## Contents

**Chapter 1  Basic Customization** .................................................. 1  
Overview of Customization ....................................................... 1  
Organize Program and Support Files ........................................... 2  
  Overview of File Organization .................................................. 2  
  Multiple Drawing Folders ......................................................... 5  
  Locate Customized Files ........................................................ 6  
    Locate Plot Style Files ....................................................... 7  
    Locate Plotter Files .......................................................... 8  
    Locate Support Files .......................................................... 8  
    Locate Drawing Template Files ............................................. 11  
Create Command Aliases .......................................................... 12  
Create Command Aliases .......................................................... 13  

**Chapter 2  Custom Linetypes** .................................................. 15  
Overview of Linetype Definitions ................................................ 15  
Simple Custom Linetypes .......................................................... 16  
  Simple Custom Linetypes ....................................................... 18  
Text in Custom Linetypes .......................................................... 19  
  Text in Custom Linetypes ....................................................... 21  

**Chapter 3  Custom Hatch Patterns** ............................................. 23  
Overview of Hatch Pattern Definitions ...................................... 23  
  Overview of Hatch Pattern Definitions .................................... 25
# Hatch Patterns

- Hatch Patterns with Dashed Lines: 25
- Hatch Patterns with Multiple Lines: 27

## Chapter 4 User Interface Customization

- Understand User Interface Customization: 31
- Overview of the Customization: 31
- Customization Glossary: 32
- Customize Commands: 32
  - Overview of Commands: 32
  - Create, Edit, and Reuse Commands: 33
    - Create, Edit, and Reuse Commands: 34
  - Create Macros: 36
    - Overview of Macros: 36
    - Use Special Control Characters in Macros: 38
    - Pause for User Input in Macros: 40
    - Provide International Support in Macros: 42
    - Repeat Commands in Macros: 42
    - Use Single Object Selection Mode in Macros: 43
    - Use Conditional Expressions in Macros: 43
  - Control the Display of Command Items: 44
  - Create Tooltips for Commands: 45
  - Create and Manage Images for Commands: 46
    - Assign Images to a Command: 46
  - Customize User Interface Elements: 46
    - Menus: 47
      - Overview of Menus: 47
      - Create and Manage Pull-down Menus: 47
    - Tool Sets: 51
      - Overview of Tool Sets and Tool Groups: 51
      - Create and Manage Tool Sets: 51

## Chapter 5 DIESEL

- DIESEL Expressions in Macros: 57
- Catalog of DIESEL Functions: 59
  - + (addition): 59
  - - (subtraction): 59
  - * (multiplication): 60
  - / (division): 60
  - = (equal to): 60
  - < (less than): 60
  - > (greater than): 60
  - != (not equal to): 61
  - <= (less than or equal to): 61
DIESEL Error Messages .................................................... 67

Chapter 6  Command Scripts ........................................... 69
Create Command Scripts ................................................. 69
  Overview of Command Scripts ....................................... 69
  Overview of Command Scripts ....................................... 70
Run Scripts at Startup ..................................................... 71
  Run Scripts at Startup ................................................. 72

Index ................................................................. 73
Basic Customization

AutoCAD LT® is a general-purpose drafting system designed with an open architecture so you can customize and extend its many features. As a result, you can expand and shape AutoCAD LT according to your needs.

Overview of Customization

AutoCAD LT can be customized in simple ways. For example, you can change the directory structure or create a custom title block to use on a layout. If you want to change the interface further, you can edit the Tool Sets palette or menu bar, and use DIESEL statements to create custom commands.

The list that follows is arranged from least to most complex:

- **Organize files.** You can organize program, support, and drawing files. For example, you can make a separate folder for each project that includes only the support files that project needs.

- **Create custom drawing templates (DWTs).** You can create custom drawing templates to use when creating a new drawing. Drawing templates are used to store the layers, blocks, and styles that you might use across all your drawings.

- **Define command aliases.** You can define simple abbreviations, or aliases, for frequently used commands from within AutoCAD LT by adding the command to the PGP file acadlt.pgp. For example, you might want to start the BLOCK command by entering `b`.

- **Create custom linetypes and hatch patterns.** You can create linetypes and hatch patterns that conform to your company standards and working methods.

- **Customize the user interface.** You can control many aspects of the user interface, including the functionality and appearance of the Tool Sets palette.
and menu bar. You use the CUI command to create and edit commands, and assign them to a menu on the menu bar or a tab on the Tool Sets palette.

- **Customize the status line.** You can toggle the display of the controls on the status bar. Right-click an empty area of the status bar and choose which controls to turn on or off.

- **Automate repetitive tasks by writing scripts.** A script is an ASCII text file containing commands that are processed like a batch file when you run the script. For example, if a set of drawings needs to be plotted a certain way, you can write a script that opens each drawing, hides and displays various layers, and issues PLOT commands.

See also:
- Organize Program and Support Files (page 2)
- Create Command Aliases
- Custom Linetypes (page 15)
- Custom Hatch Patterns (page 23)
- DIESEL (page 57)
- Command Scripts (page 69)

Organize Program and Support Files

You can change the default directory structure for the program and support files to suit your needs.

Overview of File Organization

AutoCAD LT® uses support files for purposes such as storing customization definitions and describing text fonts.

The default directory structure for the AutoCAD LT program and support files is designed to efficiently organize those files into logical groups. If this organization does not suit your needs, you can change it. However, some applications look for certain files in specific locations, and you should verify that your modifications do not conflict with the requirements of those applications. Without the full path, including drive and directory, AutoCAD LT can locate only those files that are found in the library search path.
The location of local customizable files is stored in the LOCALROOTPREFIX system variable. The location of roamable customizable files is stored in the ROAMABLEROOTPREFIX system variable. If a network supports roaming, customizable files in the user's roaming profile are available on the machine the user is logged onto.

**Library Search Path**

The library search path specifies where the program searches for files when you do not specify a full path name, as follows:

- Current directory.
- Directory that contains the current drawing file.
- Directories listed in the search path specified on the Applications tab in OPTIONS. (See Specify Search Paths and File Locations in the User's Guide.)
- Directory that contains the AutoCAD LT program files.

Depending on the current environment, two or more directories may be the same.

If a file is not in this search path, you must specify both its path name and file name before AutoCAD LT can find it. For example, if you want to insert the `part5.dwg` drawing into your current drawing and it is not in the library search path, you must specify its full path name, as shown here:

```
Command: insert
Enter block name or [?] : /files2/olddwgs/part5
```

If the drawing exists in that location, AutoCAD LT prompts you to finish the INSERT command in the usual manner.

**Directory Structure**

AutoCAD LT uses tree-structured directories and subdirectories. It is recommended that you keep supplemental files (such as third-party applications and customization files) separate from the AutoCAD LT program and support files. This makes it easier to track possible conflicts and to upgrade each application without affecting the others.

You can create a new directory on the same level as the main AutoCAD LT directory and store any customization files or third-party applications in subdirectories on the next level. If you want to store drawings by job, you can create a third directory on the same level. Subdirectories for each job could contain drawing files and a subdirectory for related support files.
Command Search Procedure

When you enter a command, AutoCAD LT goes through a series of steps to evaluate the validity of the command name. A command can be a built-in command or system variable or an alias defined in the acadlt.pgp file. Commands can also be defined by a device driver command. You can enter a command on the command prompt or choose a command from the appropriate menu. Commands can also be entered from a script file.

The following list describes the search order AutoCAD LT uses to validate a command name.

1. If the input is a null response (Spacebar or Enter), AutoCAD LT uses the name of the last command issued.

2. AutoCAD LT checks the command name against the list of built-in commands. If the command is in the list and is not preceded by a period (.), AutoCAD LT then checks the command against a list of undefined commands. If the command is undefined, the search continues. Otherwise, the command is run, unless another reason prevents it from doing so. Running it transparently or in Perspective mode might be impossible.

3. AutoCAD LT checks the command name against the names of commands defined by a device driver, and then by those defined by the display driver.

4. AutoCAD LT checks the command name against the list of system variables. If the command name is in the list, AutoCAD LT executes the SETVAR command, using the input as the variable name.

5. If the command name corresponds to a command alias defined in the program parameters file, AutoCAD LT uses the expanded command name and continues the search, starting a new search against the list of built-in commands.

6. If all the preceding steps fail, the search terminates with a warning message about illegal command names.

See also:

Specify Search Paths and File Locations in the User’s Guide
Multiple Drawing Folders

Keeping your drawing and other associated files in separate directories makes it easier to perform basic file maintenance.

Keeping your drawing files and other associated files in separate directories makes it easier to perform basic file maintenance. The scenario described in this topic is based on the sample directory structure described in Overview of File Organization (page 2), but you can expand or alter it to meet your needs.

You can set up the /AcltJobs directory to contain your drawing subdirectories. The drawing subdirectories can contain other subdirectories that hold related support files for a particular drawing type or job. The /AcltJobs/Job1/Support directory can contain blocks specific to the drawing files in /AcltJobs/Job1. Specifying support (with no path prefix) in the Support path adds the support directory within the current directory to the Support path.

To make sure that the required drawing directory is the current directory when you start AutoCAD LT, and that all files and subdirectories in that directory are easily accessible, you can create a program icon or a Start menu item that specifies the correct working directory for each job. This functionality works only if you set the AutoCAD LT system variable REMEMBERFOLDERS to 0.

You can use a batch program to create new job directories automatically. The following batch program verifies that a specified directory exists, sets that directory to be current, and then runs AutoCAD LT.

```bash
#!/bin/sh
prj="$1"
#Switch to the project folder and start AutoCAD LT
function startACADLT() {
    cd /AcltJobs/Jobs/$prj
    echo "Starting AutoCAD LT"
    "/Applications/Autodesk/AutoCAD LT 2013/AutoCAD LT
    2013.app/Contents/MacOS/AutoCAD LT 2013"
}
#Clear Terminal and check for the existence of the folder
clear
cd .
if [ -d /AcltJobs/Jobs/$prj ] then
    startACADLT
fi
#Prompt to create folder
echo .
```
echo Creating /AcltJobs/Jobs/$prj
echo 'Press Y to continue (or A to abort)'
cont="True"
answer=""while [ "$cont" = "True" ]
do
  read -n1 -t10 answer
echo
  if [ "$answer" = "y" ] || [ "$answer" = "Y" ] || [ "$answer" = "a" ] || [ "$answer" = "A" ]
    then
      cont="False"
    fi
done
#Check to see if the user requested to abort or continue
if [ "$answer" = "a" ] || [ "$answer" = "A" ]
    then
      exit 1
else
  mkdir -p /AcltJobs/Jobs/$prj
fi
#Switch to the project folder and start AutoCAD LT
startACADLT

Using an ASCII text editor (such as TextEdit), save the batch program to a file named acadlt.sh. Be sure to change the drive and directory names to match those on your system.

Place this file in your home directory or a shared location that is on your system. You can run this shell script program using the Terminal window in /Applications/Utilities on the drive the operating system is installed. If you saved the file as acadlt.sh, use the following syntax:

```
./acadlt.sh jobname
```

where jobname is the name of the job directory to make current.

**Locate Customized Files**

AutoCAD LT supports a wide range of files that can be customized. The program stores files that can be customized by user profile, as well as allow
you to add your own customized file locations. The following locations are defined by the program:

- **Local profiles.** Local profiles are used to log on a computer and they store settings and files that are not available when roaming. Some files, such as materials and drawing templates are stored under your local profile because of their size, they do not follow you from computer to computer.

- **Roaming profiles.** Roaming profiles allow you to log on to any computer within a network and retain your user settings. Some files, such as your personal settings and documents, follow you from computer to computer.

If roaming profiles are allowed on your network, your “roamable” files are located in the `<user>/Library/Application Support/Autodesk/Roaming/<Product Version>` folder, and your “nonroamable” files are located in the `<user>/Application Support/Autodesk/Local/<Product Version>` folder.

### Locate Plot Style Files

The location AutoCAD LT uses for plot style files defines on how the program was installed and configured. You can use the OPTIONS command to locate which folder is being used to store your plot styles.

**To locate your plot style files**

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3. From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**

A drawing must be open to see the Preferences menu item.

4. In the Application Preferences dialog box, Application tab, click the arrow to the left of Printer Support File Path.
5. Click the arrow to the left of the Plot Style Table Search Path file.
6. Under Plot Style Table Search Path, click the path name to view the location of your plot style files.

**NOTE** You can also locate your plot style files by entering `stylesmanager` on the AutoCAD LT command line.
Locate Plotter Files

The location AutoCAD LT looks in for custom printer settings is stored in the Application Preferences dialog box. You can use the OPTIONS command to locate which folder is being used to store your plot configuration files.

To locate your plotter files

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3. From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.
   
   **NOTE**
   
   A drawing must be open to see the Preferences menu item.

4. In the Application Preferences dialog box, Application tab, click the arrow to the left of Printer Support File Path.
5. Click the arrow to the left of Printer Configuration Search Path.
6. Under Printer Configuration Search Path, click the path name to view the location of your plotter files.

   **NOTE** You can also locate your plotter files by entering `plottermanager` on the AutoCAD LT command line.

Locate Support Files

Support files include the following:

- Custom icon files
- Help and miscellaneous files
- Font mapping file (`acadlt.fmp`)
- Alternate font file (`simplex.shx`)
- Support path files (`acadlt.lin`, `acadlt.pat`, `acadlt.pgp`, `acadlt.psf`, `acadlt.unt`, `acadltiso.lin`, `acadltiso.pat`, `gdt.shx`, `inches.pss`, and `mm.pss`)
Locate Support Files

To find the default location of the customization files

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3. From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**
A drawing must be open to see the Preferences menu item.

4. In the Application Preferences dialog box, Application tab, click the arrow to the left of Customization Files.
5. Click the arrow to the left of Main Customization File.
6. Under Main Customization File, click the path name to view the location of your main customization file.

To find the default location of the custom icon files

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3. From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**
A drawing must be open to see the Preferences menu item.

4. In the Application Preferences dialog box, Application tab, click the arrow to the left of Customization Files.
5. Under Custom Icon Location, click the path name to view the location for the custom button image files used with your customization files.

To find the default location of the Help and miscellaneous files

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3. From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**
A drawing must be open to see the Preferences menu item.

4. In the Application Preferences dialog box, Application tab, click the arrow to the left of Help and Miscellaneous File Names.

5. Click the arrow to the left of the file you want to locate, and then click the path name to view the location of the files.

**To find the default location of the font mapping file**

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3. From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**
A drawing must be open to see the Preferences menu item.

4. In the Application Preferences dialog box, Application tab, click the arrow to the left of Text Editor, Dictionary, and Font File Names.

5. Click the arrow to the left of Font Mapping File.

6. Under Font Mapping File, click the path name to view the location of your font mapping file.

**To find the default location of the alternate font file**

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3. From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**
A drawing must be open to see the Preferences menu item.

4. In the Application Preferences dialog box, Application tab, click the arrow to the left of Text Editor, Dictionary, and Font File Names.

5. Click the arrow to the left of Alternate Font File.
6 Under Alternate Font File, click the path name to view the location of your alternate font file.

**To find the default location of the support path files**

1 From the Finder menu bar, click Go ➤ Applications.
2 In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3 From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**
A drawing must be open to see the Preferences menu item.

4 In the Application Preferences dialog box, Application tab, click the arrow to the left of Support File Search Path.

5 Under Support File Search Path, click a path name to view the location of your support files.

---

**Locate Drawing Template Files**

The location AutoCAD LT looks in for drawing templates is stored in the Application Preferences dialog box. You can use the OPTIONS command to locate which folder is being used to store your drawing templates.

---

**Locate Drawing Template Files**

**To locate your drawing template files**

1 From the Finder menu bar, click Go ➤ Applications.
2 In the Applications window, double-click Autodesk ➤ AutoCAD LT 2013 ➤ AutoCAD LT.
3 From the Mac OS menu bar, click AutoCAD LT ➤ Preferences.

**NOTE**
A drawing must be open to see the Preferences menu item.

4 In the Application Preferences dialog box, Application tab, click the arrow to the left of Template Settings.
5 Under Template Settings, click the arrow sign (+) to the left of Drawing Template File Location.

6 Under Drawing Template File Location, click the path name to view the location of your drawing template files.

Create Command Aliases

A command alias is an abbreviation that you enter at the command prompt instead of entering the entire command name.

For example, you can enter \texttt{c} instead of \texttt{circle} to start the \texttt{CIRCLE} command. An alias is not the same as a keyboard shortcut, which is a combination of keystrokes, such as \texttt{Ctrl-S} for \texttt{SAVE}.

The \texttt{acadlt.pgp} file defines command aliases. You can change existing aliases or add new ones by editing \texttt{acadlt.pgp} in an ASCII text editor. In addition to command aliases in \texttt{acadlt.pgp}, you will also find comment lines which are preceded by a semicolon (;). Comment lines allow you to add textual information to \texttt{acadlt.pgp}, such as when or who revised the file last.

\textbf{NOTE} Before you edit \texttt{acadlt.pgp}, create a backup so that you can restore it later, if necessary.

To define a command alias, add a line to the \texttt{acadlt.pgp} file using the following syntax:

\begin{verbatim}
abbreviation,*command
\end{verbatim}

\texttt{abbreviation} is the command alias that you enter at the command prompt and \texttt{command} is the command being abbreviated. You must enter an asterisk (\texttt{*}) before the command name to identify the line as a command alias definition.

If you can enter a command transparently, you can also enter its alias transparently. When you enter the command alias, the full command name is displayed at the command prompt and the command is executed.

You can create command aliases that include the special hyphen (\texttt{-}) prefix, such as those listed here, that accesses the version of a command that displays command prompts instead of a dialog box.

\begin{verbatim}
BH, *-BHATCH
BD, *-BOUNDARY
\end{verbatim}
NOTE You cannot use command aliases in command scripts. Using command aliases in custom commands is not recommended.

Restarting AutoCAD LT automatically reloads the file.

Create Command Aliases

To open the program parameters file (acadlt.pgp)

1. From the Finder menu bar, click Go ➤ Home.
2. In the Finder window, double-click Library. Continue to navigate to Application Support/Autodesk/Roaming/AutoCAD LT 2013/<version>/<language>/Support and double-click acadlt.pgp. If prompted for an application to use, select TextEdit.
Custom Linetypes

AutoCAD LT® provides a library of standard linetypes in the acadlt.lin and acadltiso.lin files. You can use the linetypes as they are, modify them, or create your own custom linetypes.

Overview of Linetype Definitions

Linetypes are defined in one or more linetype definition files that have a .lin file extension.

The linetype name and definition determine the particular dash-dot sequence, the relative lengths of dashes and blank spaces, and the characteristics of any included text or shapes. You can use any of the standard linetypes that AutoCAD LT provides, or you can create your own linetypes.

Examples of linetypes

A LIN file can contain definitions of many simple and complex linetypes. You can add new linetypes to an existing LIN file, or you can create your own LIN file. To create or modify linetype definitions, edit the LIN file using a text editor or word processor or use LINETYPE at the command prompt.

When you create a linetype, you must load the linetype before you can use it.

The LIN files included in AutoCAD LT are acadlt.lin and acadltiso.lin. You can display or print these text files to better understand how to construct linetypes.
Simple Custom Linetypes

Each linetype is defined on two lines in a linetype definition file. The first line contains the linetype name and an optional description. The second line is the code that defines the actual linetype pattern.

The second line must begin with the letter A (alignment), followed by a list of pattern descriptors that define pen-up lengths (spaces), pen-down lengths (dashes), and dots. You can include comments in an LIN file by beginning the line with a semicolon (;).

Linetype Definition Format

The format of the linetype definition is

```
*linetype_name,description
A,descriptor1,descriptor2, ...
```

For example, a linetype called DASHDOT is defined as

```
*DASHDOT,Dash dot __ · __ · __ · __ · __ · __ · __
A,.5,-.25,0,.25
```

This indicates a repeating pattern starting with a dash 0.5 drawing units long, a space 0.25 drawing units long, a dot, and another space 0.25 drawing units long. This pattern continues for the length of the line, ending with a dash 0.5 drawing units long. The linetype would be displayed as shown below.

```
__ · __ · __ · __ · __ · __ · __
```

LIN files must be saved in ASCII format and use an .lin file extension.

Additional information about each field in a linetype definition follows.

Linetype Name

The linetype name field begins with an asterisk (*) and should provide a unique, descriptive name for the linetype.

Description

The description of the linetype should help you visualize the linetype when you edit the LIN file. The description is also displayed in the Linetype Manager and in the Load or Reload Linetypes dialog box.

The description is optional and can include

- A simple representation of the linetype pattern using ASCII text
An expanded description of the linetype

A comment such as "Use this linetype for hidden lines"

If you omit the description, do not insert a comma after the linetype name. A description cannot exceed 47 characters.

Alignment Field (A)

The alignment field specifies the action for pattern alignment at the ends of individual lines, circles, and arcs. Currently, AutoCAD LT supports only A-type alignment, which guarantees that the endpoints of lines and arcs start and stop with a dash.

For example, suppose you create a linetype called CENTRAL that displays the repeating dash-dot sequence commonly used as a centerline. AutoCAD LT adjusts the dash-dot sequence on an individual line so that dashes and line endpoints coincide. The pattern fits the line so that at least half of the first dash begins and ends the line. If necessary, the first and last dashes are lengthened. If a line is too short to hold even one dash-dot sequence, AutoCAD LT draws a continuous line between the endpoints. For arcs also, the pattern is adjusted so that dashes are drawn at the endpoints. Circles do not have endpoints, but AutoCAD LT adjusts the dash-dot sequence to provide a reasonable display.

You must specify A-type alignment by entering a in the alignment field.

Pattern Descriptors

Each pattern descriptor field specifies the length of segments making up the linetype, separated by commas (no spaces are allowed):

- A positive decimal number denotes a pen-down (dash) segment of that length.
- A negative decimal number denotes a pen-up (space) segment of that length.
- A dash length of 0 draws a dot.

You can enter up to 12 dash-length specifications per linetype, provided they fit on one 80-character line in the LIN file. You need to include only one complete repetition of the linetype pattern defined by pattern descriptors. When the linetype is drawn, AutoCAD LT uses the first pattern descriptor for the starting and ending dashes. Between the starting and ending dashes, the pattern dash specifications are drawn sequentially, beginning with the second
dash specification and restarting the pattern with the first dash specification when required.

A-type alignment requires that the first dash length be 0 or greater (a pen-down segment). The second dash length should be less than 0 if you need a pen-up segment and more than 0 if you are creating a continuous linetype. You must have at least two dash specifications for A-type alignment.

### Simple Custom Linetypes

**To create a simple linetype from the Command prompt**

1. At the command prompt, enter `-linetype`
2. Enter `c` (Create).
3. Enter a name for the linetype and press Enter.
   The linetype name can include up to 255 characters. Linetype names can contain letters, digits, and the special characters dollar sign ($), hyphen (-), and underscore (_). Linetype names cannot include blank spaces.
4. In the Create or Append Linetype File dialog box, select a location from the Where drop-down list and specify a LIN linetype library file. Click Save.
   If you select an existing file, the new linetype name is added to the linetype names in the file.
5. Enter text that describes the new linetype (optional).
6. At the Enter Pattern prompt, specify the pattern of the line. Follow these guidelines:
   - All linetypes must begin with a dash.
   - Enter zeros for dots.
   - Enter negative real numbers for spaces. The value defines the length of the space in drawing units.
   - Enter positive real numbers for dashes. The value defines the length of the dash in drawing units.
   - Separate each dot, dash, or space value from the next with a comma.
   - Use a space between a dot and a dash.
7. Press Enter to end the command.
NOTE
When you create a linetype, it is not loaded into your drawing automatically. Use the Load option of LINETYPE.

To add a simple linetype to a LIN file

1. Open the acadlt.lin or acadltiso.lin file in a text editor that saves in ASCII format.
2. Create a header line that includes an asterisk and a linetype pattern name. The name of the linetype pattern is limited to 31 characters.
3. (Optional) To include a description in the header line, follow the linetype pattern name with a comma and description text.
4. Create a descriptor line that includes:
   ■ All linetypes must begin with a dash.
   ■ Enter zeros for dots.
   ■ Enter negative real numbers for spaces. The value defines the length of the space in drawing units.
   ■ Enter positive real numbers for dashes. The value defines the length of the dash in drawing units.
   ■ Separate each dot, dash, or space value from the next with a comma.
   ■ Use a space between a dot and a dash.

Text in Custom Linetypes

Characters from text fonts can be included in linetypes.

Characters from text fonts can be included in linetypes. Linetypes with embedded characters can denote utilities, boundaries, contours, and so on. As with simple linetypes, lines are dynamically drawn as you specify the vertices. Characters embedded in lines are always displayed completely; they are never trimmed.

Embedded text characters are associated with a text style in the drawing. Any text styles associated with a linetype must exist in the drawing before you load the linetype.

The format for linetypes that include embedded characters is similar to that for simple linetypes in that it is a list of pattern descriptors separated by commas.
Character Descriptor Format

The format for adding text characters in a linetype description is as follows:

```
[ "text", textstylename, scale, rotation, xoffset, yoffset ]
```

This format is added as a descriptor to a simple linetype. For example, a linetype called HOT_WATER_SUPPLY is defined as

```
*HOT_WATER_SUPPLY,---- HW ---- HW ---- HW ---- HW ---- HW ----
A,.5,-.2,["HW",STANDARD,S=.1,U=0.0,X=-0.1,Y=-.05],-.2
```

This indicates a repeating pattern starting with a dash 0.5 drawing units long, a space 0.2 drawing units long, the characters HW with some scale and placement parameters, and another space 0.2 drawing units long. The text characters come from the text font assigned to the STANDARD text style at a scale of 0.1, an upright rotation of 0 degrees, an X offset of -0.1, and a Y offset of -0.05. This pattern continues for the length of the line, ending with a dash 0.5 drawing units long. The linetype would be displayed as shown below.

Notice that the total upstroke length is 0.2 + 0.2 = 0.4 and that the text origin is offset -.01 units in the X direction from the end of the first upstroke. An equivalent linetype would be

```
*HOT_WATER_SUPPLY,---- HW ---- HW ---- HW ---- HW ---- HW ----
A,.5,-.1,["HW",STANDARD,S=.1,U=0.0,X=0.0,Y=-.05],-.3
```

The total upstroke is still 0.1 + 0.3 = 0.4, but the text origin is not offset in the X direction.

Additional information about each field in the character descriptor follows. The values to be used are signed decimal numbers such as 1, -17, and 0.01.

- **text** The characters to be used in the linetype.
- **text style name** The name of the text style to be used. If no text style is specified, AutoCAD LT uses the currently defined style.
scale $S=value$. The scale factor to be used for the text style relative to the scale of the linetype. The height of the text style is multiplied by the scale factor. If the height is 0, the value for $S=value$ alone is used as the height.

rotation $U=value$, $R=value$, or $A=value$. $U=$ specifies upright or easy-to-read text. $R=$ specifies relative or tangential rotation with respect to the line. $A=$ specifies absolute rotation of the text with respect to the origin; that is, all text has the same rotation regardless of its position relative to the line. The value can be appended with a $d$ for degrees (degrees is the default value), $r$ for radians, or $g$ for grads. If rotation is omitted, 0 relative rotation is used. Rotation is centered between the baseline and the nominal cap height.

NOTE Drawings containing legacy linetypes that do not use the U (upright) rotation flag can be updated to the latest linetype definition by reloading the linetype from the LIN files. Custom linetypes can be updated by changing the R (rotation) flag to the U (upright) flag prior to reloading a linetype definition. For information on loading a linetype, see Load Linetypes.

xoffset $X=value$. The shift of the text on the $X$ axis of the linetype, which is along the line. If xoffset is omitted or is 0, the text is elaborated with no offset. Use this field to control the distance between the text and the previous pen-up or pen-down stroke. This value is not scaled by the scale factor defined by $S=value$, but it is scaled to the linetype.

yoffset $Y=value$. The shift of the text in the $Y$ axis of the linetype, which is at a 90-degree angle to the line. If yoffset is omitted or is 0, the text is elaborated with no offset. Use this field to control the vertical alignment of the text with respect to the line. This value is not scaled by the scale factor defined by $S=value$, but it is scaled to the linetype.

**Text in Custom Linetypes**

To include text characters in linetypes

1. Create a simple linetype, as described in To add a simple linetype to a LIN file (page 19).
2. Add the text character descriptor within the linetype pattern, using the following format:

   \[ "text",textstylename,\textit{scale},rotation,xoffset,yoffset] \]
Custom Hatch Patterns

AutoCAD LT® provides a library of standard hatch patterns in the `acadlt.pat` and `acadltiso.pat` files. You can use the hatch patterns as they are, modify them, or create your own custom hatch patterns.

Overview of Hatch Pattern Definitions

In addition to using the predefined hatch patterns that are supplied, you can design and create your own custom hatch patterns.

Developing a hatch pattern definition requires knowledge, practice, and patience. Because customizing hatches requires familiarity with hatch patterns, it is not recommended for new users.

The hatch patterns supplied by AutoCAD LT are stored in the `acadlt.pat` and `acadltiso.pat` text files. You can add hatch pattern definitions to this file or create your own files.

Regardless of where the definition is stored, a custom hatch pattern has the same format. It has a header line with a name, which begins with an asterisk and is no more than 31 characters long, and an optional description:

```
*pattern-name, description
```

It also has one or more line descriptors of the following form:

```
angle, x-origin, y-origin, delta-x, delta-y, dash-1, dash-2, ...
```

The default hatch pattern ANSI31 shown in the Boundary Hatch and Fill dialog box looks like this:

```
\|/\|/\|
```

and is defined as follows:
The pattern name on the first line, *ANSI31, is followed by a description: ANSI Iron, Brick, Stone masonry. This simple pattern definition specifies a line drawn at an angle of 45 degrees, that the first line of the family of hatch lines is to pass through the drawing origin (0,0), and that the spacing between hatch lines of the family is to be 0.125 drawing units.

Hatch pattern definitions follow these rules:

- Each line in a pattern definition can contain up to 80 characters. You can include letters, numbers, and the special characters underline (_), hyphen (-), and dollar sign ($). However, you must begin a pattern definition with a letter or number, not a special character.
- AutoCAD LT ignores both blank lines and text to the right of a semicolon.
- Each pattern line is considered to be the first member of a line family, created by applying the delta offsets in both directions to generate an infinite family of parallel lines.
- The delta-x value indicates the displacement between members of the family in the direction of the line. It is used only for dashed lines.
- The delta-y value indicates the spacing between members of the family; that is, it is measured perpendicular to the lines.
- A line is considered to be of infinite length. A dash pattern is superimposed on the line.

**NOTE** A blank line must be placed after the last hatch pattern definition in a PAT file. If a blank line is not placed after the last hatch pattern definition, the last hatch pattern definition will not be accessible when creating a hatch fill.

The process of hatching consists of expanding each line in the pattern definition to its infinite family of parallel lines. All selected objects are checked for intersections with any of these lines; any intersections cause the hatch lines to be turned on and off as governed by the hatching style. Each family of hatch lines is generated parallel to an initial line with an absolute origin to guarantee proper alignment.

If you create a very dense hatch, AutoCAD LT may reject the hatch and display a message indicating that the hatch scale is too small or its dash length too short. You can change the maximum number of hatch lines by setting the MaxHatch environment variable using SETENV and entering a number between 100 and 10000000 (ten million).
NOTE When changing the value of MaxHatch, you must enter MaxHatch with the capitalization as shown.

Overview of Hatch Pattern Definitions

To create a simple hatch pattern

1. Open the acadlt.pat or acadltiso.pat file in a text editor that saves in ASCII format.
2. Create a header line that includes an asterisk and a pattern name. The name of the hatch pattern is limited to 31 characters.
3. (Optional) To include a description in the header line, follow the pattern name with a comma and description text.
4. Create a descriptor line that includes
   ■ An angle at which the line is drawn
   ■ An $X,Y$ origin point
   ■ A $\Delta x$ of 0
   ■ A $\Delta y$ of any value

Hatch Patterns with Dashed Lines

To define dashed-line patterns, you append dash-length items to the end of the line definition item.

To define dashed-line patterns, you append dash-length items to the end of the line definition item. Each dash-length item specifies the length of a segment making up the line. If the length is positive, a pen-down segment is drawn. If the length is negative, the segment is pen-up, and it is not drawn. The pattern starts at the origin point with the first segment and cycles through the segments in circular fashion. A dash length of 0 draws a dot. You can specify up to six dash lengths per pattern line.

The hatch pattern ANSI33, looks like this:

```
\_\_\_\_\_\_
\_\_\_\_\_\_
```

and is defined as follows:
*ANSI33, ANSI Bronze, Brass, Copper
45, .176776695, 0, 0,.25, .125,-.0625

For example, to modify a pattern for 45-degree lines to draw dashed lines with a dash length of 0.5 units and a space between dashes of 0.5 units, the line definition would be

*DASH45, Dashed lines at 45 degrees
45, 0,0, 0,.5, .5,-.5

This is the same as the 45-degree pattern shown in Overview of Hatch Pattern Definitions (page 23), but with a dash specification added to the end. The pen-down length is 0.5 units, and the pen-up length is 0.5, meeting the stated objectives. If you wanted to draw a 0.5-unit dash, a 0.25-unit space, a dot, and a 0.25-unit space before the next dash, the definition would be

*DDOT45,Dash-dot-dash pattern: 45 degrees
45, 0,0, 0,.5, .5,-.25, 0,-.25

The following example shows the effect of \textit{delta-x} specifications on dashed-line families. First, consider the following definition:

*GOSTAK
0, 0,0, 0,.5, .5,-.5

This draws a family of lines separated by 0.5, with each line broken equally into dashes and spaces. Because \textit{delta-x} is zero, the dashes in each family member line up. An area hatched with this pattern would look like this:

\[ \begin{array}{cccc}
\_\_\_\_\_\_\_ \\
\_\_\_\_\_\_\_ \\
\_\_\_\_\_\_\_ \\
\_\_\_\_\_\_\_ \\
\end{array} \]

Now change the pattern to

*SKEWED
0, 0,0, .5,.5, .5,.5

It is the same, except that you have set \textit{delta-x} to 0.5. This offsets each successive family member by 0.5 in the direction of the line (in this case, parallel to the \textit{X} axis). Because the lines are infinite, the dash pattern slides down the specified amount. The hatched area would look like this:

\[ \begin{array}{cccc}
\_\_\_\_\_\_\_ \\
\_\_\_\_\_\_\_ \\
\_\_\_\_\_\_\_ \\
\_\_\_\_\_\_\_ \\
\end{array} \]
Hatch Patterns with Dashed Lines

To create a hatch pattern with dashed lines

1. Open the acadlt.pat or acadltiso.pat file in a text editor that saves in ASCII format.

2. Create a header line that includes an asterisk and a pattern name. The name of the hatch pattern is limited to 31 characters.

3. (Optional) To include a description in the header line, follow the pattern name with a comma and description text.

4. Create a descriptor line that includes
   - An angle at which the line is drawn
   - An X,Y origin point
   - A delta-x of any value if you want to offset alternating lines in the line family
   - A delta-y of any value
   - A value for a dash length
   - A value for a dot length
   - An optional second value for a different dash length
   - An optional second value for a different dot length

Hatch Patterns with Multiple Lines

Complex hatch patterns can have an origin that passes through offsets from the origin and can have multiple members in the line family.

Not all hatch patterns use origin points of 0,0. Complex hatch patterns can have an origin that passes through offsets from the origin and can have multiple members in the line family. In composing more complex patterns, you need to carefully specify the starting point, offsets, and dash pattern of each line family to form the hatch pattern correctly.

The hatch pattern AR-B816 looks like this:

```
+-----+-----+-----+
|     |     |     |
+-----+-----+-----+
|     |     |     |
```

and is defined as follows with multiple lines describing the pattern:
The following figure illustrates a squared-off, inverted-U pattern (one line up, one over, and one down). The pattern repeats every one unit, and each unit is 0.5 high and wide.

This pattern would be defined as follows:

*IUS, Inverted U's
90, 0,0, 0,1, .5,-.5
0, 0,.5, 0,1, .5,-.5
270, .5,.5, 0,1, .5,-.5

The first line (the up bar) is a simple dashed line with 0,0 origin. The second line (the top bar) should begin at the end of the up bar, so its origin is 0,.5. The third line (the down bar) must start at the end of the top bar, which is at .5,.5 for the first instance of the pattern, so its origin is at this point. The third line of the pattern could be the following:

90, .5,0, 0,1, .5,-.5

or

270, .5,1, 0,1, -.5,.5

The dashed pattern starts at the origin points and continues in the vector direction given by the angle specification. Therefore, two dashed-line families that are opposed 180 degrees are not alike. Two solid-line families are alike.

The following pattern creates six-pointed stars.
This example can help you refine your skills at pattern definition. (Hint: 0.866 is the sine of 60 degrees.)

The following is the AutoCAD LT definition of this pattern:

*STARS, Star of David
0, 0, 0, 0, 0.866, 0.5, -0.5
60, 0, 0, 0, 0.866, 0.5, -0.5
120, 0.25, 0.433, 0, 0.866, 0.5, -0.5

**Hatch Patterns with Multiple Lines**

To create a hatch pattern with multiple lines

1. Open the acadlt.pat or acadltiso.pat file in a text editor that saves in ASCII format.
2. Create a header line that includes an asterisk and a pattern name. The name of the hatch pattern is limited to 31 characters.
3. (Optional) To include a description in the header line, follow the pattern name with a comma and description text.
4. Create a descriptor line that includes:
   - An angle at which the line is drawn
   - An X,Y origin point
   - A delta-x of any value if you want to offset alternating lines in the line family
   - A delta-y of any value
   - A value for a dash length
   - A value for a dot length
   - An optional second value for a different dash length
   - An optional second value for a different dot length
5. Create a second line including all the parameters in the previous step.
6. (Optional) Create additional lines to complete the multiple-line hatch pattern.
When you work in the program, you use a variety of menus, tool sets, and shortcut menus to help you accomplish your tasks efficiently. You can also streamline your environment by customizing these elements.

**Understand User Interface Customization**

Using the customization tools of AutoCAD LT, you can tailor your drawing environment to suit your needs. The Customize dialog box helps you to easily create and modify the menus and tool sets that make up the user interface.

**Overview of the Customization**

Customization of the user interface is done with the Customize dialog box. From the Customize dialog box, you can
- Create new custom commands or modify existing commands
- Assign commands to various user interface elements
- Add or change menus that are displayed on the Mac OS menu bar
- Add or change tool sets that are displayed on the Tool Sets palette

**Customizable User Interface Elements**

The Customize dialog box allows you to create and manage commands that are used by the user interface. Along with commands, you are able to customize the following user interface elements
- Menu bar menus
Customization Glossary

You should know several terms for customizing AutoCAD LT 2013.

**Interface element** An object that can be customized, such as a menu or tool set.

**Interface item** The individual parts of a user interface element, such as a menu item or tool set tool.

**Macro** A series of commands that are run in a defined sequence to accomplish a drawing task.

**Palette** A modeless interface element that can be docked or floating outside of the drawing area. Palettes include the Properties Inspector, Reference Manager, Command Line, and so on.

**Tool set** An interface element that displays tool groups made up of commands and flyouts (or drop-downs) that are displayed vertically outside the drawing area.

**Tool group** An organizational structure used to lay out commands and flyouts (or drop-downs) for display on the Tool Sets palette.

Customize Commands

Commands in the Customize dialog box are used to define custom macros which are used to start standard and custom commands which can be executed from the command prompt in AutoCAD LT.

Overview of Commands

You can easily create, edit, and reuse commands. The Commands tab of the Customize dialog box allows you to create and modify existing commands that can then be added to a user interface element.

When you change the properties of a command in the Commands list, the properties of the command are changed everywhere the command is referenced. Each property of a command in the Commands list controls which
actions are taken when the command is used and how the command looks when added to a user interface element.

The following table shows the properties of the Scale command as they appear in the Properties section.

### Properties for the Scale command in the Commands list

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String displayed as the caption of a menu item or as a tooltip on the Tool Sets palette. The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_).</td>
<td>Scale</td>
</tr>
<tr>
<td>Description</td>
<td>String displayed as a tooltip when the cursor hovers over the tool on the Tool Sets palette.</td>
<td>Enlarges or reduces selected objects, keeping the proportions of the object the same after scaling</td>
</tr>
<tr>
<td>Macro</td>
<td>The command macro. It follows the standard macro syntax.</td>
<td>^C^C_scale</td>
</tr>
<tr>
<td>Image</td>
<td>ID string of the small-image resource (16 × 16 bitmap). The string must include alphanumeric characters with no punctuation other than a hyphen (-) or an underscore (_). It can also be a user-defined raster image file. Click the […] button to open the Select Image File dialog box.</td>
<td>RCDATA_16_SCALE</td>
</tr>
</tbody>
</table>

### Create, Edit, and Reuse Commands

You can create a new command from scratch, copy an existing command to create a new command, or edit the properties of an existing command.

When you change the properties of a command in the Commands list, the command is updated for all user interface elements that reference the command.
Create, Edit, and Reuse Commands

To create a custom command

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Commands tab, click Create New Command (+).
   A new command (named Command1) is added to the Commands list and properties for the new command are displayed in the Properties section.
3. In the Properties section, do the following:
   - In the Name box, enter a name for the command.
     The name is displayed in a tooltip on the Tool Sets palette and used as the caption for a menu item.
   - In the Description box, enter a description for the command.
     The description is displayed in a tooltip on the Tool Sets palette.
   - In the Macro box, enter a macro for the command.
   - In the Image box, click the [...] button to display the Select an Image File dialog box. Select the raster image you want to assign to the command, it should be 16x16 pixels.

To edit a command

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Commands tab, Commands list, select the command to edit.

   **NOTE**
   When you make a change to a command, the change is applied to all instances of the command in all menus and tool sets.
3. In the Properties section, do any of the following to edit the command:
   - In the Name box, enter a name for the command.
The name is displayed in a tooltip on the Tool Sets palette and used as the caption for a menu item.

■ In the Description box, enter a description for the command.
  The description is displayed in a tooltip on the Tool Sets palette.
■ In the Macro box, enter a macro for the command.
■ In the Image box, click the [...] button to display the Select an Image File dialog box. Select the raster image you want to assign to the command, it should be 16x16 pixels

To delete a command

1 At the Command prompt, enter `cui` and press Enter.
2 In the Customize dialog box, Commands tab, Commands list, select the command to delete.

  NOTE
  Commands assigned to a menu, tool set, shortcut key, or accelerator cannot be deleted.
3 Click the Options action menu below the Commands list, Gear icon, and click Delete.

To duplicate a command

1 At the Command prompt, enter `cui` and press Enter.
2 In the Customize dialog box, Commands tab, Commands list, select the command to duplicate.
3 Click the Options action menu below the Commands list, Gear icon, and click Duplicate.
  Make the desired changes to the new copy of the command.

To use a command

1 At the Command prompt, enter `cui` and press Enter.
2 In the Customize dialog box, click the Menus or Tool Sets tab.
3 In the Commands list, locate the command you want to use and drag it to the menu or tool set you want to add the command to.
Create Macros

A macro defines the action that results when an interface element is selected. A macro accomplishes a drawing task that would otherwise take a series of actions by a user.

Overview of Macros

A macro can contain commands, special characters, and DIESEL (Direct Interpretively Evaluated String Expression Language) programming code.

NOTE

As AutoCAD LT is revised and enhanced, the sequence of prompts for various commands (and sometimes command names) might change. Therefore, your custom macros might require minor changes when you upgrade to a new release of AutoCAD LT.

You add macros to interface elements by using the Customize dialog box. Select an existing command or create a new command in the Commands list on the Commands tab. Enter a macro in the Macros text box under the Properties section. There are no length limitations for macros. However, you do need to know how specific characters are used in macros and be aware of other considerations or limitations.

Macro Basics

A macro in a user interface element can be as simple as a command (such as circle) and some special characters (such as ^C^C).

For example, the macro ^C^C_circle \1, draws a circle with a radius of 1 unit. The components that define this macro are explained in the table below.

Components in CIRCLE macro

<table>
<thead>
<tr>
<th>Component</th>
<th>Component type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>^C^C</td>
<td>Special control character</td>
<td>Cancels any running commands</td>
</tr>
</tbody>
</table>
Components in CIRCLE macro

<table>
<thead>
<tr>
<th></th>
<th>Special control character</th>
<th>Automatically translates the command that follows into other languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIRCLE</td>
<td>Command</td>
<td>Starts the CIRCLE command</td>
</tr>
<tr>
<td>\</td>
<td>Special control character</td>
<td>Creates a pause for the user to specify the center point</td>
</tr>
<tr>
<td>1</td>
<td>Special control character</td>
<td>Responds to the prompt for the circle's radius (1)</td>
</tr>
</tbody>
</table>

For a list of special control characters that you can use in macros, see Use Special Control Characters in Macros (page 38).

Cancel Running Commands

Make sure that you have no AutoCAD LT commands in progress before you execute a macro. To automatically cancel a command before executing a macro, enter ^c^c at the beginning of the macro (which is the same as pressing ESC twice). Although a single ^c cancels most commands, ^c^c is required to return to the command prompt from a dimensioning command and ^c^c^c is required based on the current option of the -LAYER command. ^c^c^c handles canceling out of most command sequences and therefore is the recommended sequence used to ensure no command is active before the macro is started.

Verify Macro Characters

Every character in a macro is significant, even a blank space.

When you place a space at the end of the macro, AutoCAD LT processes the macro as though you had entered a command (circle, for example) and then pressed the Spacebar to complete the command.

Terminate Macros

Some macros require special terminators. Some commands (TEXT, for example) require you to press Enter rather than Spacebar to terminate the command. Some commands require more than one space (or Enter) to complete, but some text editors cannot create a line with trailing blanks.
Two special conventions resolve these problems.

- A semicolon (;) in a macro automatically issues Enter at the Command prompt.
- If a line ends with a control character, a backslash (\), a plus sign (+), or a semicolon (;), AutoCAD LT does not add a blank space after it.

An item that ends with a backslash (\) pauses a macro for user input.

Compare the following macros:

```plaintext
ucs
ucs ;
```

The first example enters `ucs` at the command prompt and presses Spacebar. The following prompt is displayed.

Specify origin of UCS or [Face/Named/Object/Previous/View/World/X/Y/Z/ZAxis] <World>:

The second example enters `ucs`, presses Spacebar, and presses Enter, which accepts the default value (World).

Suppress Echoes and Prompts in Macros

Characters in a macro appear at the command line as though you had typed the characters on the keyboard. This display duplication is called “echoing”.

You can suppress the “echoed” displays with the MENUECHO system variable. If echoes and prompts from item input are turned off, a ^P in the item turns them off.

Use Special Control Characters in Macros

You can use special characters, including control characters, in macros. In a macro, the caret (^) is equivalent to pressing the Command key on the keyboard. You can combine the caret with another character to construct macros that do such things as turn the grid on and off (^G) or cancel a command (^C).

The macro for the Address command below uses the backslash (\) to pause for user input and the semicolon (;) for Enter.

```plaintext
text \ .4 0 DRAFT Inc; ;Main St.; ;City, State;
```
The macro starts the TEXT command, pauses for the user to specify a start point, and then enters the address on three lines. In the triple semicolon (;;;), the first semicolon ends the text string, the second repeats TEXT, and the third accepts the default placement below the previous line.

Macros use the special characters listed in the following table.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>Issues Enter</td>
</tr>
<tr>
<td>^M</td>
<td>Issues Enter</td>
</tr>
<tr>
<td>^I</td>
<td>Issues Tab</td>
</tr>
<tr>
<td>[blank space]</td>
<td>Enters a space; a blank space between command sequences in a command is equivalent to pressing the Spacebar</td>
</tr>
<tr>
<td>\</td>
<td>Pauses for user input (cannot be used with accelerators)</td>
</tr>
<tr>
<td>_</td>
<td>Translates AutoCAD LT commands and options that follow</td>
</tr>
<tr>
<td>*^C^C</td>
<td>Repeats a command until another command is chosen</td>
</tr>
<tr>
<td>$</td>
<td>Introduces a conditional DIESEL macro expression ($M=)</td>
</tr>
<tr>
<td>^B</td>
<td>Turns Snap on or off (equivalent to Command-B)</td>
</tr>
<tr>
<td>^C</td>
<td>Cancels the active command or command option (equivalent to Esc)</td>
</tr>
<tr>
<td>^E</td>
<td>Sets the next isometric plane</td>
</tr>
<tr>
<td>^G</td>
<td>Turns Grid on or off (equivalent to Control-G)</td>
</tr>
<tr>
<td>^H</td>
<td>Issues Backspace</td>
</tr>
<tr>
<td>^O</td>
<td>Turns Ortho on or off</td>
</tr>
</tbody>
</table>

Customize Commands | 39
### Special characters used in macros

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^P</td>
<td>Turns MENUECHO on or off</td>
</tr>
<tr>
<td>^Q</td>
<td>Echoes all prompts, status listings, and input to the printer</td>
</tr>
<tr>
<td>^R</td>
<td>Turns command versioning on or off. Command versioning is required for some commands to ensure command macros written in an older release work properly in the latest release.</td>
</tr>
<tr>
<td>^V</td>
<td>Changes the current viewport</td>
</tr>
<tr>
<td>^Z</td>
<td>Null character that suppresses the automatic addition of Spacebar at the end of a command</td>
</tr>
</tbody>
</table>

### Pause for User Input in Macros

To accept input from the keyboard or pointing device in the middle of a command, place a backslash (\) in the macro at the point where you want input.

```
circle \1
```

In the circle example, \1 pauses for the user to specify the center point and then reads a radius of 1. Note that there is no space after the backslash.

```
-layer off \;
```

In this example, the macro starts -LAYER at the Command prompt, enters the Off option (off), and then pauses for the user to enter a layer name (\). The macro then turns that layer off and exits the -LAYER command (;).

**NOTE**

LAYER normally prompts for another operation and exits only if you press Spacebar or Enter. In the macro, the semicolon (;) is the equivalent of pressing Enter.
A macro typically resumes after one user input, such as a single point location. Therefore, you cannot construct a macro that accepts a variable number of inputs (as in object selection) and then continues. However, an exception is made for SELECT: a backslash (\) suspends the SELECT command until object selection has been completed. Consider the following example:

```
select \change previous ;properties color red ;
```

In this macro, SELECT creates a selection set of one or more objects (select \). The macro then starts CHANGE (change), references the selection set using the Previous option (previous;), and changes the color of all selected objects to red (properties color red ;).

**NOTE**
The backslash character (\) causes a macro to pause for user input. You cannot use a backslash for any other purpose in a macro. When you need to specify a file directory path, use a forward slash (/) as the path delimiter: for example, /direct/file.

The following circumstances delay resumption of a macro after a pause:

- If input of a point location is expected, object snap modes may be used before the point is specified.
- If X/Y/Z point filters are used, the command remains suspended until the entire point has been accumulated.
- For SELECT only, the macro does not resume until object selection has been completed.
- If the user responds with a transparent command, the suspended macro remains suspended until the transparent command is completed and the originally requested input is received.
- If the user responds by choosing another command (to supply options or to execute a transparent command), the original macro is suspended, and the newly selected item is processed to completion. Then, the suspended macro is resumed.

**NOTE** When command input comes from a command, the settings of the PICKADD and PICKAUTO system variables are assumed to be 1 and 0, respectively. This preserves compatibility with previous releases of AutoCAD LT and makes customization easier because you are not required to check the settings of these variables.
Provide International Support in Macros

To develop menus that can be used with a non-English-language version of AutoCAD LT, precede each command or option with the underscore character (_). The underscore character allows the standard commands and options to be translated automatically.

Repeat Commands in Macros

You can use a leading asterisk (*) to repeat a command in a macro until you choose another command.

Once you have selected a command, you might want to use it several times before moving on to another command. In a macro, you can repeat a command until you choose another command. You cannot use this feature to choose options.

If a macro begins with *^C^C, the command is repeated until you terminate by pressing Esc on the keyboard or by selecting another command.

**NOTE**

Do not use ^C (Cancel) within a macro that begins with the string **^C^C; this cancels the repetition.

The macros in the following examples repeat the commands:

```
*^C^Cmove Single
*^C^Ccopy Single
*^C^Cerase Single
*^C^Cstretch Single Crossing
*^C^Crotate Single
*^C^Cscale Single
```

Each macro in the example starts a command and then prompts you to select an object. Any other prompts necessary to complete the command are displayed, and then the command ends and starts again.

**NOTE**

Command repetition cannot be used in macros for image tile menus.
Use Single Object Selection Mode in Macros

Single Object Selection mode cancels the normal repetition of the Select Objects prompt in editing commands. After you select one object and respond to any other prompts, the command ends.

Consider the macro in the following example:

\[ *^C^C \text{Cerase single} \]

This macro terminates the current command and starts ERASE in Single Object Selection mode. After you choose this command, you either select a single object to be erased or click a blank area in the drawing and specify window selection. Any objects selected in this way are erased, and the command is repeated (due to the leading asterisk) so that you can erase additional objects. Press Esc to exit this mode.

Use Conditional Expressions in Macros

You can add conditional expressions to a macro by using a command that introduces macro expressions written in DIESEL (Direct Interpretively Evaluated String Expression Language).

The format is:

\[ $M=\text{expression} \]

Introducing the macro with $M= tells AutoCAD LT to evaluate a string as a DIESEL expression, and that expression is the DIESEL expression. The following example defines a conditional expression in a macro:

\[ \text{FILLMODE } $M=(-,1,$(getvar,fillmode)) \]

The macro switches the FILLMODE system variable on and off by subtracting the current value of FILLMODE from 1 and returning the resulting value to the FILLMODE system variable. You can use this method to toggle system variables whose valid values are 1 or 0.

Termination of Macros That Contain Conditional Expressions

If you use the DIESEL string language to perform “if-then” tests, conditions might exist where you do not want the normal terminating space or semicolon
(resulting in Enter). If you add ^z to the end of the macro, AutoCAD LT does not automatically add a space (Enter) to the end of the macro expression.

As with other control characters in commands, the ^z used here is a string composed of ^ (a caret) and z and is not equivalent to pressing Cmd-Z.

In the following examples, ^z is used as a macro terminator.

```
^C^CM=$(if,$(=,$(getvar,tilemode),0),$S=mview _mspace )^Z
^C^CM=$(if,$(=,$(getvar,tilemode),0),$S=mview _pspace )^Z
```

If these macros did not end with ^z, AutoCAD LT would automatically add a space (Enter), repeating the last command entered.

See also:

- Use Special Control Characters in Macros (page 38)
- DIESEL (page 57)

## Control the Display of Command Items

The way a menu item is displayed indicates its availability in the program.

A menu item can be displayed as:

- Grayed out (disabled)
- Marked with a check marker or border
- Both grayed out and marked

### Gray Out (Disable) Menu Items

You gray out a menu item by doing one of the following:

- Beginning a name with a tilde (~)
- Using a DIESEL string expression

For more information about using DIESEL expressions, see DIESEL Expressions in Macros. When grayed out, the macro and submenus associated with the menu item are made inaccessible.

DIESEL string expressions are used to conditionally disable or enable a menu item each time they are displayed. For example, the DIESEL string expression in the Macro text box in the Properties section disables the MOVE command while any other command is active.
Mark Menu Items

You can mark a menu item by doing one of the following:
- Beginning a command name with an exclamation point and a period (!.)
- Using a DIESEL string expression

A menu item is marked with or without a check mark.

Menu items can contain DIESEL string expressions to conditionally mark them each time they are displayed. When the following DIESEL string is added to the Macro text box for the applicable command in the Properties section on the Commands tab, a check mark is placed to the left of the menu item whose related system variable is currently enabled.

\[ \text{$(if,$(getvar,orthomode),!.)Ortho^O} ~\text{$(if,$(getvar,snapmode),!.)Snap^B} ~\text{$(if,$(getvar,gridmode),!.)Grid^G} \]

Simultaneously Disable and Mark Command Items

You can mark and disable commands at the same time using either of the following formats:

\[ \simbol{~!} \text{.labeltext} \]
\[ \simbol{!.} \text{~labeltext} \]

The tilde (\(\simbol{~}\)) is the special character code to disable a command and an exclamation point and period (\(\simbol{!} .\)) is the special character code to mark a command.

See also:

DIESEL Expressions in Macros (page 57)

Create Tooltips for Commands

Tooltips are descriptive messages that are displayed near the cursor when it hovers over a tool set tool.
The Description property associated to a command provides a simple description for what the command does. The value of the Description property is displayed as part of a basic tooltip that is displayed when the cursor hovers over a tool set tool. Along with the description, the name of the command is also displayed.

See also:

Customize Commands (page 32)

Create and Manage Images for Commands

Images can be assigned to commands, and can be of two different types: standard and custom. Standard images come with AutoCAD LT in a library, while external raster images can be assigned to commands in the Customize dialog box.

Assign Images to a Command

Standard and custom images can be assigned to a command.

When a command is added to a tool set, the assigned image is displayed on the tool.

AutoCAD LT comes with a library of standard images that are used for the standard commands; these can be assigned to your own custom commands. You can create your own custom commands using an external image editor.

See also:

Customize Commands (page 32)

Customize User Interface Elements

User interface elements in the Customize dialog box are used to control how commands can be started.
Menus

Menus are displayed on the Mac OS menu bar and are used to organize commands by general task.

Overview of Menus

Each menu can contain both standard and custom commands. Commands can be grouped with separators and sub-menus. If a menu is longer than the current display resolution in the vertical direction, it is truncated to fit.

When a menu is truncated, two arrows are added to the menu; one is added at the top and another to the bottom. With the arrows, you can scroll through the list of menu items. Sub-menus are indicated on a menu by an arrow that points to the right. When using a sub-menu, it appears to the right of its associated menu item normally. If a sub-menu reaches the edge of the display, additional nested sub-menus will be displayed to the left of the menu item.

Menu items on the Mac OS menu bar are executed by clicking it. Unlike a menu item, you do not need to click a sub-menu to access the assigned menu items. Position the cursor over the sub-menu to expand it and then click the menu item you want to use.

Create and Manage Pull-down Menus

You can create pull-down menus, and add commands and sub-menus for display on the Mac OS menu bar.

You can create and modify menus to display and organize the commands to best match the way you work. Commands are added to a menu from the Commands list in the Customize dialog box. Once commands are added to a menu, you can reposition commands, and organize commands using separators and sub-menus.

Sub-menus and Separators

Sub-menus are used to organize and group similar commands together. You create sub-menus in much the same way that you create a menu. Separators can also be inserted to group similar commands without adding an additional navigation level. There is no clear decision that can be made as to when you might use a sub-menu over a separator. Some possible reason why you might

Customize User Interface Elements | 47
consider to use a sub-menu over a separator is the current length of the menu and how frequently a command might be used.

See also:

Customize Commands (page 32)

Create and Manage Pull-down Menus

To create a pull-down menu

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Menus tab, Menus List, select the menu in which you want to create the new menu after.
3. Click the Create Menu Element (+) button below the Menus list and click Add Menu.
   A new menu (named Menu1) is created. The default name changes based on the number of menus you previously created.
4. Enter a name for the new menu.
   The name entered is what will appear on the Mac OS menu bar. Click a menu name twice to rename it.
5. In the Commands list, drag a command to the new menu.
6. Release the button on the pointer device when the menu or sub-menu is highlighted, or the desired location for the command is indicated by a horizontal line.
   For information about creating a command, see Create, Edit, and Reuse Commands (page 33).

To create a sub-menu

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Menus tab, Menus list, expand the menu you want to add a sub-menu to.
3. Select the menu item you want to insert the new sub-menu after.
4. Click the Create Menu Element (+) button below the Menus list and click Add Sub-menu.
   A new sub-menu (named Sub-menu1) is created. The default name changes based on the number of sub-menus you previously created.
5 Enter a name for the new menu. The name entered is what will appear on the Mac OS menu bar. Click a sub-menu name twice to rename it.

6 In the Commands list, drag a command to the new sub-menu.

7 Release the button on the pointer device when the menu or sub-menu is highlighted, or the desired location for the command is indicated by a horizontal line.

   For information about creating a command, see Create, Edit, and Reuse Commands (page 33).

To add a command to a pull-down menu or sub-menu

1 At the Command prompt, enter `cui` and press Enter.

2 In the Customize dialog box, click Menus tab.

3 In the Commands list, drag a command to the menu or sub-menu which you want to add a command.

   NOTE

   If you hold the cursor over a menu or sub-menu, it will expand over a short interval of time.

4 Release the button on the pointer device when a menu or sub-menu is highlighted, or the desired location for the command is indicated by a horizontal line.

   For information about creating a command, see Create, Edit, and Reuse Commands (page 33).

To change the display name for a pull-down menu, sub-menu, or command

1 At the Command prompt, enter `cui` and press Enter.

2 In the Customize dialog box, Menus tab, Menus list, select a menu, sub-menu, or menu item so it is highlighted.

3 You need to click in the Display Name column to edit the name displayed for a menu item.

4 Click the menu or menu item again to edit its name or display name.

5 Enter the new name and press Enter.
To insert a separator

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Menus tab, Menus list, select a sub-menu or menu item to insert a separator after.
3. Click the Options action menu below the Menus list, Gear icon, and click Insert Separator.

To duplicate a pull-down menu, sub-menu, command, or separator

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Menus tab, Menus list, select a menu, sub-menu, command, or separator item to duplicate.
3. Click the Options action menu below the Menus list, Gear icon, and click Duplicate.
4. Enter a new name for the duplicated item.

To delete a pull-down menu, sub-menu, command, or separator

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Menus tab, Menus list, select a menu, sub-menu, command, or separator item to delete.
3. Click the Options action menu below the Menus list, Gear icon, and click Delete.

**NOTE**

Be careful which item is selected because you cannot undo the deletion of the item. Click Cancel to abort the changes you made.

To reposition a pull-down menu, sub-menu, command, or separator

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Menus tab, Menus list, select a menu, sub-menu, command, or separator item to reposition.
3. Click and drag the selected item to its new location in the Menus list.
4. Release the button on the pointer device when the menu or sub-menu is highlighted, or the desired location for the command is indicated by a horizontal line.
Tool Sets

You can customize the Tool Sets palette by creating and modifying tool sets and tool groups.

Overview of Tool Sets and Tool Groups

Tool Sets are made up of tool groups that contain tools. A tool can be either a command or flyout (also known as a drop-down). Tool sets are accessed from the Tool Sets palette. Each tool group is a maximum of two tools wide and has two different display states: collapsed and expanded.

Position the cursor over a tool on the Tool Sets palette and click to use it. If the tool is a flyout (or drop-down), click and hold the button on the pointer device, and release over the tool you want to use.

Tools are not the only interactions that you will interact on the Tool Sets palette. When the cursor is positioned over a tool group, a disclosure triangle might appear that indicates the tool group can be expanded. Click the disclosure triangle to expand the panel to access additional tools. After a tool group is expanded, click the Lock icon to keep the tool group from collapsing.

See also:

Customize Commands (page 32)

Create and Manage Tool Sets

Tool sets are created to organize tool groups on the Tool Sets palette. AutoCAD LT comes with three tool sets that are used to organize drafting, annotation, and modeling tools. You create and manage tool sets on the Tool Sets tab of the Customize dialog box. After a tool set is created, you click the Tool Sets button near the top of the Tool Sets palette to set it current and use the commands and flyouts (drop-downs) assigned to it.

Tool Groups

Tool sets are often made up of multiple tool groups which are used to organize commands and flyouts (drop-downs). Each tool group is divided into two different parts, commands that are displayed by default and those that are displayed when the tool group is expanded.
You insert a separator to divide a tool group into the two parts, by default a tool group is not divided. Tools above the separator are those displayed by default. When using the Tool Sets palette, click the disclosure triangle to expand a tool group so you can access the tools that are hidden by default. Click the Lock icon when the tool group is expanded to keep the tool group from collapsing.

The order tool groups are displayed on the Tool Sets palette are controlled by the Customize dialog box. Drag a tool set up or down on the Tool Sets tab to change the order it appears on the Tool Sets palette.

**Flyouts and Separators**

Flyouts (or drop-downs) are used to help reduce the amount of space that related commands take up. Commands on a flyout are displayed by pressing and holding the button to display a menu with all the commands assigned to the flyout. Release the button over a tool from the flyout to execute the associated macro.

Separators can be added to a flyout to help provide an additional level of organize. When a separator is added to a flyout, a solid horizontal line is created across the flyout when it is displayed. A separator can be added to a tool group, but it is used to control which commands and flyouts are displayed by default and those that are available only when the tool group is expanded.

See also:

- [Customize Commands](#) (page 32)

**Create and Manage Tool Sets**

**To create a tool set**

1. At the Command prompt, enter `cui` and press Enter.
2. In the Customize dialog box, Tool Sets tab, Tool Sets list, select the tool set in which you want to create the new tool set after.
3. Click the Create Tool Set Element (+) button below the Tool Sets list and click Add Tool Set.
   
   A new tool set (named Tool Set1) is created. The default name changes based on the number of tool sets you previously created.
4. Enter a name for the new tool set.
The name entered is what will appear on drop-down menu when the Tool Sets button is clicked near the top of the Tool Sets palette. Click a tool set name twice to rename it.

5 Expand the new tool set.

6 In the Commands list, drag a command to the default tool group under the new tool set.

7 Release the button on the pointer device when the tool group or flyout is highlighted, or the desired location for the command is indicated by a horizontal line.

For information about creating a command, see Create, Edit, and Reuse Commands (page 33).

To create a tool group

1 At the Command prompt, enter `cui` and press Enter.

2 In the Customize dialog box, Tool Sets tab, Tool Sets list, expand the tool set you want to add a tool group to.

3 Select the tool group you want to insert the new tool group after.

4 Click the Create Tool Set Element (+) button below the Tool Sets list and click Add Tool Group.

   A new panel (named Panel1) is created. The default name changes based on the number of panels you previously created.

5 Enter a name for the new tool group.

   The name entered is displayed on the title bar of the tool group when it is expanded.

6 In the Commands list, drag a command to the new tool group.

7 Release the button on the pointer device when the tool group or flyout is highlighted, or the desired location for the command is indicated by a horizontal line.

   For information about creating a command, see Create, Edit, and Reuse Commands (page 33).

To create a flyout

1 At the Command prompt, enter `cui` and press Enter.

2 In the Customize dialog box, Tool Sets tab, Tool Sets list, expand the tool set and then the tool group you want to add a flyout to.

3 Select the command or flyout you want to insert the new flyout after.
4 Click the Create Tool Set Element (+) button below the Tool Sets list and click Add Drop-down.
   A new flyout (named Drop-down1) is created. The default name changes based on the number of flyouts you previously created.

5 Enter a name for the new flyout.
   The name entered is not displayed on the Tool Sets palette.

6 In the Commands list, drag a command to the new flyout.
7 Release the button on the pointer device when the flyout is highlighted, or the desired location is indicated by a horizontal line.
   For information about creating a command, see Create, Edit, and Reuse Commands (page 33).

To add a command to a tool group or flyout

1 At the Command prompt, enter `cui` and press Enter.
2 In the Customize dialog box, click Tool Sets tab.
3 In the Commands list, drag a command to the tool group or flyout which you want to add a command.

   **NOTE**
   If you hold the cursor over a tool group or flyout, it will expand after a short interval of time.

4 Release the button on the pointer device when a tool group or flyout is highlighted, or the desired location is indicated by a horizontal line.
   For information about creating a command, see Create, Edit, and Reuse Commands (page 33).

To insert a separator

1 At the Command prompt, enter `cui` and press Enter.
2 In the Customize dialog box, Tool Sets tab, Tool Sets list, select a tool group, flyout, or command to insert a separator after.
3 Click the Options action menu below the Tool Sets list, Gear icon, and click Insert Separator.

To duplicate a tool group, flyout, command, or separator

1 At the Command prompt, enter `cui` and press Enter.
2 In the Customize dialog box, Tool Sets tab, Tool Sets list, select a tool group, flyout, command, or separator to duplicate.

3 Click the Options action menu below the Tool Sets list, Gear icon, and click Duplicate.

4 Enter a new name for the duplicated item.

**To delete a tool group, flyout, command, or separator**

1 At the Command prompt, enter `cui` and press Enter.

2 In the Customize dialog box, Tool Sets tab, Tool Sets list, select a tool group, flyout, command, or separator to delete.

3 Click the Options action menu below the Tool Sets list, Gear icon, and click Delete.

**NOTE**
Be careful which item is selected because you cannot undo the deletion of the item. Click Cancel to abort the changes you made.

**To reposition a tool group, flyout, command, or separator**

1 At the Command prompt, enter `cui` and press Enter.

2 In the Customize dialog box, Tool Sets tab, Tool Sets list, select a tool group, flyout, command, or separator to reposition.

3 Click and drag the selected item to its new location in the Tool Sets list.

4 Release the button on the pointer device when the tool group or flyout is highlighted, or the desired location is indicated by a horizontal line.
Expressions in DIESEL (Direct Interpretively Evaluated String Expression Language) take strings as input and generate string results. DIESEL can be used in menu macros to make complex decisions.

**DIESEL Expressions in Macros**

You can use DIESEL string expressions in customization (CUIx) files as an additional method of creating macros.

These expressions can return string values (text strings) in response to standard AutoCAD LT commands. They can also return string values to the menu itself, thereby altering the appearance or content of a menu label.

A DIESEL expression that you use in a menu item must follow the $section=submenu format where the section name is and the submenu is the DIESEL expression you want.

Consider the following example:

```
^C^C^P$M=$(if,$(=,$(getvar,cvport),1),mspace,pspace)
```

This string provides a way to toggle between paper space and model space if TILEMODE is set to 0. This expression is evaluated transparently. If the special character ^P (which toggles MENUECHO on and off) is omitted, the expression displays only the issued command.

This menu item uses the DIESEL expression:

```
^C^C^P$M=$(if,$(=,$(getvar,cvport),1),mspace,pspace)
```

The next example is based on these assumptions:

- The CUIx excerpt is included in the current customization file.
- The symbols to insert are one unit high by one unit wide.
The DIMSCALE variable is set to the drawing's scale factor (that is, a drawing to be plotted at a scale of 1" = 10' would have a scale factor of 120, or a 1/4" = 1' scale drawing would have a scale factor of 48).

The DIESEL expressions in the following example multiply the current value of DIMSCALE by the specified value, and return an appropriate scale factor.

\[
\begin{align*}
M &= (*, (\text{getvar, dimscale}), 0.375) \\
M &= (*, (\text{getvar, dimscale}), 0.5) \\
M &= (*, (\text{getvar, dimscale}), 0.625)
\end{align*}
\]

DIESEL expressions can also return string values to pull-down menu item labels, so that you can make menus unavailable or otherwise alter the way they are displayed. To use a DIESEL expression in a pull-down menu label, make sure that the first character is the $ character.

In the next example, the current layer is set to BASE and the following DIESEL expression is used as the label.

\[
\text{$(eval, "Current layer: \" \$(\text{getvar, clayer})\")}$
\]

The result is that the appropriate pull-down menu is displayed and updated whenever the current layer changes.

**Current Layer: BASE**

**NOTE** The width of pull-down and shortcut menus is determined when the customization file is being loaded. Menu labels generated or changed by DIESEL expressions after a menu is loaded are truncated to fit within the existing menu width.

If you anticipate that a DIESEL-generated menu label will be too wide, you can use the following example to ensure that the menu width will accommodate your labels. This example displays the first 10 characters of the current value of the USERNAME system variable.

\[
\text{$(eval, "Current value: \" \$(\text{getvar, username})\") + $(if, $(eq, $(\text{getvar, username}), ""), 10 spaces )^C^CUsername}$
\]

You cannot use trailing spaces in a menu label to increase the menu width, because trailing spaces are ignored while the menu is being loaded. Any spaces you use to increase the width of a menu label must be within a DIESEL expression.

The next example uses the same DIESEL expression as the label and a portion of the menu item. It provides a practical way to enter the current day and date into a drawing.
Also, you can use a DIESEL macro to mark pull-down menu labels or make them unavailable. The following pull-down menu label displays an unavailable ERASE while a command is active. The text is displayed normally when a command is not active.

$(if,$(getvar,cmdactive),~)ERASE

You can use a similar approach to place a mark beside a pull-down menu item or to interactively change the character used for the mark.

You can use the CLIPBOARD system variable to enable or disable Clipboard commands on your menu. For example, the following DIESEL expression disables the Paste Special menu item when the Clipboard is empty:

$(if,$(getvar,clipboard),,,~)/Paste &Special...

^C^C_pastespec

Catalog of DIESEL Functions

Status retrieval, computation, and display are performed by DIESEL functions. All functions have a limit of 10 parameters, including the function name itself. If this limit is exceeded, you get a DIESEL error message.

+ (addition)

Returns the sum of the numbers val1, val2, ..., val9.

$+(, val1 [, val2, ..., val9])

If the current thickness is set to 5, the following DIESEL string returns 15.

$(+, $(getvar,thickness),10)

- (subtraction)

Returns the result of subtracting the numbers val2 through val9 from val1.

$(-, val1 [, val2, ..., val9])
* (multiplication)

Returns the result of multiplying the numbers \( val1, val2, \ldots, val9 \).

\[ s(*, \{val1, val2, \ldots, val9\}) \]

/ (division)

Returns the result of dividing the number \( val1 \) by \( val2, \ldots, val9 \).

\[ s(/, \{val1, val2, \ldots, val9\}) \]

= (equal to)

If the numbers \( val1 \) and \( val2 \) are equal, the string returns 1; otherwise, it returns 0.

\[ s(=, val1, val2) \]

< (less than)

If the number \( val1 \) is less than \( val2 \), the string returns 1; otherwise, it returns 0.

\[ s(<, val1, val2) \]

The following expression gets the current value of HPANG; if the value is less than the value stored in the system variable USERR1, it returns 1. If the value 10.0 is stored in USERR1 and the current setting of HPANG is 15.5, the following string returns 0.

\[ s(<, s(getvar, hpang), s(getvar, userr1)) \]

> (greater than)

If the number \( val1 \) is greater than \( val2 \), the string returns 1; otherwise, it returns 0.

\[ s(>, val1, val2) \]
!= (not equal to)

If the numbers \( \text{val1} \) and \( \text{val2} \) are not equal, the string returns 1; otherwise, it returns 0.

\[ S(\neq, \text{val1}, \text{val2}) \]

\( \leq \) (less than or equal to)

If the number \( \text{val1} \) is less than or equal to \( \text{val2} \), the string returns 1; otherwise, it returns 0.

\[ S(\leq, \text{val1}, \text{val2}) \]

\( \geq \) (greater than or equal to)

If the number \( \text{val1} \) is greater than or equal to \( \text{val2} \), the string returns 1; otherwise, it returns 0.

\[ S(\geq, \text{val1}, \text{val2}) \]

and

Returns the bitwise logical AND of the integers \( \text{val1} \) through \( \text{val9} \).

\[ S(\text{and}, \text{val1}[, \text{val2},..., \text{val9}]) \]

angtos

Returns the angular value in the format and precision specified.

\[ S(\text{angtos}, \text{value}[,, \text{mode}, \text{precision}]) \]
Edits the given value as an angle in the format specified by the mode and precision. (The values for mode are shown in the following table.) If mode and precision are omitted, it uses the current values chosen by the UNITS command.

Angular units values

<table>
<thead>
<tr>
<th>Mode value</th>
<th>String format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Degrees</td>
</tr>
<tr>
<td>1</td>
<td>Degrees/minutes/seconds</td>
</tr>
<tr>
<td>2</td>
<td>Grads</td>
</tr>
<tr>
<td>3</td>
<td>Radians</td>
</tr>
<tr>
<td>4</td>
<td>Surveyor's units</td>
</tr>
</tbody>
</table>

edtime

Returns a formatted date and time based on a given picture.

$\text{edtime}(time, picture)$

Edits the AutoCAD LT Julian date given by time (obtained, for example, from $\text{getvar(date)}$) according to the given picture. The picture consists of format phrases replaced by specific representations of the date and time. Characters not interpretable as format phrases are copied literally into the result of $\text{edtime}$. Format phrases are defined as shown in the following table. Assume that the date and time are Saturday, 5 September 1998 4:53:17.506.

edtime format phrases

<table>
<thead>
<tr>
<th>Format</th>
<th>Output</th>
<th>Format</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>5</td>
<td>H</td>
<td>4</td>
</tr>
<tr>
<td>DD</td>
<td>05</td>
<td>HH</td>
<td>04</td>
</tr>
</tbody>
</table>
edtime format phrases

<table>
<thead>
<tr>
<th>Format</th>
<th>Output</th>
<th>Format</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDD</td>
<td>Sat</td>
<td>MM</td>
<td>53</td>
</tr>
<tr>
<td>DDDD</td>
<td>Saturday</td>
<td>SS</td>
<td>17</td>
</tr>
<tr>
<td>M</td>
<td>9</td>
<td>MSEC</td>
<td>506</td>
</tr>
<tr>
<td>MO</td>
<td>09</td>
<td>AM/PM</td>
<td>AM</td>
</tr>
<tr>
<td>MON</td>
<td>Sep</td>
<td>am/pm</td>
<td>am</td>
</tr>
<tr>
<td>MONTH</td>
<td>September</td>
<td>A/P</td>
<td>A</td>
</tr>
<tr>
<td>YY</td>
<td>98</td>
<td>a/p</td>
<td>a</td>
</tr>
<tr>
<td>YYYY</td>
<td>1998</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enter the entire AM/PM phrase as shown in the preceding table; if AM is used alone, the A will be read literally and the M will return the current month.

If any AM/PM phrases appear in the picture, the H and HH phrases edit the time according to the 12-hour civil clock (12:00-12:59 1:00-11:59) instead of the 24-hour clock (00:00-23:59).

The following example uses the date and time from the preceding table. Notice that the comma must be enclosed in quotation marks because it is read as an argument separator.

```bash
$(edtime, $(getvar,date),DDD"," DD MON YYYY - H:MMam/pm)
```

It returns the following:
Sat, 5 Sep 1998 - 4:53am

If time is 0, the time and date at the moment that the outermost macro was executed is used. This avoids lengthy and time-consuming multiple calls on $(getvar,date) and guarantees that strings composed with multiple $(edtime) macros all use the same time.
**eq**

If the strings \texttt{val1} and \texttt{val2} are identical, the string returns 1; otherwise, it returns 0.

\[ (eq, val1, val2) \]

The following expression gets the name of the current layer; if the name matches the string value, it returns 1.

\[ $(eq,"PART12",$(getvar,clayer)) \]

Returns 1

**eval**

Passes the string \texttt{str} to the DIESEL evaluator and returns the result of evaluating it.

\[ $(eval, str) \]

**fix**

Truncates the real number \texttt{value} to an integer by discarding any fractional part.

\[ $(fix, value) \]

**getenv**

Returns the value of the environment variable \texttt{varname}.

\[ $(getenv, varname) \]

If no variable with that name is defined, it returns the null string.

**getvar**

Returns the value of the system variable with the given \texttt{varname}.

\[ $(getvar, varname) \]
**if**

Conditionally evaluates expressions.

\[ \text{S(if, expr, dotrue [, dofalse])} \]

If `expr` is nonzero, it evaluates and returns `dotrue`. Otherwise, it evaluates and returns `dofalse`. Note that the branch not chosen by `expr` is not evaluated.

**index**

Returns the specified member of a comma-delimited string.

\[ \text{S(index, which, string)} \]

Assumes that the `string` argument contains one or more values delimited by the macro argument separator character, the comma. The `which` argument selects one of these values to be extracted, with the first item numbered 0. This function is most frequently used to extract `X`, `Y`, or `Z` coordinate values from point coordinates returned by \$\text{(getvar)}\$.

**nth**

Evaluates and returns the argument selected by `which`.

\[ \text{S(nth, which, arg0 [, arg1, ..., arg7])} \]

If `which` is 0, `nth` returns `arg0`, and so on. Note the difference between \$\text{S(nth)}\$ and \$\text{S(index)}\$; \$\text{S(nth)}\$ returns one of a series of arguments to the function, while \$\text{S(index)}\$ extracts a value from a comma-delimited string passed as a single argument. Arguments not selected by `which` are not evaluated.

**or**

Returns the bitwise logical OR of the integers `val1` through `val9`.

\[ \text{S(or, val1 [, val2,..., val9])} \]
rtos

Returns the real value in the format and precision specified.

$rtos(value [, mode, precision])$

Edits the given value as a real number in the format specified by mode and precision. If mode and precision are omitted, it uses the current values selected with the UNITS command.

strlen

Returns the length of string in characters.

$strlen(string)$

substr

Returns the substring of string, starting at character start and extending for length characters.

$substr(string, start [, length])$

Characters in the string are numbered from 1. If length is omitted, it returns the entire remaining length of the string.

upper

Returns the string converted to uppercase according to the rules of the current locale.

$upper(string)$

xor

Returns the bitwise logical XOR of the integers $val1$ through $val9$.

$xor(val1 [, val2,..., val9])$
**DIESEL Error Messages**

Generally, if you make a mistake in a DIESEL expression, what went wrong will be obvious. Depending on the nature of the error, DIESEL embeds an error indication in the output stream.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$?</td>
<td>Syntax error (usually a missing right parenthesis or a runaway string)</td>
</tr>
<tr>
<td>$(func,??)</td>
<td>Incorrect arguments to <code>func</code></td>
</tr>
<tr>
<td>$(func)??</td>
<td>Unknown function <code>func</code></td>
</tr>
<tr>
<td>$(++)</td>
<td>Output string too long—evaluation truncated</td>
</tr>
</tbody>
</table>
Command Scripts

A script reads and executes commands from a text file.

You can run a script when you start AutoCAD LT®, or you can run a script from within AutoCAD LT using the SCRIPT command.

Create Command Scripts

A script is a text file that contains a series of commands. Common uses for scripts are to customize startup and to automate repetitive tasks.

Overview of Command Scripts

A script is a text file with one command on each line.

You can invoke a script at startup, or you can run a script during a work session by using the SCRIPT command. A script also provides an easy way to create continuously running displays for product demonstrations and trade shows.

You create script files outside the program using a text editor (such as TextEdit) that can save the file in ASCII format. The file extension must be .scr.

Each line of the script file contains a command. Each blank space in a script file is significant because Spacebar is accepted as a command or data field terminator. You must be very familiar with the sequence of prompts to provide an appropriate sequence of responses in the script file.

NOTE Keep in mind that prompts and command names may change in future releases, so you may need to revise your scripts when you upgrade to a later version of this program. For similar reasons, avoid the use of abbreviations; future command additions might create ambiguities.
A script can execute any command at the command prompt except a command that displays a dialog box. In most cases, a command that displays a dialog box has an alternative version of the command that displays command prompts instead of a dialog box.

Script files can contain comments. Any line that begins with a semicolon (;) is considered a comment, and it is ignored while the script file is being processed. The last line of the file must be blank.

All references to long file names that contain embedded spaces must be enclosed in double quotes. For example, to open the drawing my house.dwg from a script, you must use the following syntax:

```plaintext
open "my house"
```

The following commands are useful in scripts:

- `'DELAY` Provides a timed pause within a script (in milliseconds)
- `RESUME` Continues an interrupted script
- `RSCRIPT` Repeats a script file

When command input comes from a script, it is assumed that the settings of the PICKADD and PICKAUTO system variables are 1 and 0, respectively; therefore, you do not have to check the settings of these variables.

A script is treated as a group, a unit of commands, reversible by a single U command. However, each command in the script causes an entry in the undo log, which can slow script processing. If you like, you can use UNDO Control None to turn off the undo feature before running the script, or you can write it at the beginning of the script itself. Remember to turn it back on (UNDO Control All) when the script is finished.

The script that is running stops when another script command is invoked.

**Overview of Command Scripts**

**To create a script that changes settings in a drawing**

This script turns on the grid, sets the global linetype scale to 3.0, and sets layer 0 as the current layer with red as the color.

1. In a text editor, enter `grid on`.
2. On the next line, enter `ltscale 3.0`.
3. On the next line, enter `layer set 0 color red 0`. 
Add a blank line.

Save the file as ASCII text (TXT file), with a file extension of `.scr`.

The script file may contain comments, as follows:

```
; Turn grid on
grid on
; Set scale for linetypes
ltscale 3.0
; Set current layer and its color
layer set 0 color red 0
; Blank line above to end LAYER command
```

Run Scripts at Startup

A script that runs at startup can open a drawing and change its settings.

Suppose that every time you begin a new drawing, you turn on the grid, set the global linetype scale to 3.0, and set layer 0 as your current layer, with red as the color. You can do this using a drawing template, but you could do it instead with the following script and store it in a text file called `setup.scr`:

```
grid on
ltscale 3.0
layer set 0 color red 0
```

The first line turns on the grid. The second line sets the global scale for linetypes. The third line sets the current layer to layer 0 and sets its default color to red. AutoCAD LT assumes that in a script you want to use the version of LAYER command that displays command prompts rather than the palette version. The result is equivalent to entering `-layer` at the command prompt. The fourth line is blank, ending LAYER.

You could run a script at startup to open a drawing by using the following syntax in a Terminal window:

```
AutoCAD LT drawing_name -b setup
```

All file names that contain embedded spaces must be enclosed in double quotes, for example, “guest house”.

Including the file extensions `.app`, `.dwg`, `.dwt`, and `.scr` is optional. If AutoCAD LT cannot find the script file, AutoCAD LT reports that it cannot open the file.
To run the same script at startup but create a new drawing using the MyTemplate.dwt file as the template, enter the following in a Terminal window:

```
AutoCAD LT -t MyTemplate -b setup
```

This command creates a new drawing and issues a sequence of setup commands from the setup.scr file. When the script has finished running, the command prompt is displayed.

If you want to use the default template for the new drawing, you can omit the -t switch and the template file name.

**Run Scripts at Startup**

**To run a script at startup**

1. Click local drive ➤ Applications ➤ Utilities ➤ Terminal.
2. In the Terminal window, enter "AutoCAD LT" drawing_name -b script_name.

   **NOTE** AutoCAD listed above is the path to the AutoCAD executable. By default it is located at: /Applications/Autodesk/AutoCAD LT.app/Contents/MacOS/AutoCAD LT

   To start a new file, instead of a drawing file name, enter the -t switch and the name of a template file: -t template_drawing.

   The name of the script file must be the last parameter listed. The file extensions are optional.

3. Press Enter.

   AutoCAD LT opens the drawing and executes the commands in the script file. When the script has been completed, the Command prompt is displayed.
Index

A

addition DIESEL function 59
aliases
creating for commands 12
alignment
linetype patterns 17
alternate font mapping files 8
and DIESEL function 61
angtos DIESEL function 61

B

batch files 5
borders (menu items) 45

C

canceling commands 37
caret syntax in macros 38
characters
including in linetypes 19
macro control characters 36, 38
command line
switches 71
command line switches
running scripts and 71
command scripts 69
commands
about 32
aliases 12
canceling 37
creating 33
customizing 32
editing 33
Help topics about 46
images for 46
macros and 36, 37
menus 47
name validity 4
pausing for input 40
properties 32
repeating 42
reusing 33
scripts 69
submenus and 47
tool sets and tool groups 51
tooltips 46
transparent commands 40
comments
script commenting 70
conditional expressions in macros 43
control characters in macros 38
custom icon files 8
customization
about 1
customization files (CUI)
DIESEL expressions in 57
directory structure and 3
Customize dialog box
about 31
customizing
about customization 31
commands 32
Customize dialog box 31
functions and features 1
glossary of terms for 32
hatch patterns 23
hatches 23
linetypes 15
menus 47
tool groups 51
tool sets 51
user interface 31

D

dashes
hatch pattern definitions 25
linetype definitions 16
dates
edtime DIESEL function 62
DIESEL expressions
about 57
error messages 67
macros and 36, 43, 57
pull-down menu labels 58

DIESEL functions
addition 59
and 61
angtos 61
catalog 59
division 60
etime 62
eq 64
equal to 60
eval 64
fix 64
getenv 64
getvar 64
greater than 60
greater than or equal to 61
if 65
index 65
less than 60
less than or equal to 61
multiplication 60
not equal to 61
nth 65
or 65
rtos 66
strlen 66
substr 66
subtraction 59
upper 66
xor 66
directories 2
directory paths
program and support files 3
division DIESEL function 60
dots in linetype definitions 16
drawing files (DWG)
folders and maintenance 5
drawing templates
locating files 11
drop-downs
tool groups 52

DWT files
locating 11

E
echoes in macros 38
editing
commands 33
etime DIESEL function 62
eq DIESEL function 64
equal to DIESEL function 60
error messages
DIESEL 67
eval DIESEL function 64
extended help 46

F
files
locations 2
maintenance 5
roaming profiles and 7
fix DIESEL functions 64
flyouts
tool groups 52
FMP files (font mapping) 8
folders
file locations 2
font mapping files 8
fonts
including in linetypes 19

G
getenv DIESEL function 64
getvar DIESEL function 64
graphics for commands 46
graying out menu item labels 44
greater than DIESEL function 60
greater than or equal to DIESEL function 61
H

hatch patterns
  about  23
  creating 23
  examples 23, 25, 27
  multiple lines 27
hatches
  creating 23
  customizing 23
Help
  locating Help files  8
  tooltip command Help 46
I

icons
  customizations 8
if DIESEL function 65
images for commands 46
index DIESEL function 65
input
  pausing macros for 40
interface
  Customize dialog box 31
  customizing 31
  terminology 32
interface elements
  defined 32
interface items 32
international character sets 42
K

keyboard input in macros 40
L

languages
  macro translations 42
less than DIESEL function 60
less than or equal to DIESEL function 61
library search path 3
LIN files (linetype library) 8, 15
linetypes
  about 15
  creating 16
  customizing 15
  examples 16
  simple 16
  text characters in 19
M

macros
  about 36
  conditional expressions in 43
  defined 32
  DIESEL expressions in 43, 57
  international language support 42
  pausing for user input 38, 40
  repeating commands 42
  scripts 69
  selecting objects with 43
  special character codes for commands 38
  suppressing echoes and prompts 38
  syntax 36
  terminating 37
menu items
  DIESEL expressions and 58
  disabling 44
  graying out 44
menus
  creating 47
  creating sub-menu 47
  customizing 47
  international language support 42
  menu bar 47
  submenus 47
modifying
  commands 33
multiple-line hatch patterns 27
multiplication DIESEL function 60
N

naming
  linetypes 16
not equal to DIESEL function 61
nth DIESEL function 65

O
or DIESEL function 65

P
palettes
  defined 32
PAT files 8
patterns
  linetypes 17
pausing macros 38, 40
plot styles
  locating files 7
plotter configuration files
  locating files 8
prefixes for commands 12
profiles 7
program file locations 2
prompts
  suppressing 38
properties
  commands 32
PSF files 8

R
rearranging
  tools 52
repeating commands 42
ribbon
  command images 46
roaming profiles 7
rtos DIESEL function 66

S
SCR files (scripts) 69
scripts
  about 69
  file names and 70, 71
  running at startup 71
  switches and 71
  search paths
    default file locations 3
    selecting
      objects with macros 43
    spaces in linetypes 16
special characters
  commands and 47
  macros and 36, 38, 47
startup scripts 71
strlen DIESEL function 66
submenus 47
substr DIESEL function 66
subtraction DIESEL function 59
support files
  locations 2
  suppressing echoes and prompts 38
  switches
    running scripts and 71
symbols
  commands and 47
  macros and 36, 38, 47
syntax
  macros 36
system variables
  toggling values with macros 43

T
templates
  locating files 11
terminating macros 37
text
  including in linetypes 19
time
  edtime DIESEL function 62
tool groups
  customizing 51
  defined 32
  drop-downs on 52
  expanding 52
  flyouts on 52
tool sets
  creating 51
  customizing 51
  defined 32
  drop-downs 52
flyouts 52
Tool Sets palette 51
toolbars
  command images 46
tooltips
  command Help in 46
  translating macros 42
  transparent commands 40

U
UNT files 8

upper DIESEL function 66
user input
  pausing macros 40
user interface
  Customize dialog box 31
  customizing 31
  terminology 32

X
xor DIESEL function 66