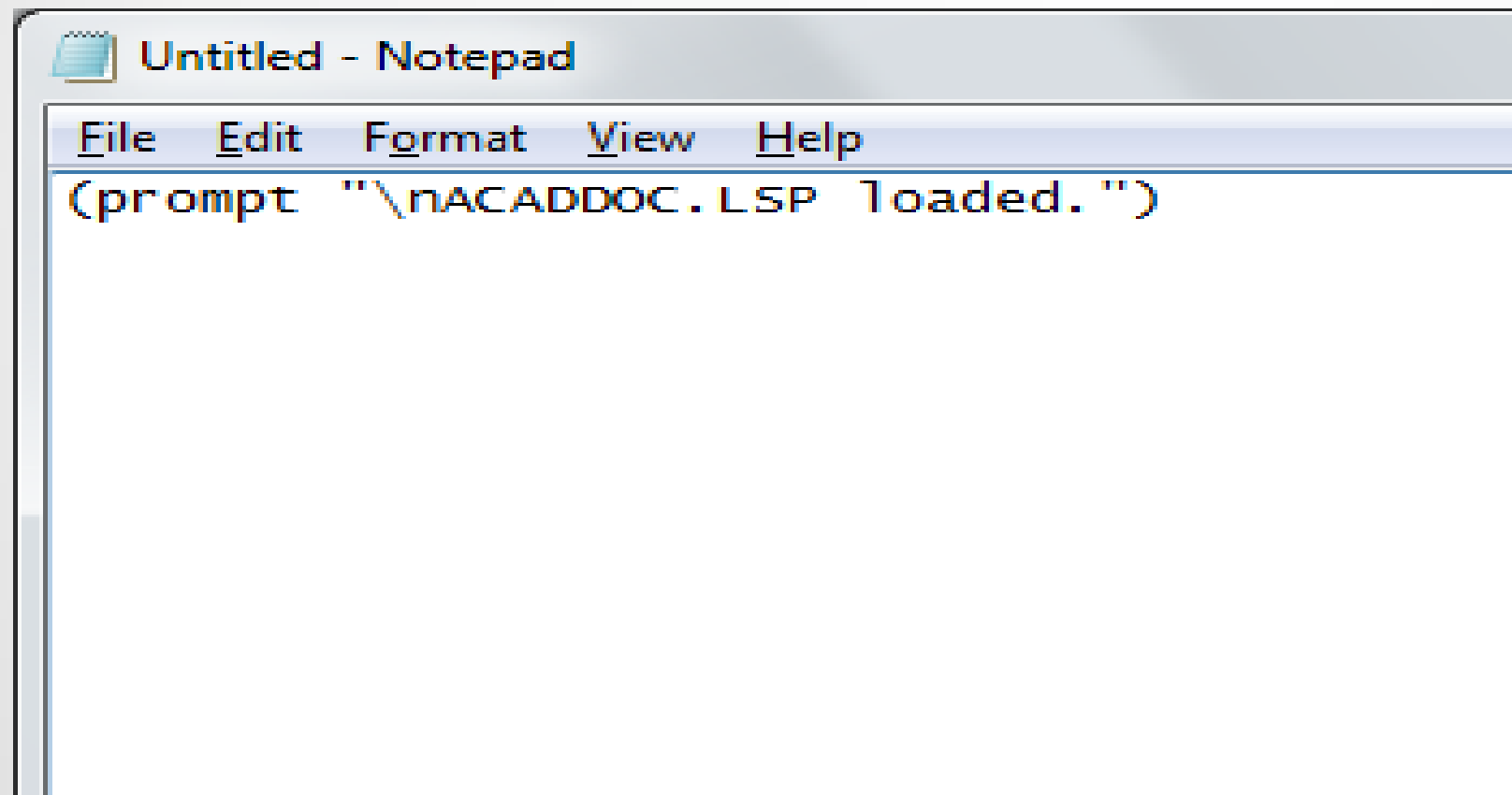
Set trusted locations...

- Use **OPTIONS** to set the folder ...



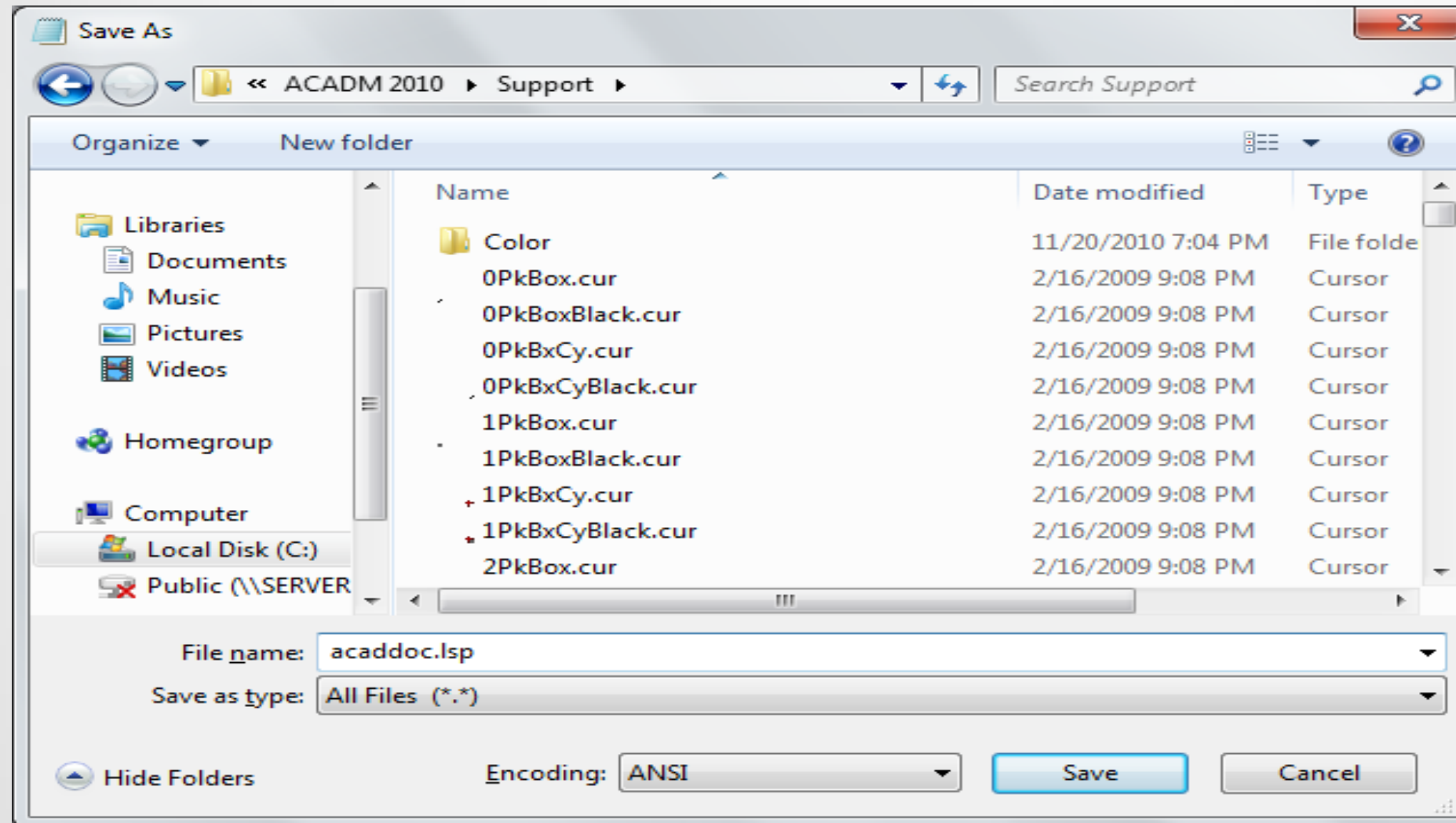
Create the file ...

- Use Notepad – not Word!
- You may also use Visual LISP VLIDE if you prefer
- Use (prompt "\nACADDOC.LSP loaded.") as text



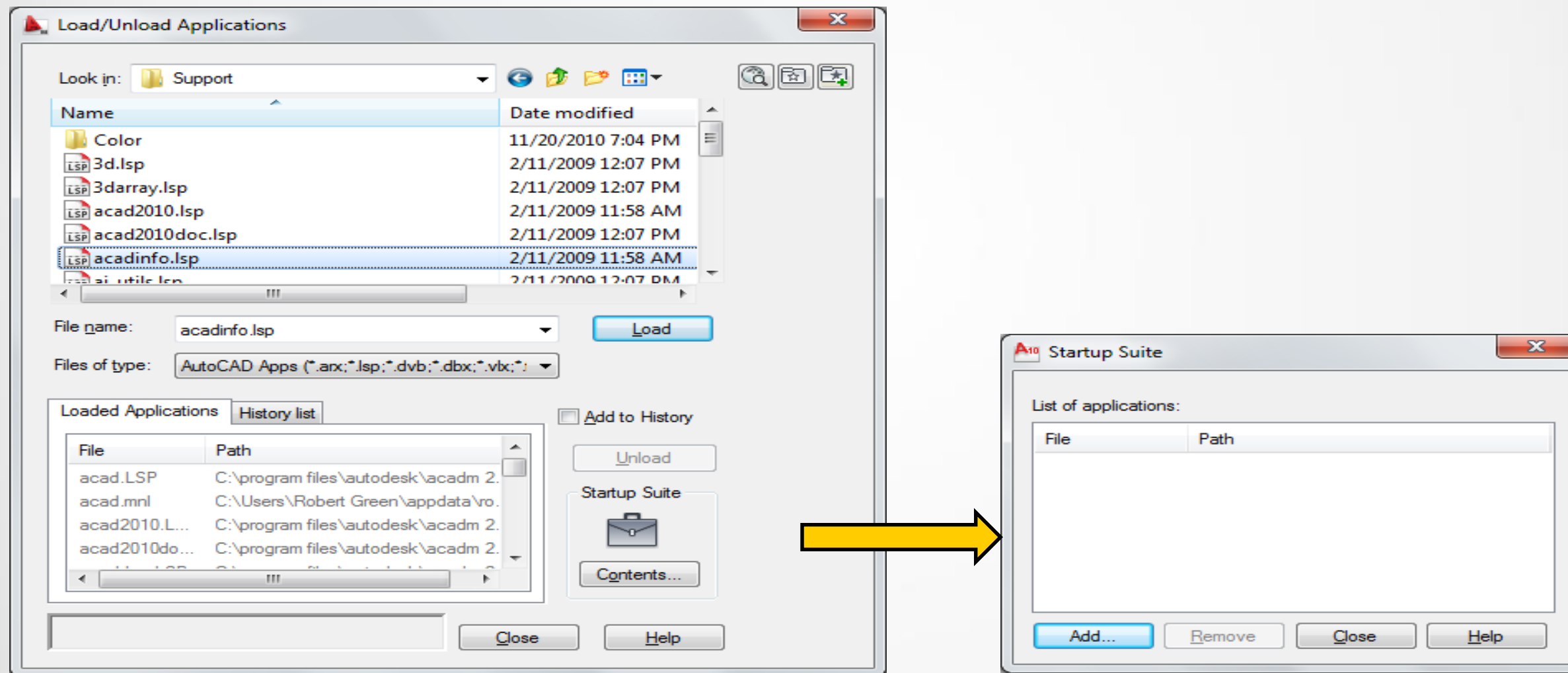
Save the file ...

- To the SUPPORT folder
- Use ACADDOC.LSP as the name



Alternately ...

- You can use APPLOAD to load files
- You can use STARTUP SUITE to load at each start



Syntax Examples

Controlling Variables

Using SETVAR/GETVAR functions

SETVAR (SET VARIABLE)

- To set the system variable for the fillet radius to 0.25 do this:
(setvar "filletrad" 0.25)
- To set the system variable for expert status do this:
(setvar "expert" 5)
- To set the system variable for dimension style do this:
(setvar "dimstyle" "am_ansi")
- **GETVAR** simply **"gets"** rather than **"sets"** variables ...

Command line access

- (command "viewres" "y" "5000")
- (command "-color" "BYLAYER")
- (command "-linetype" "set" "BYLAYER" "")
- (command "menu" "menuname.cuix")
- Do you see a way to enforce standards ...



SETQ (SET eQual)

- To create the variable MYNAME and set it with your proper name as a value do this:
(setq myname "Robert Green")
- To store the currently logged in user to the variable MYNAME do this:
(setq myname (getvar "loginname"))
- To store a custom path do this:
(setq lisp_path "z:\\lisp")



Syntax Examples

Custom functions that add to the command set

User functions

- **Speed for the user**
- **Lower support for you**
- **A win-win scenario**
- **Let's put everything we've learned into action to build some functions.**

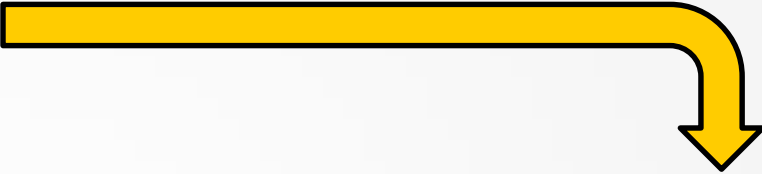


User Function Examples

```
(defun C:ZA ()  
  (command ".zoom" "a")  
  (princ)  
)
```

User Function Examples

```
(defun C:ZA ()  
  (command ".zoom" "a")  
  (princ)  
)
```



```
acaddoc.lsp - Notepad  
File Edit Format View Help  
(defun C:ZA ()  
  (command ".zoom" "a")  
  (princ)  
)  
  
(prompt "\nACADDOC.LSP loaded.")
```


Auto Purge Function

Auto Purge

```
(defun c:atp ()  
  (command "-purge" "a" "*" "n" ".qsave")  
  (princ)  
)
```

```
(defun c:atp ()  
  (command "-purge" "b" "*" "n" ".qsave")  
  (princ)  
)
```

Note – form commands for command line access



User Function Examples

```
(defun C:VR ()  
  (command "viewres" "y" "5000")  
  (princ)  
)
```

```
(defun C:BL ()  
  (command "-color" "BYLAYER")  
  (command "-linetype" "set" "BYLAYER" "")  
  (princ)  
)
```

Fillet Zero Function

Fillet Zero

```
(defun c:fz ()  
  (setvar "filletrad" 0.0)  
  (command ".fillet" pause pause)  
  (princ)  
)
```

What have I not done in this function?

What does PAUSE do?

Improved Fillet Zero

```
(defun c:fz ()  
  (setq old_filletrad (getvar "filletrad"))  
  (setvar "filletrad" 0.0)  
  (command ".fillet" pause pause)  
  (setvar "filletrad" old_filletrad)  
  (princ)  
)
```

*** Note how we store and recall the FILLETRAD so the function puts things back the way they were!**

Syntax Examples

Calling external programs

Call outside programs ...

- To invoke a browser do this:

(command “browser” “file goes here”)

- To invoke an app:

(startapp “C:\\folder\\progrname.exe”)

From a function it looks like this ...

```
(defun c:np ()  
  (startapp "notepad.exe")  
  (princ)  
)
```

```
(defun c:myprog ()  
  (startapp "c:\\progrpath\\myprogram.exe")  
  (princ)  
)
```


Syntax Examples

Error Handlers

What's the problem with FZ?

```
(defun c:fz ()  
  (setq old_filletrad (getvar "filletrad"))  
  (setvar "filletrad" 0.0)  
  (command ".fillet" pause pause)  
  (setvar "filletrad" old_filletrad)  
  (princ)  
)
```

* Hint – what could the user do?

Step 1: Create the error handler

```
(defun *fz_error* (msg)
  (if (/= msg "Function cancelled")
    (princ (strcat "\nError: " msg "\n"))
    (princ)
  )

  (if old_filletrad (setvar " filletrad " old_filletrad))
  (if *old_error* (setq *error* *old_error*)) ; note!
  (princ)
)
```

Step 2: Modify FZ to use error handler

```
(defun c:fz ()  
  (setq old_filletrad (getvar "filletrad"))  
  (setq *old_error* *error*)           ; *error* is the default  
  (setq *error* *fz_error*)           ; *fz_error* is now default  
  (setvar "filletrad" 0.0)  
  (command ".fillet" pause pause)  
  (setvar "filletrad" old_filletrad)  
  (setq *error* *old_error*)           ; default restored  
  (princ)  
)
```



Step 3: Does the IF now make sense?

```
(defun *fz_error* (msg)
  (if (/= msg "Function cancelled")
    (princ (strcat "\nError: " msg "\n"))
    (princ)
  )
  (if old_filletrad (setvar " filletrad " old_filletrad))
  (if *old_error* (setq *error* *old_error*)) ; Note!
  (princ)
)
```

Summing up ...

- **Create an error handler for each routine**
- **Reference the error handler in each routine**
- **Be sure the error routine fixes any system variables use**

▶ Now you can create routines that are well behaved no matter how the user exits the program.

Syntax Examples

Undefined commands to subtract
from the command set

Undefining ...

- (command “.undefine” “LINE”)
- (command “.undefine” “TORUS”)
- Don't want users messing with a command?
- Just undefine it ...

► Now you can SUBTRACT from the AutoCAD Command set in your ACADDOC.LSP file.

The DOT form ...

- Invoke commands like this: `.LINE`
- Note the dot “.” character?
- This allows you to invoke a command whether it has been undefined or not!
- This is our little secret right ...

Syntax Examples

Redefining commands that have been undefined

Redefining ...

- (command “.redefine” “LINE”)
- (command “.redefine” “TORUS”)
- Want to be sure that a command is active?
- Just redefine it ...

► Now you can UNSUBTRACT from the AutoCAD Command set with ease.

Undefining revisited ...

- What if your users find out about REDEFINE and start REDEFINING your UNDEFINES?
- Just undefine the redefine like this:
- (command “.undefine” “REDEFINE”)
- That’ll stop users from redefining ...

Syntax Examples

Undefined + Functions

to change the way commands operate

Alerting the user ...

- You can send a message to the user like this:

(alert “Message goes here”)



Redefining ...

- You can undefine a command and redefine it like this:

```
(command ".undefine" "TORUS")
```

```
(defun C:TORUS ()  
  (alert "Don't use that command!")  
  (princ)  
)
```

▶ Now you do whatever you want!

What Does This Do?

(command “.undefine” “QSAVE”)

```
(defun c:qsave ()  
  (command “-purge” “b” “*” “n”)  
  (command “.qsave”)  
  (princ)  
)
```

Command echo (CMDECHO)

Run in STEALTH mode like this:

```
(defun C:QSAVE ()  
  (setvar "cmdecho" 0)  
  (command "-purge" "b" "*" "n")  
  (command ".qsave")  
  (setvar "cmdecho" 1)  
  (princ)  
)
```

* CMDECHO is on or off so no need to store it's value

Syntax Examples

CUI Manipulation/Loading

Load a CUI like this ...

(command “menu” “c:\\path\\cuiname.cuix”)

- **Note: Not MENULOAD but MENU ...**

Load a workspace from the CUI like this ...

(command “-workspace” “workspacename”)

Supporting Power Users

LSP/CUI Manipulation/Loading

When power users need freedom

Give them “backdoor” way to load their own LISP or CUI files.

However, make sure that the backdoor is standard!

Let’s see how ...

USER.LSP

Agree that all LSP files for the power user will reside in a file called USER.LSP

Detect the file, if there, and load it.

The power is totally responsible for the file!

USER.LSP

```
(if (findfile "user.lsp")  
    (load "user.lsp")  
)
```

USER.CUIx

We'll use the same trick for CUI files:

```
(if (findfile "user.cuix")  
  (command "menu" "user.cuix")  
)
```

USER.CUIx + Workspace

Now we can load the CUI file and set a current workspace.
Note the use of PROGN:

```
(if (findfile "user.cuix")  
  (progn  
    (command "menu" "user.cuix")  
    (command "-workspace" "user")  
  )  
)
```

Network Loading

ACADDOC.LSP local loads a network INIT.LSP

Wouldn't it be great if ...

We could maintain all our code on a server.

Go to the user's machine just once and set it so it loads from the server?

Let's see how ...

ACADDOC.LSP on user's machine

(setq lisp_path "X:\\\\AUTOLISP\\") ; sets the path

**(if (findfile (strcat lisp_path "init.lsp"))
 (load (strcat lisp_path "init.lsp"))
)**

Have a different LISP_PATH for each AutoCAD version!

INIT.LSP

This is the file on the network drive

It contains all the SETVAR, functions, commands that we've talked about to this point.

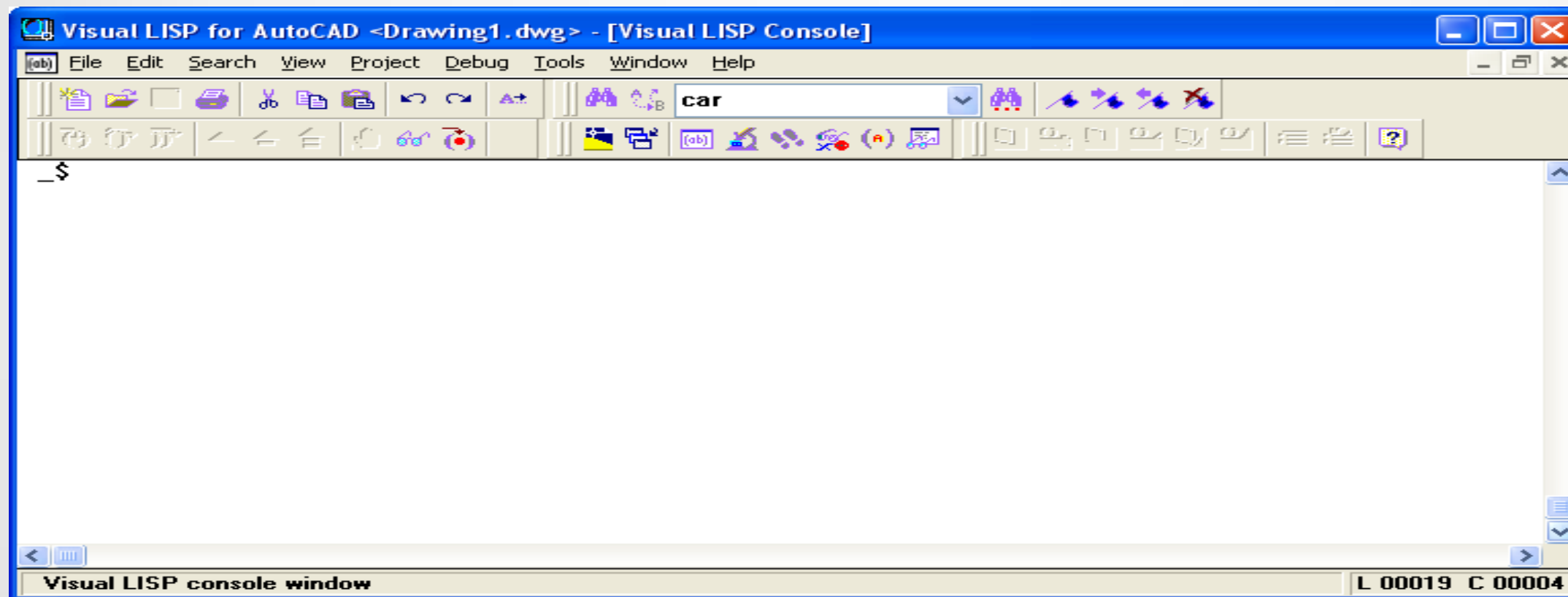
Now you can update everybody very easily

VLIDE

Compiling/writing code

VLIDE environment ...

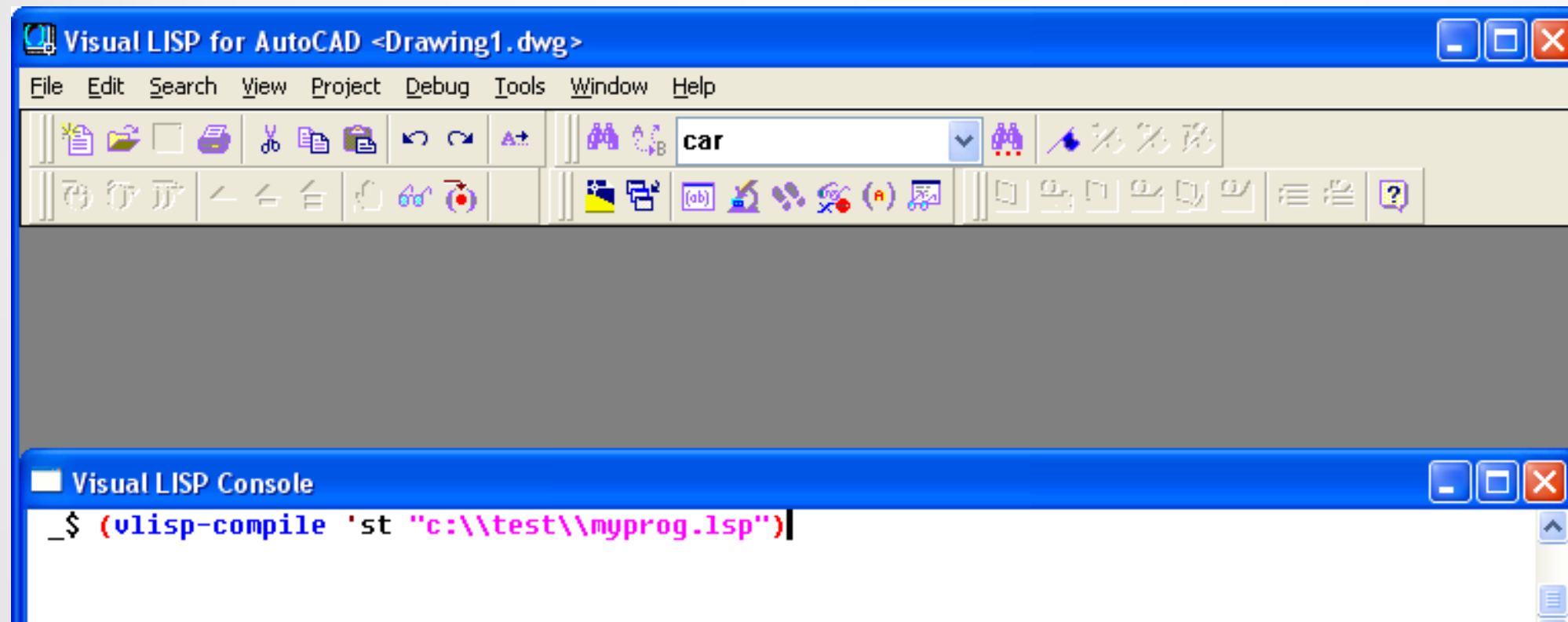
- You can write code in the VLIDE window like this:



Compile your code ...

- Use the VLIDE environment like this:

(vlisp-compile 'st "c:\\test\\myprog.lsp")



- You'll get MYPROG.FAS as a result ...

Load compiled code ...

- Use a LOAD statement like this:
`(load "c:\\test\\myprog.fas")`
- Now your LSP code is secure!
- Be sure not to lose your source LSP file though!

Bypassing Profiles

LSP/CUI Manipulation/Loading

Wouldn't it be great if ...

You could set printer directories, palette directories, etc, without using profiles?

AutoLISP gives us a powerful, yet easy, set of tools to do so.

Lets see how ...

GETENV and SETENV

These functions read (GET) or write (SET) profile values in the user's current active profile.

What does this mean for CAD managers?

It means we no longer care which profile is current!

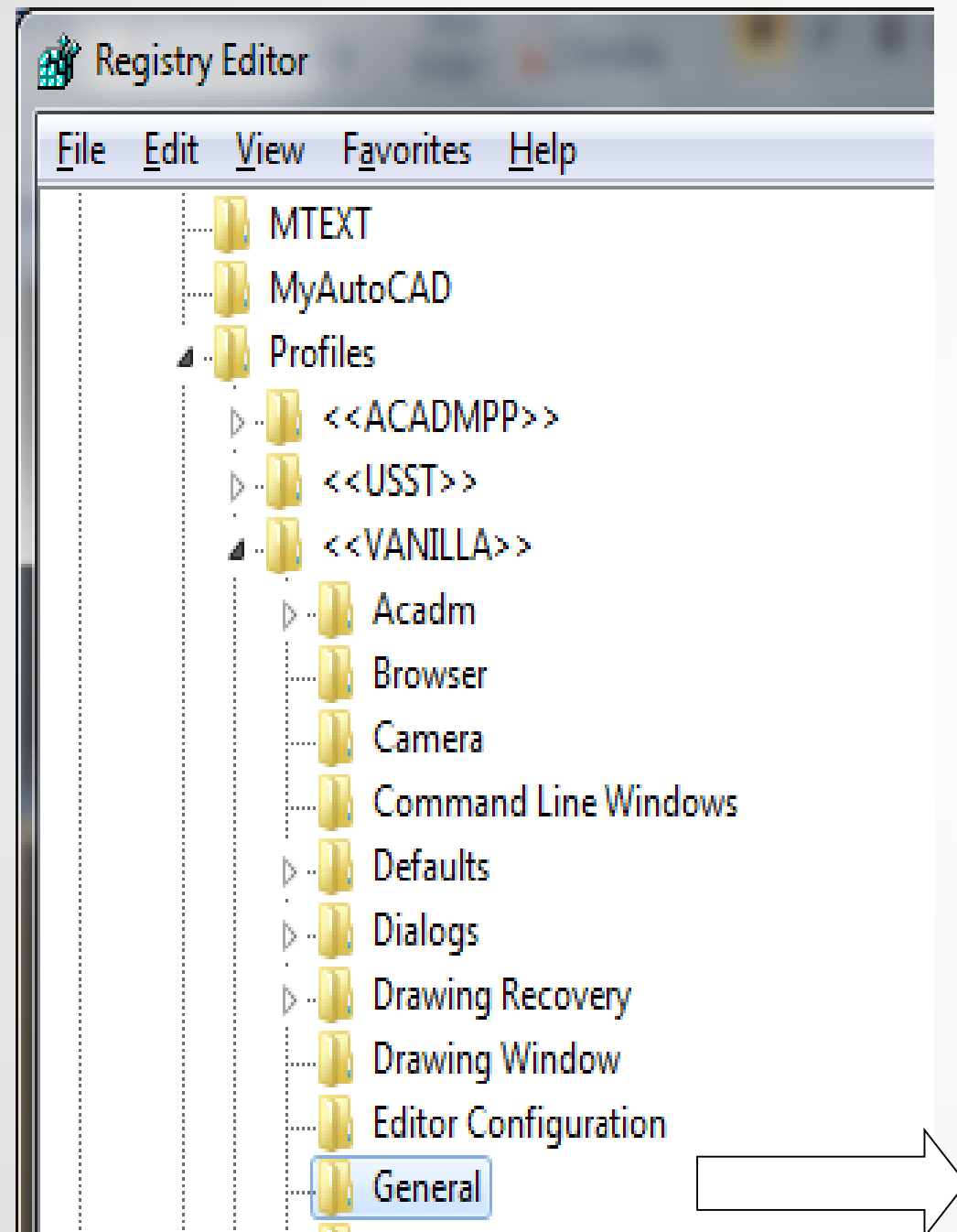
The tricky part ...

You've got to know the exact profile variable key to read/write from the CURRENT USER registry area.

This can require some detective work on your part.

Let's see how ...

Using REGEDIT ...



Find the current profile
Look under GENERAL
Find the key
Note exact spelling

Pickstyle	REG_DWORD	0x00000001 (1)
PlotLogPath	REG_EXPAND_SZ	%USERPROFILE%\appdata\local\autodesk\autocad r
PlotToFilePath	REG_EXPAND_SZ	%USERPROFILE%\documents
PrinterConfigDir	REG_EXPAND_SZ	%USERPROFILE%\appdata\roaming\autodesk\autoc
PrinterDescDir	REG_EXPAND_SZ	%USERPROFILE%\appdata\roaming\autodesk\autoc
PrinterStyleSheetDir	REG_EXPAND_SZ	%USERPROFILE%\appdata\roaming\autodesk\autoc
ProfileStorage	REG_EXPAND_SZ	%USERPROFILE%\appdata\roaming\autodesk\autoc
RegisteredToolsPath	REG_EXPAND_SZ	%RoamableRootFolder%\Support\RegisteredTools
Scrollbars	REG_DWORD	0x00000001 (1)

Now the code ...

Set the printer path like this:

```
(setenv "PrinterConfigDir" "Z:\\acad\\Plotters")
```

This is better:

```
(setenv "PrinterConfigDir"  
  (strcat "Z:\\acad\\Plotters;" (getenv "PrinterConfigDir"))  
)
```

Note ...

In this example:

```
(setenv "PrinterConfigDir"  
  (strcat "Z:\\acad\\Plotters;" (getenv "PrinterConfigDir"))  
)
```

I use STRCAT to glue two path strings together.

I had to use “;” in the first path to support multiple paths!

Better still ...

```
(if (not (wcmatch (getenv "PrinterConfigDir") *Z:\\acad\\Plotters*)) )  
  (setenv "PrinterConfigDir"  
    (strcat "Z:\\acad\\Plotters;" (getenv "PrinterConfigDir"))  
  )  
)
```

I used WCMATCH to test if the current PrinterConfigDir had the new path in it already. (Note the * qualifiers in the path)

This way I only reset the printer folder once.

Useful cheat codes ...

Support path: **ACAD**

DRV path: **ACADDRV**

Printer definitions: **PrinterConfigDir**

Palettes directory: **ToolPalettePath**

Templates location: **TemplatePath**

Plot Styles: **PrinterStyleSheetDir**

Current CUI: **MenuFile**

Enterprise CUI: **EnterpriseMenuFile**

Current Workspace: **WSCURRENT**

2014 Trusted Paths: **TRUSTEDPATHS**



Wrapping Up

Drawing some conclusions

AutoLISP is for CAD Managers!

I hope you've started to see what a great management tool AutoLISP is!

I will have final downloadable handouts, code samples, and slides in ZIP form for you by next Monday.

I'll send you an email to remind you.



AutoLISP® Strategies for CAD Managers



Robert Green

Robert Green Consulting

RGreen@GreenConsulting.com

