IT20496-L - AutoCAD Customization Boot Camp—Beyond the Basics

Lee Ambrosius

Principal Learning Experience Designer

Twitter: @LeeAmbrosius

Where Am I and Who Should Be Here

You are in session:

IT10485-L - AutoCAD Customization Boot Camp-Beyond the Basics

You should know:

AutoCAD 2017 (or AutoCAD 2009 and later)

You should want to:

- Automate tasks through scripting and programming
- Get AutoCAD to work for you

Who Am I?

My name is Lee Ambrosius

- Principal Learning Experience Designer at Autodesk
- Work on the Customization, Developer, and CAD Administration documentation
- Customizing and programming AutoCAD for about two decades
- Author of the AutoCAD Customization Platform book series published by Wiley

My job in a nutshell:

I document the present and past AutoCAD releases for the future

Who Are the Lab Assistants?

The lab assistants for this session are:

- Craig Black
- Alex Lepeska
- Scott Wilcox

Their roles are to:

- Help out when you get stuck
- Ensure no one gets left behind

Session Rules

A few rules for this session:

- Silent your mobile phone and any other device
- If you have to leave at anytime, please do so quietly
- Hold all questions to the end
- If you get stuck, raise your hand and one of the lab assistants will help you out

Thanks for your cooperation

Welcome to Specialist Training

What You Will Learn Today

By the end of this session, you will know how to:

- Create and run a script file
- Record and playback an action macro
- Write and deploy basic AutoLISP programs
- Create and set a user profile current

What is Going to be Covered

The handouts are broken into two separate parts/files:

- **Supplemental** – Content for the flight back
- **Exercises** – What we will be doing during this session

What You Need to Get Started

For this session, you will be using:

- AutoCAD 2017
- Action Recorder
- Notepad; part of the Windows operating system

Script Files

Script Files

What is a script file?

- An ASCII text file with the SCR extension
- Sequence of commands and system variables to be executed in a linear order
- Can include AutoLISP statements

Script Files

Why create or use scripts?

- Execute many commands rapidly without user input
- No special editor or programming skills required
- Low learning curve
- Work across multiple releases and verticals
- Transparent execution is supported

Script Files

Known Limitations

- User can’t be prompted for input
- Dialog boxes can’t be displayed
- Only one script can be executed at a time; except in AutoCAD 2016 and later

Script Files

Example of input entered at the Command prompt

```
Command: LIMITS
Reset Model space limits:
Specify lower left corner or [ON/OFF] <0.0000,0.0000>: 0,0
Specify upper right corner <12.0000,9.0000>: 1056,816
Command: ZOOM
Specify corner of window, enter a scale factor (nX or nXP),
   or [All/Center/Dynamic/Extents/Previous/Scale/Window
   /Object] <real time>: E
Command: GRIDDISPLAY
Enter new value for GRIDDISPLAY <3>: 2
```

Script Files

Examples of the same input as a script:

```
LIMITS
0,0
1056,816
ZOOM
E
GRIDDISPLAY
2
```

```
LIMITS 0,0 1056,816
ZOOM E
GRIDDISPLAY 2
```

Script Files

Formatting of a script file:

- Commands and options can be lower or uppercase
- Only values are case sensitive
- A space or new line is equivalent to pressing Enter
- Text to the right of a semi-colon isn't executed, semi-colon denotes a comment in a script
- Must always end with a blank line

Script Files

Formatting of a script file:

- A period in front of a command name ensures the execution of the natively defined command
- An underscore in front of a command name forces the use of a localized command name or option
- Commands are executed as if the FILEDIA and CMDDIA system variables were set to a value of 0

Script Files

Running a script file:

- SCRIPT command
- Drag and drop (Windows only)
- /b (Windows) or –b (Mac OS X) command line switch
- ScriptPro (Windows only)

Only one script file could be ran at a time until AutoCAD 2016; nested scripts can now be executed with the SCRIPTCALL command

Script Files

Commands related to script files:

- DELAY – Pauses the execution of a script for a specified duration in milliseconds
- RESUME – Resumes the execution of a script that was paused by pressing the Backspace key
- RSCRIPT – Repeats the previous executed script in the current AutoCAD session
- SCRIPT – Runs a SCR file
- SCRIPTCALL – Runs a nested script file; AutoCAD 2016 and later

Script Files

To create a script file, you need to:

1. At the Command prompt, walkthrough the commands and options to be executed by a script.
2. Create the script (SCR) file with Notepad.
3. Add the commands and options to the SCR file to be executed.
4. Save the SCR file.
5. Create or open a drawing file.
6. Run the SCR file and validate the results.

Script Files

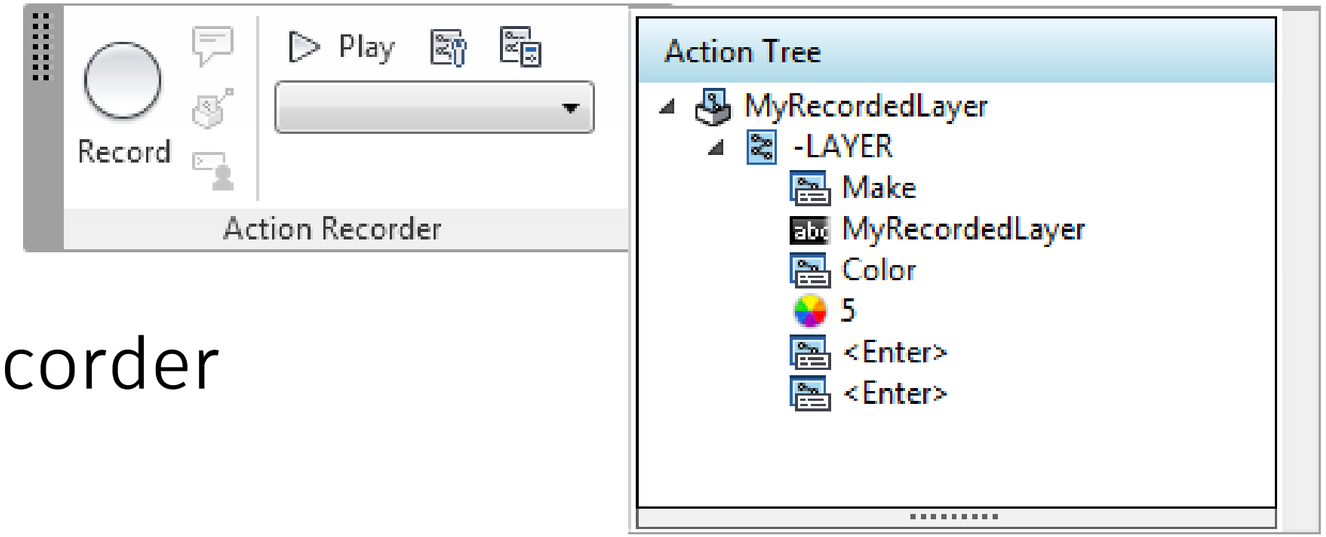
Do exercise “E1 - Creating and Running a Script”

In this exercise, you will

- Create a new SCR file that performs some basic drawing setup tasks
- Run a SCR file with the SCRIPT command

Action Macros

Action Macros



Recorded with the Action Recorder

Actions can be:

- Commands and user input
- Changes made with the Properties or Quick Properties palette
- Tools from a tool palette, and changes made with the Layer Properties Manager palette

Action Macros

Once recorded, action macros can be played back by:

- Entering its name at the Command prompt
- Selecting and playing it from the Action Recorder panel

Things you should know

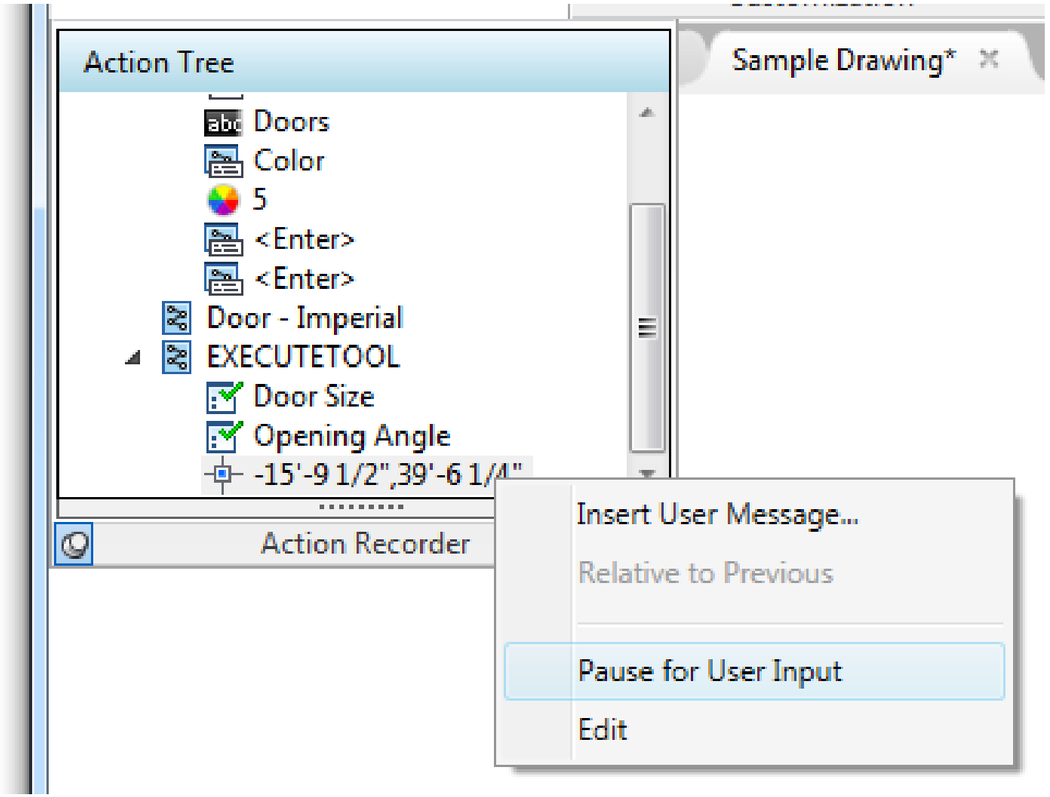
- Avoid recording commands that display a dialog box
- Can change system variable values while recording
- Action macros can be shared

Action Macros

User interactions can be added that affect the playback

User interactions can be:

- Displaying a user message
- Prompting for a value, selection set, or point
- Using the currently selected objects



Action Macros

To record an action macro, you need to:

1. Start recoding with the Action Recorder.
2. Perform the actions in the application and drawing you want to record.
3. Stop recording and save the action macro.
4. Edit the actions that were recorded.
5. Playback and test the action macro.

Action Macros

Do exercise “E2.A - Recording and Playing Back a Custom Action Macro”

In this exercise, you will

- Record the actions performed at the Command prompt that create a new layer and rectangular revision cloud
- Save and modify an ACTM file
- Playback a recorded action macro

AutoLISP

AutoLISP

AutoLISP as a programming language:

- Is specific to AutoCAD
- Has been around for a very long time; 30 years (January 1986), introduced in AutoCAD Version 2.18
- Based on the LISt Processing (LISP) programming language
- Does not require the use of a special editor
- Does not need to be compiled; interpreted language

AutoLISP

AutoLISP statements can be:

- Entered directly at the Command prompt in AutoCAD
- Stored and loaded from a LSP file
- Written using Notepad or the Visual LISP Editor
- Compiled as a FAS or VLX file to protect the source code

AutoLISP Expressions

AutoLISP expressions must:

- start with **(**
- end with **)**

Example:

```
(prompt "\nHello AU 2015!")
```

At the Command prompt, entering **(** indicates to AutoCAD that you want to work with AutoLISP

AutoLISP Syntax

Syntax of an AutoLISP expression:

(*function_name argumentX*)

- *function_name* indicates the name of the function to execute
- *argumentX* indicates the value(s) the function should do something with
- Some functions except no arguments while others accept one or more arguments

command Function

AutoLISP can be used to execute commands

A command is executed using the `command` function

Syntax:

```
(command command_name valueX)
```

- *`command_name`* indicates the name of the command to execute
- *`valueX`* indicates the option(s) and value(s) that the command expects

command Function

Examples:

Draws a single line segment from 0,0 to 5,5

```
(command "line" "0,0" "5,5" "")
```

Draws a circle at 0,0 with a radius of 5

```
(command "._circle" "0,0" 5)
```

Draws a circle at 0,0 with a diameter of 5

```
(command "._circle" "0,0" "_d" 5)
```

command Function

Two commonly used values with the `command` function are:

- `""` (pair of double quotations) indicates the press of the Enter key
- *`PAUSE`* indicates the command should wait for the user to provide a value, such as a point or object selection

These characters can be appended to the name of a command:

- **.** (period) – Uses the default/internal definition of a command
- **_** (underscore) – Indicates the use of a global command name

`setq` Function

A value can be assigned to a user-defined variable for future reference

A user-defined variable is defined with the `setq` function

Syntax:

`(setq` *`variable_name value`*`)`

- *`variable_name`* indicates the name of the variable to define or update
- *`value`* indicates the value to be assigned to the variable

`setq` Function

Examples:

Assigns the text AU 2016 to the variable named `strEvent`

```
(setq strEvent "AU 2016")
```

Assigns the numeric value of 1.25 to the variable named `dRadius`

```
(setq dRadius 1.25)
```

`setq` Function

At the Command prompt, entering **!** indicates to AutoCAD you want to know the value of a user-defined variable:

Example:

```
Command: (setq dRadius 1.25)
Command: !dRadius
1.25
```

AutoLISP Data Types

AutoLISP functions accept several types of data and the most comment are:

- **Integer** - Any number without a decimal point

  Examples: `12` , `0`
- **Real** - Any number with a decimal point

  Examples: `12.125`, `0.0`
- **String** - Any alphanumeric characters enclosed in double quotes

  Examples: `"12.125"`, `"Welcome to AU 2016"`

AutoLISP Data Types

Other commonly used data types:

- **List** - Any expression in parentheses

  Examples: `(0.0 5.0 0.0)`

  `(command "._LINE" "0,0" "5,5" "")`

- **Symbol** - Internal or user-defined AutoLISP variable

  Examples: `PAUSE`, `dRadius`

Entering AutoLISP Expressions

Do exercise “E3 - Entering AutoLISP Expressions at the Command Prompt”

In this exercise, you will

- Enter AutoLISP expressions at the Command prompt
- Execute commands
- Store values in user-defined variables

Defining Custom Functions

AutoLISP can be used to create reusable functions

A custom function:

- Defined with the defun function
- Executed similar to standard AutoCAD commands
- Used to build standardized components for complex programs

`defun` Function

Syntax of the `defun` function

Syntax:

```
(defun c:function_name ( / )
    expressionX
)
```

- *`function_name` represents the name of the function*
- *`c:` indicates the function name can be typed at the Command prompt*
- *`expressionX` represents the expressions to execute*

defun Function

Examples:

```
(defun c:HelloWorld ( / )
   (alert "Hello World!")
)

(defun c:ZP ( / )
   (command "._zoom" "_p")
)
```

Defining Custom Functions

Do exercise “E4 - Creating Simple Custom AutoLISP Functions”

In this exercise, you will

- Define two custom functions
- Execute the custom functions at the AutoCAD Command prompt

Storing AutoLISP Expressions in an External File

Expressions can be stored in an AutoLISP file

- ASCII text file with a .lsp extension
- LSP files can be created/modified with Notepad or the Visual LISP Editor
- Comments can be added to a LSP file
- A LSP file must be loaded into each drawing to be used

Documenting AutoLISP Expressions

Comments in an AutoLISP file

- Indicated by an ; (semi-colon)
- Expressions to the right of a ; are not executed
- Comments are used to provide information about an LSP file for expressions in an LSP file

Examples:

```
; Created on: 8/19/16 by Lee Ambrosius

(setq dRad 1.25)  ; Default radius value
```

Manually Loading a LSP File

These methods can be used to manually load an LSP file:

- APPLOAD command
- AutoLISP `load` function
- Drag and drop an LSP file onto the drawing area (Windows only)

Automatically Loading a LSP File

These methods can be used to automatically load an LSP file:

- Startup Suite in the Load/Unload Applications dialog box (APPLOAD command)
- LISP Files node in the CUI Editor (Windows only)
- Menu AutoLISP (MNL) files
- acad.lsp and acaddoc.lsp files
- Plug-in Bundle

Working with LSP Files

Do exercise “E5 - Creating and Loading a LSP File”

In this exercise, you will

- Create a new LSP file
- Add expressions and comments to an LSP file
- Load an LSP file

Deploying a LSP File with a Plug-in Bundle

Plug-in bundles:

- Provide a consistent way to deploy and load LSP files
- A file and folder structure that contains an XML file named *PackageContents.xml*

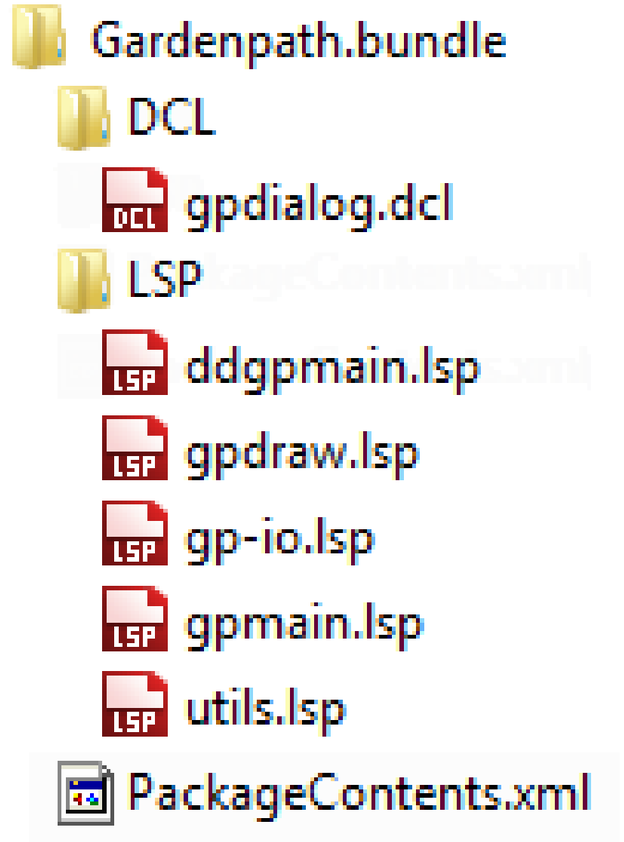
PackageContents.xml is

- Placed in the root folder of a bundle
- Describes the files in the bundle to AutoCAD and defines how they should be loaded

Deploying a LSP File with a Plug-in Bundle

Example structure of a bundle named GardenPath:

```
Gardenpath.bundle
  |-> DCL
        |-> gpdialog.dcl
  |-> LSP
        |-> ddgpmain.lsp
        |-> gpdraw.lsp
        |-> gp-io.lsp
        |-> gpmain.lsp
        |-> utils.lsp
  |-> PackageContents.xml
```



Deploying a LSP File with a Plug-in Bundle

Basic example of a *PackageContents.xml* file:

```
<?xml version="1.0" encoding="utf-8"?>
<ApplicationPackage
    SchemaVersion="1.0"
    AppVersion="1.0"
    Name="AU2016 IT20496-L"
    Description="AU2016 Example for session IT20496-L."
    Author="HyperPics, LLC"
    ProductCode="{45F619FE-E286-4C4E-8134-B50E8DFC23E3}"
>
```

Deploying a LSP File with a Plug-in Bundle

```
    <CompanyDetails
        Name="HyperPics, LLC"
        Url="http://www.hyperpics.com"
    />
<Components Description="Windows and Mac OS operating systems">
    <RuntimeRequirements
        OS="Win32|Win64|Mac"
        SeriesMin="R19.0"
        Platform="AutoCAD*"
    />
```

Deploying a LSP File with a Plug-in Bundle

```
    <ComponentEntry Description="Your custom file"
        AppName="AU2016Examples"
        Version="1.0"
        ModuleName="./au2016.lsp">
     </ComponentEntry>
  </Components>
</ApplicationPackage>
```

Deploying a LSP File with a Plug-in Bundle

Access the AutoCAD Online Help system for more information on the *PackageContents.xml* file.

Note: The ProductCode value (GUID) must be unique for each bundle. - *http://www.guidgenerator.com/*

A bundle is deployed by copying all the files and folders of a bundle to one of these folders:

- All Users Profile folder
- User Profile folder

Deploying a LSP File with a Plug-in Bundle

All Users Profile folder

- Windows 7 and later:
  %ALLUSERSPROFILE%\Autodesk\ApplicationPlugins
- Mac OS X:
  /Applications/Autodesk/ApplicationAddins

User Profile folder

- Windows 7 and later:
  %APPDATA%\Autodesk\ApplicationPlugins
- Mac OS X:
  ~/Autodesk/ApplicationAddins

Deploying a LSP File with a Plug-in Bundle

Do exercise “E6 - Creating a Basic Plug-in Bundle to Load an AutoLISP Program”

In this exercise, you will

- Create the folder structure for a plug-in bundle
- Update a *PackageContents.xml* file for a plug-in bundle
- Deploy a plug-in bundle

User Profiles

User Profiles

Profiles are used to control application and user preferences:

- Search paths used to locate support files,
- trusted locations for custom program files,
- colors and fonts used by grips, application, and Command window,
- plot/publish, open and save file options,
- and many other settings.

Profiles are:

- Created using the Options dialog box
- Set current using the Profiles tab of the Options dialog box or `/p` command line switch

User Profiles

To create a user profile, you need to:

1. Display the Options dialog box.
2. Set the Profiles tab current.
3. Add a new profile and set it current.
4. Adjust the preferences and settings in the Options dialog box.

User Profiles

Do exercise “E7 - Creating and Modifying a New Profile”

In this exercise, you will

- Create a new user profile
- Change the settings associated with a user profile
- Set a user profile current

Final Thoughts and Questions

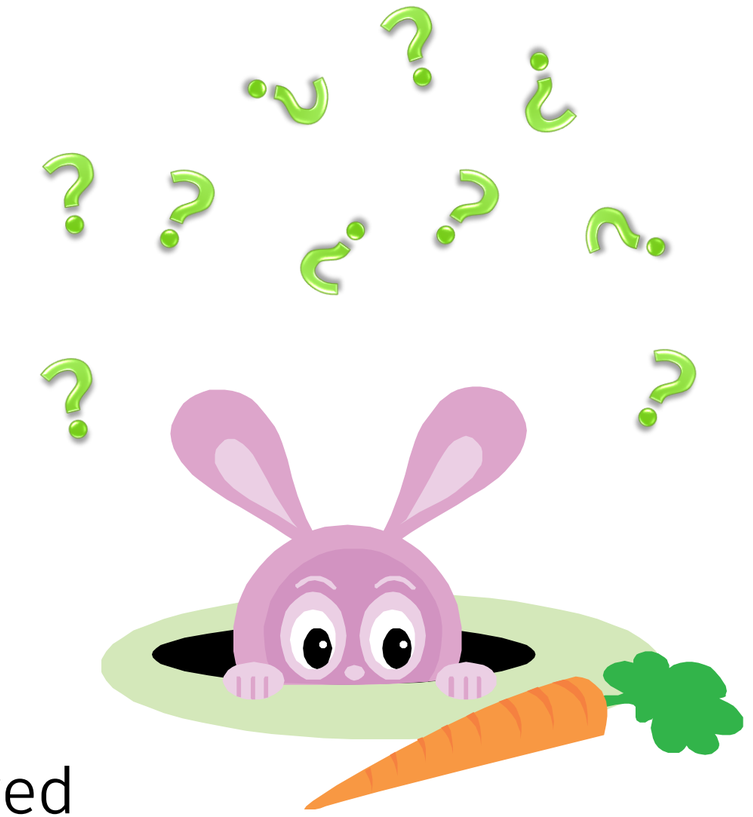
Final Thoughts and Questions

Scripting and programming can:

- Enhance productivity
- Improve or introduce new workflows

Programming has many similarities to the rabbit hole in Lewis Caroll's *Alice's Adventures in Wonderland*. Both:

- Are virtually endless
- Hold many mysteries waiting to be discovered



Closing Remarks

Thanks for choosing this session.

Don’t forget to complete this session’s online evaluation.

If you have any further questions, contact me via:

email: lee.ambrosius@autodesk.com
twitter: @leeAmbrosius