Contents

Chapter 1 Basic Customization .................................................. 1
  Overview of Customization .............................................. 1
  Organize Program and Support Files ................................. 2
    Overview of File Organization ..................................... 3
    Multiple Drawing Folders ......................................... 5
    Locate Customized Files ........................................... 7
      Locate Plot Style Files ......................................... 7
      Locate Plotter Files ............................................ 8
      Locate Support Files ........................................... 9
      Locate Drawing Template Files ................................. 12
      Locate Texture Files ........................................... 13
    Create Command Aliases ............................................ 13
    Create Command Aliases ............................................ 14

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
    Shapes in Custom Linetypes ..................................... 22

Chapter 3 Custom Hatch Patterns .......................................... 25
< (less than) ................................................. 65
> (greater than) ........................................... 66
!= (not equal to) .......................................... 66
<= (less than or equal to) ......................... 66
>= (greater than or equal to) ..................... 66
and ......................................................... 66
angtos .................................................... 67
dtime ....................................................... 67
eq .......................................................... 69
eval ......................................................... 69
fix ......................................................... 69
getenv ..................................................... 69
getvar ..................................................... 70
if .......................................................... 70
index ....................................................... 70
nth ........................................................ 70
or .......................................................... 71
rtos ......................................................... 71
strlen ..................................................... 71
substr ..................................................... 71
upper ....................................................... 71
xor ........................................................ 72

DIESEL Error Messages .................................... 72

Chapter 6  Command Scripts ................................. 73
Create Command Scripts ................................ 73
Overview of Command Scripts ....................... 73
Run Scripts at Startup ................................ 75

Chapter 7  Introduction to Programming Interfaces .... 77
AutoLISP ..................................................... 77
Overview of AutoLISP ................................ 77
Use AutoLISP Applications .......................... 78
Automatically Load and Run AutoLISP Routines .... 80
Overview of AutoLISP Automatic Loading ........ 80
The ACAD.LSP File ..................................... 81
The ACADDOC.LSP File ................................ 81
The MNL File for an AutoLISP Menu ................. 83
Prevent AutoLISP Errors When Loading Startup Files . 83
S::STARTUP Function: Postinitialization Execution . 84
ObjectARX .................................................. 85
Overview of ObjectARX .............................. 86
Use ObjectARX Applications ....................... 86

Contents | v
Basic Customization

Overview of Customization

AutoCAD for Mac 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 Workflow palette or menu bar, and use DIESEL statements to create custom commands.

You can also use a number of powerful application programming interfaces (APIs) to add to and modify AutoCAD for Mac to suit your needs.

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.

- **Run external programs and utilities from within AutoCAD for Mac.** You can, for example, copy a disk or delete a file from within AutoCAD for Mac by adding the appropriate external command to the program parameters (PGP) file, acad.pgp.

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

- **Create custom linetypes, hatch patterns, shapes, and text fonts.** You can create linetypes, hatch patterns, shapes, and text fonts 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 Workflow 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 Workflow 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.

■ **Redefine or disable selected AutoCAD for Mac commands, either at the command prompt or as part of an AutoLISP or ObjectARX® program.** You can redefine certain AutoCAD for Mac commands to issue supplementary messages and instructions or, for example, to create a drawing management system in which the QUIT command is redefined to write billing information to a log file before ending the editing session.

In addition to the methods described in the *Customization Guide*, there are application programming interfaces (APIs) available for customizing AutoCAD for Mac. *Introduction to Programming Interfaces* (page 77) briefly describes these APIs and provides cross-references to more information.

See also:

- Organize Program and Support Files (page 2)
- Create Command Aliases (page 13)
- Custom Linetypes (page 15)
- Custom Hatch Patterns (page 25)
- DIESEL (page 61)
- Introduction to Programming Interfaces (page 77)
- Slides and Command Scripts (page 73)

---

### 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 for Mac uses support files for purposes such as storing customization definitions, loading AutoLISP and ObjectARX applications, and describing text fonts.

The default directory structure for the AutoCAD for Mac 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 for Mac 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 for Mac 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 for Mac 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 for Mac prompts you to finish the INSERT command in the usual manner.
Directory Structure

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

The default location for AutoCAD for Mac is in the Applications folder. You can create a new directory on the same level (for example, /AcadApps) and store your custom AutoLISP and ObjectARX application files, custom linetypes and hatch pattern files, and other third-party applications in subdirectories on the next level. If you want to maintain multiple drawing directories (for separate job files), you can create a directory such as /AcadJobs with subdirectories for each job.

Command Search Procedure

When you enter a command, AutoCAD for Mac 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, an external command or alias defined in the acad.pgp file, or a user-defined AutoLISP command. Commands can also be defined by ObjectARX applications or 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 or by an AutoLISP or ObjectARX application.

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

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

2. AutoCAD for Mac 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 for Mac 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 for Mac 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 for Mac checks the command name against the external commands defined in the program parameters file (acad.pgp). If the
command name corresponds to a defined external command, that command runs, and the search is complete.

5 AutoCAD for Mac checks the command name against the list of commands defined by AutoLISP or ObjectARX applications. At this point, an autoloaded command is loaded.

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

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

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

See also:
- Overview of AutoLISP Automatic Loading (page 80)
- 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 3), but you can expand or alter it to meet your needs.

You can set up the /AcadJobs 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 /AcadJobs/Job1/Support directory can contain blocks and AutoLISP files specific to the drawing files in /AcadJobs/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 for Mac, 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 for Mac 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 for Mac.

```bash
#!/bin/sh
prj="$1"
# Switch to the project folder and start AutoCAD
function startACAD() {
    cd /AcadJobs/Jobs/$prj
    echo "Starting AutoCAD"
    "/Applications/Autodesk/AutoCAD2012/AutoCAD.app/Contents/MacOS/AutoCAD"
}
# Clear Terminal and check for the existence of the folder
clear
cd .
if [ -d /AcadJobs/Jobs/$prj ]
then
    startACAD
fi
# Prompt to create folder
echo .
echo Creating /AcadJobs/Jobs/$prj
echo 'Press Y to continue (or A to abort)'
echo .
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
    # Your code here
```

6 | Chapter 1 Basic Customization
Using an ASCII text editor (such as TextEdit), save the batch program to a file named `acad.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 `acad.sh`, use the following syntax:

```
./acad.sh jobname
```

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

### Locate Customized Files

AutoCAD for Mac 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 for Mac 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 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ 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 for Mac command line.

 Locate Plotter Files

The location AutoCAD for Mac 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 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ 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 for Mac command line.

---

**Locate Support Files**

Support files include the following:
- Configuration file (`acad.cfg`)
- Custom icon files
- Help and miscellaneous files
- Font mapping file (`acad.fmp`)
- Alternate font file (`simplex.shx`)
- Support path files (`acad.lin`, `acad.mln`, `acad.mnl`, `acad.pat`, `acad.pgp`, `acad.psf`, `acad.unt`, `acadiso.lin`, `acadiso.pat`, `gdt.shx`, `inches.pss`, and `mm.pss`)

---

**Locate Support Files**

To find the default location of the configuration file

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ 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 Configuration File.
6. Under Configuration File, click the path name to view the location of your configuration file.
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 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ 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 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ 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 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ 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 2012 ➤ AutoCAD.
3 From the Mac OS menu bar, click AutoCAD ➤ 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 2012 ➤ AutoCAD.
3 From the Mac OS menu bar, click AutoCAD ➤ 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 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ Preferences.
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 for Mac 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 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ Preferences.
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.

**Locate Texture Files**

The locations AutoCAD for Mac looks in for material texture files are stored in the Application Preferences dialog box. You can use the OPTIONS command to locate which folders are being used to store your material texture files.

**Locate Texture Files**

To locate your texture files

1. From the Finder menu bar, click Go ➤ Applications.
2. In the Applications window, double-click Autodesk ➤ AutoCAD 2012 ➤ AutoCAD.
3. From the Mac OS menu bar, click AutoCAD ➤ 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 Texture Maps Search Path.
5. Under Texture Maps Search Path, click the path name to view the location of your texture 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 `c` instead of `circle` to start the CIRCLE command. An alias is not the same as a keyboard shortcut, which is a combination of keystrokes, such as Ctrl-S for SAVE.

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

**NOTE** Before you edit acad.pgp, create a backup so that you can restore it later, if necessary.

To define a command alias, add a line to the acad.pgp file using the following syntax:

```
abbreviation,*command
```

*abbreviation* is the command alias that you enter at the command prompt and *command* is the command being abbreviated. You must enter an asterisk (*) 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 (-) prefix, such as those listed here, that accesses the version of a command that displays command prompts instead of a dialog box.

```
BH, *-BHATCH
BD, *-BOUNDARY
```

**NOTE** You cannot use command aliases in command scripts. Using command aliases in custom commands is not recommended.

Restarting AutoCAD for Mac automatically reloads the file.

### Create Command Aliases

**To open the program parameters file (acad.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 2012/<version>/<language>/Support and double-click acad.pgp. If prompted for an application to use, select TextEdit.
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 for Mac 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 for Mac are acad.lin and acadiso.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 for Mac 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 for Mac 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 for Mac 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 for Mac 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 for Mac 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 `acad.lin` or `acadiso.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:

\[
\text{["text",textstylename,\text{scale},\text{rotation},xoffset,yoffset\]}
\]

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

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

This indicates a repeating pattern starting with a dash 0.5 drawing units long, a space 0.2 drawing units long, the characters \textit{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 \textit{X} offset of -0.1, and a \textit{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 -0.01 units in the \textit{X} direction from the end of the first upstroke. An equivalent linetype would be

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

The total upstroke is still 0.1 + 0.3 = 0.4, but the text origin is not offset in the \textit{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 for Mac 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:
Shapes in Custom Linetypes

A complex linetype can contain embedded shapes that are saved in shape files. Complex linetypes can denote utilities, boundaries, contours, and so on.

As with simple linetypes, complex lines are dynamically drawn as the user specifies vertices. Shapes and text objects embedded in lines are always displayed completely; they are never trimmed.

The syntax for complex linetypes is similar to that of simple linetypes in that it is a comma-delimited list of pattern descriptors. Complex linetypes can include shape and text objects as pattern descriptors, as well as dash-dot descriptors.

The syntax for shape object descriptors in a linetype description is as follows:

\[ \text{shapename,shxfilename} \] or \[ \text{shapename,shxfilename,transform} \]

where \text{transform} is optional and can be any series of the following (each preceded by a comma):

- \text{R=##} Relative rotation
- \text{A=##} Absolute rotation
- \text{U=##} Upright rotation
- \text{S=##} Scale
- \text{X=##} X offset
- \text{Y=##} Y offset

In this syntax, ## is a signed decimal number (1, -17, 0.01, and so on), the rotation is in degrees, and the remaining options are in linetype-scaled drawing units. The preceding transform letters, if they are used, must be followed by an equal sign and a number.

The following linetype definition defines a linetype named CON1LINE that is composed of a repeating pattern of a line segment, a space, and the embedded shape CON1 from the \text{ep.shx} file. (Note that the \text{ep.shx} file must be in the support path for the following example to work properly.)

```
*CON1LINE, --- [CON1] --- [CON1] --- [CON1]
A,1.0,-0.25,[CON1,ep.shx],-1.0
```
Except for the code enclosed in square brackets, everything is consistent with the definition of a simple linetype.

As previously described, a total of six fields can be used to define a shape as part of a linetype. The first two are mandatory and position-dependent; the next four are optional and can be ordered arbitrarily. The following two examples demonstrate various entries in the shape definition field.

[CAP, ep.shx, S=2, R=10, X=0.5]

The code above draws the CAP shape defined in the ep.shx shape file with a scale of two times the unit scale of the linetype, a tangential rotation of 10 degrees in a counterclockwise direction, and an X offset of 0.5 drawing units before shape elaboration takes place.

[DIP8, pd.shx, X=0.5, Y=1, R=0, S=1]

The code above draws the DIP8 shape defined in the pd.shx shape file with an X offset of 0.5 drawing units before shape drawing takes place, and a Y offset of one drawing unit above the linetype, with 0 rotation and a scale equal to the unit scale of the linetype.

The following syntax defines a shape as part of a complex linetype.

[shapename, shapefilename, scale, rotate, xoffset, yoffset]

The definitions of the fields in the syntax follow.

**shapename** The name of the shape to be drawn. This field must be included. If it is omitted, linetype definition fails. If shapename does not exist in the specified shape file, continue drawing the linetype but without the embedded shape.

**shapefilename** The name of a compiled shape definition file (SHX). If it is omitted, linetype definition fails. If shapefilename is unqualified (that is, no path is specified), search the library path for the file. If shapefilename is fully qualified and not found at that location, remove the prefix and search the library path for the file. If it is not found, continue drawing the linetype but without the embedded shape.

**scale** $s=value$. The scale of the shape is used as a scale factor by which the shape's internally defined scale is multiplied. If the shape's internally defined scale is 0, the $s=value$ alone is used as the scale.

**rotate** $U=value$, $R=value$ or $A=value$. $R=$ signifies relative or tangential rotation with respect to the line's elaboration. $A=$ signifies absolute rotation of the shape with respect to the origin; all shapes have the same rotation regardless of their relative position to the line. The value can be appended with a $d$ for
degrees (if omitted, degree is the default), \( \pi \) for radians, or \( g \) for grads. If rotation is omitted, 0 relative rotation is used.

**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 shape in the \( X \) axis of the linetype computed from the end of the linetype definition vertex. If \( xoffset \) is omitted or is 0, the shape is elaborated with no offset. Include this field if you want a continuous line with shapes. This value is not scaled by the scale factor defined by \( s=\). 

**yoffset** \( y=value \). The shift of the shape in the \( Y \) axis of the linetype computed from the end of the linetype definition vertex. If \( yoffset \) is omitted or 0, the shape is elaborated with no offset. This value is not scaled by the scale factor defined by \( s=\).

See also:

* Shapes and Shape Fonts (page 105)
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 for Mac are stored in the `acad.pat` and `acadiso.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 for Mac 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 for Mac 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 “MaxHatch” “n”) where n is 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 acad.pat or acadiso.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. 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:

```
```

Hatch Patterns with Dashed Lines | 27
and is defined as follows:

*ANSI33, ANSI Bronze, Brass, Copper
45, .176776695, 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, -.0625

This is the same as the 45-degree pattern shown in Overview of Hatch Pattern Definitions (page 25), 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 delta-x specifications on dashed-line families. First, consider the following definition:

*GOSTAK
0, 0, 0, 0, .5, .5, -.0625

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

---
---
---
---
---

Now change the pattern to

*SKEWED
0, 0, 0, .5, .5, .5, -.0625

It is the same, except that you have set 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 X axis). Because the lines are infinite, the dash pattern slides down the specified amount. The hatched area would look like this:
Hatch Patterns with Dashed Lines

To create a hatch pattern with dashed lines

1. Open the acad.pat or acadiso.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:

```
*AR-B816, 8x16 Block elevation stretcher bond
0, 0,0, 0,8
90, 0,0, 8,8, 8,-8
```

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 for Mac definition of this pattern:

```
*STARS, Star of David
0, 0, 0, .866, .5, -.5
60, 0, 0, .866, .5, -.5
120, .25, .433, 0, .866, .5, -.5
```

**Hatch Patterns with Multiple Lines**

**To create a hatch pattern with multiple lines**

1. Open the `acad.pat` or `acadiso.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.
Understand User Interface Customization

Using the customization tools of AutoCAD for Mac, 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
- Tool sets
Customization Glossary

You should know several terms for customizing AutoCAD 2012 for Mac.

**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 for Mac.

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.

<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.

**See also:**

*Create Menu Macros* (page 37)
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, DIESEL (Direct Interprettively Evaluated String Expression Language) or AutoLISP programming code.

NOTE

As AutoCAD for Mac 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 for Mac.

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.

<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>
<tr>
<td>_</td>
<td>Special control character</td>
<td>Automatically translates the command that follows into other languages</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>Command</td>
<td>Starts the CIRCLE command</td>
</tr>
</tbody>
</table>
Components in CIRCLE macro

<table>
<thead>
<tr>
<th></th>
<th>Special control character</th>
<th>Creates a pause for the user to specify the center point</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td></td>
<td></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 40).

Cancel Running Commands

Make sure that you have no AutoCAD for Mac 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 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 for Mac 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 for Mac 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:

```
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.

```
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>Allows you to access a built-in AutoCAD for Mac command even if it was undefined using the UNDEFINE command.</td>
</tr>
<tr>
<td>_</td>
<td>Translates AutoCAD for Mac 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>^D</td>
<td>Turns Dynamic UCS on or off (equivalent to Control-D)</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>
</tbody>
</table>
### Special characters used in macros

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^O</td>
<td>Turns Ortho on or off</td>
</tr>
<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 for Mac 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 for Mac, precede each command or option with the underscore character (_). The underscore character allows the standard commands and options to be translated automatically.

Use Built-in Commands in Macros

To develop macros that use built-in commands that are part of AutoCAD for Mac, precede each command with the period character (.). The period character allows the built-in command to be used even if it has been undefined with the UNDEFINE command making the macro predicable when it is used on other systems that share the same customization file.

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:
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^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=
```
expression

Introducing the macro with $M=$ tells AutoCAD for Mac 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:

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 for Mac 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 Ctrl-Z.

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

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

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

See also:

Use Special Control Characters in Menu Macros (page 40)
DIESEL (page 61)

Use AutoLISP in Macros

Creating commands that use AutoLISP is a more advanced way to use the AutoCAD for Mac customization feature.

You can use AutoLISP variables and expressions to create macros that perform complex tasks. To use AutoLISP efficiently in macros, place AutoLISP code in
a separate MNL file. AutoCAD for Mac loads the MNL file when it loads a customization file with the same name and in the same location.

Creating commands that use AutoLISP is a more advanced way to use the AutoCAD for Mac customization feature. Carefully study the following examples and the information in the AutoLISP Reference and the AutoLISP Developer's Guide.

**Preset Values**

An application that uses block insertion presets could provide commands like these: [Set WINWID][Set WALLTHK][Insert Window]

```
^C^C^P(setq WINWID (getreal "Enter window width: ")) ^P
^C^C^P(setq WALLTHK (getreal "Enter wall thickness: ")) ^P
^C^C_INSERT window XScale !WINWID YScale !WALLTHK
```

This code inserts the block named “window,” scaling its X axis to the current window width and its Y axis to the current wall thickness. In this example, the actual values come from the user-defined AutoLISP symbols WINWID and WALLTHK. The rotation is up to the user to decide so that the window can be rotated in the wall.

**Resize Grips**

With the following commands, grip size adjustment can be done on the fly:

```
^P(setvar "gripsize"(1+(getvar "gripsize")))(redraw)(princ)
^P(setvar "gripsize"(1-(getvar "gripsize")))(redraw)(princ)
```

To add validity checking to these commands, values less than 0 and greater than 255 cannot be used for the GRIPSIZE system variable.

**Prompt for User Input**

The following item prompts for two points and draws a rectangular polyline with the specified points as its corners.

```
^P(setq a (getpoint "Enter first corner: "));\+
(setq b (getpoint "Enter opposite corner: "));\+
pline !a (list (car a)(cadr b)) !b (list (car b)(cadr a))
c;^P
```
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.

\[$(\text{if},$(\text{getvar,cmdactive}),\sim)\text{MOVE}^\sim\text{move}$

The AutoLISP `menucmd` function can also be used to disable and enable menu items from a macro or application. For examples, see Reference Pull-Down or Shortcut Menus.

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.

```auto-lisp
$(if,$(getvar,orthomode),!.)Ortho^O
$(if,$(getvar,snapmode),!.)Snap^B
$(if,$(getvar,gridmode),!.)Grid^G
```

The AutoLISP `menucmd` function can be used to mark labels from a macro or application. For examples, see Reference Pull-Down or Shortcut Menus.

### Simultaneously Disable and Mark Command Items

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

```auto-lisp
~!.  
    labeltext
  !..~

  labeltext
```

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

See also:

- DIESEL Expressions in Macros (page 61)

### 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 34)
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 for Mac 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 for Mac 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 34)

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 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:**

[Cornell](#) (page 34)

### 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 35).

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 35).
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 35).

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 flytout (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:

**Create and Manage Tool Sets**

Tool sets are created to organize tool groups on the Tool Sets palette. AutoCAD for Mac 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:

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 35).
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 is indicated by a horizontal line.
   For information about creating a command, see Create, Edit, and Reuse Commands (page 35).

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 35).
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 35).

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.
You can also use DIESEL in menu items as a macro language instead of AutoLISP®. DIESEL expressions accept strings and generate string results.

Because DIESEL expressions handle strings exclusively, the USERS1-5 system variables are useful for passing information from an AutoLISP routine to a DIESEL expression. DIESEL expressions are evaluated by AutoLISP routines through the use of the AutoLISP menucmd function.

**DIESEL Expressions in Macros**

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

These expressions can return string values (text strings) in response to standard AutoCAD for Mac commands, AutoLISP and ObjectARX® routines, and other macros. 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 M and the submenu is the DIESEL expression you want. Frequently, you can implement a macro more easily with AutoLISP.

The following examples show two menu items that produce the same result; one uses DIESEL, and the other uses AutoLISP.

This menu item uses the DIESEL expression:

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

This menu item uses the AutoLISP expression:

```
^C^C^P(if (= (getvar "cvport") 1)(command "mspace")+(command "pspace"))(princ) ^P
```
Both menu items provide a way to switch between paper space and model space (if TILEMODE is set to 0), but the DIESEL expression is shorter and is evaluated transparently, not requiring the call to the AutoLISP princ function. If the special character ^P (which switches MENUECHO on and off) is omitted in both cases, the DIESEL expression displays only the issued command, whereas the AutoLISP expression displays the entire line of code.

Because the value returned by a DIESEL expression is a text string, it can be used in response to an AutoLISP getxxx function call. This functionality enables menu items to evaluate current drawing conditions and to return a value to an AutoLISP routine.

The next example is based on these assumptions:
- The AutoLISP routine is loaded into memory.
- 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).

If you load and execute the sample AutoLISP routine, AutoCAD for Mac inserts the symbol at the size and location you have specified. When plotted, the symbols are the specified size (if the drawing is plotted at the same scale as that specified by DIMSCALE).

The following is a sample AutoLISP routine.

```lisp
(defun C:SYMIN ( )
  (setq sym
    (getstring
      "\nEnter symbol name: ") ; Prompts for a symbol name
  )
  ; Display the custom toolbar named Symsize
  (command ".-_toolbar" "SymSize" "._show")
  (setq
    siz (getreal
      "\nSelect symbol size: ") ; Prompts for a symbol size
    p1 (getpoint
      "\nInsertion point: ") ; Prompts for insertion point
  )
  (command "._insert" ; Issues the INSERT command
    sym ; using the desired symbol
  )
)```

62 | Chapter 5  DIESEL
NOTE
An AutoLISP routine that you use regularly should include error checking to verify the validity of user input.

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

This cannot be done with similar AutoLISP code; a value returned by an AutoLISP expression cannot typically be used as a response to a `getxxx` function call (such as, the `getreal` function in the preceding sample).

\[
M = (.5 \times \text{getvar(dimscale)})
\]

\[
M = (0.375 \times \text{getvar(dimscale)})
\]

\[
M = (0.625 \times \text{getvar(dimscale)})
\]

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

You can also use this method to interactively change the text displayed in a pull-down menu. You use an AutoLISP routine that sets the USERS1-5 system variables to the selected text, which can be retrieved by a DIESEL macro in a menu label.

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 USERS3 (USERS1-5) system variable.

\[
\text{\$\{eval,"Current value: " \$\{getvar,users3\} +}
\text{\$\{if, \$\{eq,\$\{getvar,users3\},""\}, 10 \ spaces \}^\text{C}^\text{C} users3}
\]

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.

\[
\text{\$\{edtime,$\{getvar,date\},DDD", "D MON YYYY\}^\text{C}^\text{C} text +}
\\ \ \text{\$\{M=$\{edtime,$\{getvar,date\},DDD", "D MON YYYY\};}
\]

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.

\[
\text{\$\{if,$\{getvar,cmdactive\},~\}} \text{ERASE}
\]

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.

## 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 \(val1, val2, ..., val9\).

\[
\text{\$\{+, \{val1 [, \{val2, ..., \{val9\}\}\}\}}
\]

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

\[
\text{\$\{+, \$\{getvar,thickness\},10\}}
\]
- (subtraction)
Returns the result of subtracting the numbers val2 through val9 from val1.
$(-, \text{val1}, \text{val2}, ..., \text{val9})$

* (multiplication)
Returns the result of multiplying the numbers val1, val2, ..., val9.
$(*, \text{val1}, \text{val2}, ..., \text{val9})$

/ (division)
Returns the result of dividing the number val1 by val2, ..., val9.
$(/\, \text{val1}, \text{val2}, ..., \text{val9})$

= (equal to)
If the numbers val1 and val2 are equal, the string returns 1; otherwise, it returns 0.
$=, \text{val1}, \text{val2}$

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

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.
$<, $(\text{getvar,hpang}),$(\text{getvar,userr1})$
> (greater than)

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

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

!= (not equal to)

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

\[
\text{\textit{S}(!=, val1, val2)}
\]

<= (less than or equal to)

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

\[
\text{\textit{S}(<=, val1, val2)}
\]

>= (greater than or equal to)

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

\[
\text{\textit{S}(>=, val1, val2)}
\]

and

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

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

Returns the angular value in the format and precision specified.

$\text{(angtos, value [, mode, precision])}$

Edits the given value as an angle in the format specified by the mode and precision as defined for the analogous AutoLISP function. (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.

<table>
<thead>
<tr>
<th>Angular units values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode value</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

edtime

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

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

Edits the AutoCAD for Mac 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
\textit{S(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.

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
Format & Output & Format & Output \\
\hline
D & 5 & H & 4 \\
\hline
DD & 05 & HH & 04 \\
\hline
DDD & Sat & MM & 53 \\
\hline
DDDD & Saturday & SS & 17 \\
\hline
M & 9 & MSEC & 506 \\
\hline
MO & 09 & AM/PM & AM \\
\hline
MON & Sep & am/pm & am \\
\hline
MONTH & September & A/P & A \\
\hline
YY & 98 & a/p & a \\
\hline
YYYY & 1998 & & \\
\hline
\end{tabular}
\end{center}

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

If any \textit{AM/PM} phrases appear in the picture, the \textit{H} and \textit{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.

\texttt{$(edtime, $(getvar, date),DDD"," DD MON YYYY - H:MMam/pm)$}

It returns the following:
If \textit{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 \texttt{\$(getvar, date)} and guarantees that strings composed with multiple \texttt{\$(edtime)} macros all use the same time.

\textbf{eq}

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

\$$\text{eq, val1, val2}$$

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

\$$\text{eq,"PART12", \$(getvar, clayer)}$$

\textit{Returns} 1

\textbf{eval}

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

\$$\text{eval, str}$$

\textbf{fix}

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

\$$\text{fix, value}$$

\textbf{getenv}

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

\$$\text{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 varname.

\$\text{getvar, varname}\$

if

Conditionally evaluates expressions.

\$\text{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{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}\$.

Applications can use this function to retrieve values stored as comma-delimited strings from the USERS1-5 system variables.

nth

Evaluates and returns the argument selected by which.

\$\text{nth, which, arg0 [, arg1, ..., arg7]}\$

If which is 0, nth returns arg0, and so on. Note the difference between \$\text{nth}\$ and \$\text{index}\$; \$\text{nth}\$ returns one of a series of arguments to the function, while \$\text{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 \textit{val1} through \textit{val9}.

\texttt{\$or, val1 [, val2, \ldots, val9]} \\

**rtos**

Returns the real value in the format and precision specified.

\texttt{\$rtos, value [, mode, precision]} \\

Edits the given \textit{value} as a real number in the format specified by the \textit{mode} and \textit{precision} as defined by the analogous AutoLISP function. If \textit{mode} and \textit{precision} are omitted, it uses the current values selected with the UNITS command.

**strlen**

Returns the length of \textit{string} in characters.

\texttt{\$strlen, string} \\

**substr**

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

\texttt{\$substr, string, start [, length]} \\

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

**upper**

Returns the \textit{string} converted to uppercase according to the rules of the current locale.

\texttt{\$upper, string}
**xor**

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

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

You can run a script when you start AutoCAD for Mac®, or you can run a script from within AutoCAD for Mac 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:

```
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`.

---

74 | Chapter 6  Command Scripts
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:

```plaintext
; 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`.

```plaintext
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 for Mac 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:

```plaintext
AutoCAD 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 for Mac cannot find the script file, AutoCAD for Mac 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 -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 drawing_name -b script_name`.

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

   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 for Mac opens the drawing and executes the commands in the script file. When the script has been completed, the Command prompt is displayed.
AutoLISP

AutoLISP is based on the LISP programming language, which is simple to learn and very powerful. Because AutoCAD for Mac has a built-in LISP interpreter, you can enter AutoLISP code at the command prompt or load AutoLISP code from external files.

Overview of AutoLISP

AutoLISP is an application interface in AutoCAD for Mac that automating of design tasks. When an AutoLISP application or routine is loaded, it functions in its own namespace for each drawing that is open. A namespace is an insulated environment keeping AutoLISP routines that are specific to one drawing from having symbol or variable name and value conflicts with those in another drawing. For example, the following line of code sets a different value to the symbol \textit{a} for different documents.

\begin{verbatim}
(setq a (getvar "DWGNAME"))
\end{verbatim}

AutoLISP applications or routines can interact with AutoCAD for Mac in many ways. These routines can prompt the user for input, access built-in AutoCAD for Mac commands directly, and modify or create objects in the drawing database. By creating AutoLISP routines you can add discipline-specific commands to AutoCAD for Mac. Some of the standard AutoCAD for Mac commands are actually AutoLISP applications.
AutoLISP provides two file formats for applications:

- Reading an LSP file (\texttt{.lsp})—an ASCII text file that contains AutoLISP program code.
- Reading an FAS file (\texttt{.fas})—a binary, compiled version of a single LSP program file.

**NOTE**

Like-named AutoLISP application files are loaded based on their Modified time stamp; the LSP or FAS file with the most recent time stamp is loaded unless you specify the full file name (including the file name extension).

Because AutoCAD for Mac can read AutoLISP code directly, no compiling is required. This makes AutoLISP an easy language to experiment with, regardless of your programming experience.

Even if you are not interested in writing AutoLISP applications, your AutoCAD for Mac package includes many useful routines. Routines are also available as shareware through third-party developers. Knowing how to load and use these routines can enhance your productivity.

**NOTE** When command input comes from the AutoLISP \texttt{command} function, 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 for Mac and makes customization easier (because you don’t have to check the settings of these variables).

For information about AutoLISP programming, see the \textit{AutoLISP Developer's Guide}, and for information about AutoLISP, see the \textit{AutoLISP Reference} in the Help system.

## Use AutoLISP Applications

AutoLISP applications are stored in ASCII text files that you can edit. Before you can use an AutoLISP application, it must first be loaded.

AutoLISP applications are stored in ASCII text files with the \texttt{.lsp} extension. These files generally have a header portion that describes a routine, its use, and any specific instructions. This header might also include comments that document the author and the legal information regarding the use of the routine. Comments are preceded by a semicolon (\texttt{;}). You can view and edit these files with a text editor or word processor that can produce an ASCII text file.
Before you can use an AutoLISP application, it must first be loaded. You can use the APPLOAD command or the AutoLISP **load** function to load an application. Loading an AutoLISP application loads the AutoLISP code from the LSP file into your system's memory.

Loading an application with the **load** function involves entering AutoLISP code at the command prompt. If the **load** function is successful, it displays the value of the last expression in the file at the command prompt. This is usually the name of the last function defined in the file or instructions on using the newly loaded function. If **load** fails, it returns an AutoLISP error message. A **load** failure can be caused by incorrect coding in the file or by entering the wrong file name at the command prompt. The syntax for the **load** function is

```auto-lisp
(load filename [onfailure])
```

This syntax shows that the **load** function has two arguments: `filename`, which is required, and `onfailure`, which is optional. When loading an AutoLISP file at the command prompt, you typically supply only the `filename` argument. The following example loads the AutoLISP file `newfile.lsp`.

Command: `(load "newfile")`

The `.lsp` extension is not required. This format works for any LSP file in the current library path.

To load an AutoLISP file that is not in the library path, you must provide the full path and file name as the `filename` argument.

Command: `(load "/files/morelisp/newfile")`

---

**NOTE** When specifying a directory path, you must use a slash (/) or two backslashes (\\) as the separator, because a single backslash has a special meaning in AutoLISP.

---

See also:

- Overview of File Organization (page 3)
Automatically Load and Run AutoLISP Routines

You can load AutoLISP routines each time you run AutoCAD for Mac. You can also execute certain commands or functions at specific times during a drawing session.

Overview of AutoLISP Automatic Loading

AutoCAD for Mac loads the contents of two user-definable files automatically: `acad.lsp` and `acaddoc.lsp`.

By default, the `acad.lsp` file is loaded only once, when AutoCAD for Mac starts, whereas `acaddoc.lsp` is loaded with each individual document (or drawing). This lets you associate the loading of the `acad.lsp` file with application startup, and the `acaddoc.lsp` file with document (or drawing) startup. The default method for loading these startup files can be modified by changing the setting of the ACADLSPASDOC system variable.

If one of these files defines a function of the special type `S::STARTUP`, this routine runs immediately after the drawing is fully initialized. The `S::STARTUP` function is described in S::STARTUP Function: Postinitialization Execution (page 84). As an alternative, the APPLOAD command provides a Startup Suite option that loads the specified applications without the need to edit any files.

The `acad.lsp` and `acaddoc.lsp` startup files are not provided with AutoCAD for Mac. It is up to the user to create and maintain these files.

Command Autoloader

When you load an AutoLISP file, the command definitions in the file take up memory whether or not you actually use the commands. The AutoLISP `autoload` function makes a command available without loading the entire routine into memory. Adding the following code to your `acaddoc.lsp` file automatically loads the commands CMD1, CMD2, and CMD3 from the `cmds.lsp` file and the NEWCMD command from the `newcmd.lsp` file.

```lisp
(autoload "CMDS" '("CMD1" "CMD2" "CMD3"))
(autoload "NEWCMD" '("NEWCMD")
```

The first time you enter an automatically loaded command at the command prompt, AutoLISP loads the entire command definition from the associated
file. AutoLISP also provides the `autoarxload` function for ObjectARX applications. See `autoload` and `autoarxload` in the *AutoLISP Reference*.

**NOTE**

Like-named AutoLISP startup files are loaded based on their Modified time stamp; the LSP file with the most recent time stamp is loaded unless you specify the full file name (including the file name extension).

See also:

*S::STARTUP Function: Postinitialization Execution* (page 84)

**The ACAD.LSP File**

You can create an `acad.lsp` file if you regularly use specific AutoLISP routines. When you start AutoCAD for Mac, it searches the support file search path for an `acad.lsp` file. If an `acad.lsp` file is found, it is loaded into memory.

The `acad.lsp` file is loaded at each drawing session startup when AutoCAD for Mac is launched. Because the `acad.lsp` file is intended to be used for application-specific startup routines, all functions and variables defined in an `acad.lsp` file are only available in the first drawing. You will probably want to move routines that should be available in all documents from your `acad.lsp` file into the `acaddoc.lsp` file.

The recommended functionality of `acad.lsp` and `acaddoc.lsp` can be overridden with the ACADLSPASDOC system variable. If the ACADLSPASDOC system variable is set to 0 (the default setting), the `acad.lsp` file is loaded just once: upon application startup. If ACADLSPASDOC is set to 1, the `acad.lsp` file is reloaded with each new drawing.

The `acad.lsp` file can contain AutoLISP code for one or more routines, or just a series of `load` function calls. The latter method is preferable, because modification is easier. If you save the following code as an `acad.lsp` file, the files `mysessionapp1.lsp`, `databasesynch.lsp`, and `drawingmanager.lsp` are loaded every time you start AutoCAD for Mac.

```lisp
(load "mysessionapp1")
(load "databasesynch")
(load "drawingmanager")
```
WARNING  Do not modify the reserved acad2012.lsp file. Autodesk provides the acad2012.lsp file, which contains AutoLISP defined functions that are required by AutoCAD for Mac. This file is loaded into memory immediately before the acad.lsp file is loaded.

See also:

- Overview of File Organization (page 3)
- Prevent AutoLISP Errors When Loading Startup Files (page 83)

The **ACADDOC.LSP** File

The acaddoc.lsp file is intended to be associated with each document (or drawing) initialization. This file is useful if you want to load a library of AutoLISP routines to be available every time you start a new drawing (or open an existing drawing).

Each time a drawing opens, AutoCAD for Mac searches the library path for an acaddoc.lsp file. If it finds one, it loads the file into memory. The acaddoc.lsp file is always loaded with each drawing regardless of the settings of ACADLSPASDOC.

Most users will have a single acaddoc.lsp file for all document-based AutoLISP routines. AutoCAD for Mac searches for an acaddoc.lsp file in the order defined by the library path; therefore, with this feature, you can have a different acaddoc.lsp file in each drawing directory, which would load specific AutoLISP routines for certain types of drawings or jobs.

The acaddoc.lsp file can contain AutoLISP code for one or more routines, or just a series of load function calls. The latter method is preferable, because modification is easier. If you save the following code as an acaddoc.lsp file, the files mydocumentapp1.lsp, build.lsp, and counter.lsp are loaded every time a new document is opened.

```
(load "mydocumentapp1")
(load "build")
(load "counter")
```

WARNING  Do not modify the reserved acad2012doc.lsp file. Autodesk provides the acad2012doc.lsp file, which contains AutoLISP-defined functions that are required by AutoCAD for Mac. This file is loaded into memory immediately before the acaddoc.lsp file is loaded.
The MNL File for an AutoLISP Menu

When AutoCAD for Mac loads a customization file, it searches for an MNL file with a matching file name. If it finds the file, it loads the file into memory. This function ensures that AutoCAD for Mac loads the AutoLISP functions that are needed for proper operation of a menu.

This function ensures that AutoCAD for Mac loads the AutoLISP functions that are needed for proper operation of a menu. For example, the default AutoCAD for Mac customization file, \textit{acad.cuix}, relies on the file \textit{acad.mnl}. This file defines numerous AutoLISP functions used by the menu. The MNL file is loaded after the \textit{acaddoc.lsp} file.

Prevent AutoLISP Errors When Loading Startup Files

If an AutoLISP error occurs while you are loading a startup file, the remainder of the file is ignored and is not loaded.

Files specified in a startup file that do not exist or that are not in the AutoCAD for Mac library path generally cause errors. Therefore, you may want to use the \texttt{onfailure} argument with the \texttt{load} function. The following example uses the \texttt{onfailure} argument:

```lisp
(princ (load "mydocapp1" "\nMYDOCAPP1.LSP file not loaded.\n"))
(princ (load "build" "\nBUILD.LSP file not loaded.\n"))
(princ (load "counter" "\nCOUNTER.LSP file not loaded.\n"))
(princ)
```

If a call to the \texttt{load} function is successful, it returns the value of the last expression in the file (usually the name of the last defined function or a message regarding the use of the function). If the call fails, it returns the value of the \texttt{onfailure} argument. In the preceding example, the value returned by the
The load function is passed to the princ function, causing that value to be displayed at the command prompt.

For example, if an error occurs while AutoCAD for Mac loads the mydocapp1.lsp file, the princ function displays the following message and AutoCAD for Mac continues to load the two remaining files:

MYDOCAPP1.LSP file not loaded.

If you use the command function in an acad.lsp or acaddoc.lsp, it should be called only from within a defun statement. Use the S::STARTUP function to define commands that need to be issued immediately when you begin a drawing session.

See also:
S::STARTUP Function: Postinitialization Execution (page 84)

### S::STARTUP Function: Postinitialization Execution

You can define an S::STARTUP function to perform any needed setup operations after the drawing is initialized.

The startup LISP files (acad.lsp, acaddoc.lsp, and MNL) are all loaded into memory before the drawing is completely initialized. Typically, this does not pose a problem, unless you want to use the command function, which is not guaranteed to work until after a drawing is initialized.

If the user-defined function S::STARTUP is included in an acad.lsp, acaddoc.lsp, or MNL file, it is called when you enter a new drawing or open an existing drawing. Thus, you can include a definition of S::STARTUP in the AutoLISP startup file to perform any setup operations.

For example, if you want to override the standard HATCH command by adding a message and then switching to the BHATCH command, use an acaddoc.lsp file that contains the following:

```lisp
(defun C:HATCH ()
  (alert "Using the BHATCH command!")
  (princ \nEnter OLDHATCH to get to real HATCH command.\n"

  (command "BHATCH")
  (princ)
)
```

84 | Chapter 7  Introduction to Programming Interfaces
Before the drawing is initialized, new definitions for HATCH and OLDHATCH are defined with the defun function. After the drawing is initialized, the S::STARTUP function is called and the standard definition of HATCH is undefined.

**NOTE** To be appended, the S::STARTUP function must have been defined with the defun-q function rather than defun.

Because an S::STARTUP function can be defined in many places (an acad.lsp, acad.doc.lsp, or MNL file or any other AutoLISP file loaded from any of these), it’s possible to overwrite a previously defined S::STARTUP function.

The following example shows one method of ensuring that your startup function works with other functions.

```lisp
(defun-q MYSTARTUP ()
  ... your startup function ...
)
(setq S::STARTUP (append S::STARTUP MYSTARTUP))
```

The previous code appends your startup function to that of an existing S::STARTUP function and then redefines the S::STARTUP function to include your startup code. This works properly regardless of the prior existence of an S::STARTUP function.

**ObjectARX**

ObjectARX technology provides the foundation for design software applications to share intelligent object data. You can run third-party ObjectARX application programs or write your own.
Overview of ObjectARX

ObjectARX® (AutoCAD for Mac Runtime Extension) is a compiled-language programming environment for developing AutoCAD for Mac applications. The ObjectARX programming environment allows you to load and run your compiled projects in the same address space as AutoCAD for Mac. This allows your programs to operate directly with core AutoCAD for Mac data structures and code.

The ObjectARX libraries allow you to take advantage of the AutoCAD for Mac open architecture, providing direct access to the AutoCAD for Mac database structures, graphics system, and AutoCAD for Mac geometry engine to extend AutoCAD for Mac classes and capabilities at runtime. Additionally, you can define new commands that operate exactly the same way as native AutoCAD for Mac commands.

You can use ObjectARX libraries in conjunction with AutoLISP, enabling cross-API integration.

The ObjectARX programming environment is described in the ObjectARX Developer's Guide. The documentation is part of the ObjectARX Software Development Kit (SDK), which can be downloaded from the Development Tools section of the Autodesk website.

Use ObjectARX Applications

To load an ObjectARX application, you use the Load option of the ARX command. After loading, all commands defined by this application are available at the command prompt.

Some ObjectARX applications use large amounts of system memory. If you are finished using an application and want to remove it from memory, use the Unload option of ARX.

You can also load an ObjectARX application with the arxload AutoLISP function. The syntax for the arxload function is almost identical to that of the load function used with AutoLISP files. If the arxload function loads the ObjectARX program successfully, it returns the program name. The syntax for the arxload function is as follows:

```
(arxload
   filename [onfailure]
```
The two arguments for the `arxload` function are `filename` and `onfailure`. As with the `load` function, the `filename` argument is required and must be the complete path name description of the ObjectARX program file to load. The `onfailure` argument is optional and typically not used when you load ObjectARX programs from the command prompt. The following example loads the ObjectARX application `myapp.arx`.

```
(arxload "myapp")
```

As with AutoLISP files, AutoCAD for Mac searches the library path for the specified file. If you need to load a file that is not in the library path, you must provide the full path name description of the file.

NOTE
When specifying a directory path, you must use a slash (/) or two backslashes (\\) as the separator, because a single backslash has a special meaning in AutoLISP.

Attempting to load an application that has previously been loaded results in an error. Before using `arxload` you should use the `arx` function to check the currently loaded applications.

To unload an application with AutoLISP, use the `arxunload` function. The following example unloads the `myapp` application.

```
(arxunload "myapp")
```

Using the `arxunload` function not only removes the application from memory but also removes the command definitions associated with that application.

See also:
- Overview of File Organization (page 3)

---

**Install and Uninstall Plug-In Applications**

The plug-in auto loader mechanism allows for the easy deployment of custom applications using a package format.

The package format is a common folder structure than contains the extension `.bundle` in its name and has an XML file that defines the various components.
of the plug-in. By deploying custom applications as a BUNDLE, it makes it
easier to target multiple operating systems and product releases since the
parameters of your plug-in are defined in the XML file of the package. A
BUNDLE can be used as a replacement to creating complex installer scripts
when deploying a plug-in.

The plug-in defined by each package is loaded into AutoCAD for Mac by
placing it in the /Applications/Autodesk/ApplicationAddins folder on your local
drive. When AutoCAD for Mac starts, the ApplicationAddins folder is checked
for plug-in applications. The packages found are automatically registered and
loaded based on the metadata in the XML file of each package.

**Install Plug-in Packages**

A package can be deployed using an installer or manually copying the files
and folder structure to the ApplicationAddins folder.

**Loading Plug-ins**

By default, plug-ins are automatically registered with AutoCAD and when a
new plug-in is installed during the current session. The load behavior for
plug-ins is controlled with the APPAUTOLOAD system variable. When
APPAUTOLOAD is set to 0, no plug-ins are loaded unless the APPAUTOLOADER
command is used.

**Uninstall Plug-in Packages**

A package can be uninstalled by removing the appropriate folder with a .bundle
extension from the ApplicationAddins folder.

**PackageContents.xml Format**

The PackageContents.xml file contains information about the application
package, including information about the developer that authored it.

The information contained in the file can be used to specify which
AutoCAD-based products and releases the plug-in can be loaded into, their
supported Operating System, and how the application should be loaded, such
as load on startup or when a command is invoked.

The following elements are used to define and describe the plug-in:

- ApplicationPackage (page 91)
- CompanyDetails (page 93)
PackageContents.xml for Download versus Local Deployment

How you plan to deploy your plug-in determines what should be in the PackageContents.xml file. You need to supply more information when providing a plug-in for download versus a local deployment. The following table explains which elements and attributes are required, optional, or recommended for the deployment you want to use.

<table>
<thead>
<tr>
<th>Element</th>
<th>For Download from the Apps tab on Autodesk Exchange</th>
<th>For Local Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationPackage element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SchemaVersion</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>AppVersion</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>Author</td>
<td>Required</td>
<td>Optional</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Recommended</td>
</tr>
<tr>
<td>Description</td>
<td>Required</td>
<td>Recommended</td>
</tr>
<tr>
<td>Icon</td>
<td>Required</td>
<td>Recommended</td>
</tr>
<tr>
<td>Helpfile</td>
<td>Required</td>
<td>Recommended</td>
</tr>
<tr>
<td>ProductCode</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>UpgradeCode</td>
<td>Required</td>
<td>Optional</td>
</tr>
<tr>
<td>CompanyDetails element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Optional</td>
</tr>
<tr>
<td>Phone</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>For Download from the Apps tab on Autodesk Exchange</td>
<td>For Local Deployment</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>Email</td>
<td>Required</td>
<td>Optional</td>
</tr>
</tbody>
</table>

RuntimeRequirements element – Required, if Components element present

| OS                                                   | Optional             | Optional |
| Platform                                             | Optional             | Optional |
| SeriesMin                                            | Optional             | Optional |
| SeriesMax                                            | Optional             | Optional |
| SupportPath                                          | Optional             | Optional |

ComponentEntry element – Required, if Components element present

| AppName                                              | Required             | Required |
| AppDescription                                       | Optional             | Optional |
| AppType                                              | Optional             | Optional |
| ModuleName                                           | Required             | Required |
| PerDocument                                          | Optional             | Optional |
| LoadReasons                                          | Optional             | Optional |

Commands element

<p>| GroupName                                            | Required             | Optional |</p>
<table>
<thead>
<tr>
<th>For Download from the Apps tab on Autodesk Exchange</th>
<th>For Local Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command element – Required, if Commands element present</td>
<td></td>
</tr>
<tr>
<td>Global</td>
<td>Required</td>
</tr>
<tr>
<td>Local</td>
<td>Required</td>
</tr>
</tbody>
</table>

**ApplicationPackage Element**

Each *PackageContents.xml* file must contain an ApplicationPackage element. The ApplicationPackage element, in the form of XML Attributes, contains general information about the plug-in. It also encapsulates other the element types that help to define the contents of the plug-in.

An ApplicationPackage element can have any of the following attributes attached to it:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SchemaVersion</strong></td>
<td>PackageContents.xml format version number. The value should always be 1.0 until a newer version of the schema is introduced.</td>
</tr>
<tr>
<td><strong>AppVersion</strong></td>
<td>Application version number. AutoCAD for Mac uses this value to determine if the installed version is the latest version. If an updated version is available, the user is informed and able to download and install the latest version. It is recommended to use an application version that includes major and minor values, such as “1.0.0.0”.</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Name of the plug-in author.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Plug-in name. A localized plug-in name can be specified by combining Name with a locale code. See Supported Locale Codes (page 100) for a full list of supported locale codes.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>Short description of the plug-in. Localized descriptions can be specified by combining Description with a locale code. See Supported Locale Codes (page 100) for a full list of supported locale codes.</td>
</tr>
<tr>
<td>Icon</td>
<td>Icon for the plug-in; used in the installer and the Apps tab on Autodesk Exchange. The icon should be 32x32 pixels in size and support 32-bit (Truecolor) color depth. Recommend using a BMP or ICO file format. NOTE All path specifiers are ‘/’ and not ‘\’, and paths are relative to the root .bundle folder.</td>
</tr>
<tr>
<td>Helpfile</td>
<td>Help file that explains how to use the plug-in and provides additional information about the plug-in. It is recommended to place a How To section that explains how to use the plug-in. The file can be an ASCII text file, an HTML document, or PDF that contains all the full documentation for the plug-in or contains a set of redirects to where the content might be located online. Localized help files can be specified by combining Helpfile with a locale code. See Supported Locale Codes (page 100) for a full list of supported locale codes. NOTE All path specifiers are ‘/’ and not ‘\’, and paths are relative to the root .bundle folder.</td>
</tr>
<tr>
<td>ProductCode</td>
<td>Unique GUID for the plug-in. A GUID must be generated for each unique plug-in, and is used for first run notifications and as the installer ID for Add/Remove Programs in Windows when installed from the Apps tab on Autodesk Exchange. ProductCode should be updated if the AppVersion is changed. This is so upgrade installs work properly and a notification is displayed for the upgrade when loaded into AutoCAD. On Windows, you can use the MSI installer ProductCode or generate a GUID using an application such as</td>
</tr>
</tbody>
</table>
Description Attribute

GuidGen.exe. There are also websites that allow you to generate a GUID.

UpgradeCode

Unique GUID for the plug-in that must never be changed. The GUID is used by the Apps tab on Autodesk Exchange to allow for upgrading from an old version to a newer version of a plug-in without the need to uninstall the plug-in first.

**NOTE**

You must increment AppVersion in order to allow for proper upgrading of a plug-in.

An ApplicationPackage element can contain, or encapsulate, the following elements:
- **CompanyDetails** (page 93)
- **Components** (page 94)

**CompanyDetails Element**

The CompanyDetails element is used to specify information about the company that created the plug-in.

**NOTE**

The CompanyDetails element is required when releasing a plug-in through the Apps tab on Autodesk Exchange. You must also populate each of the attributes for the CompanyDetails element.

A CompanyDetails element can have any of the following attributes attached to it:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the developer or company that authored the plug-in.</td>
</tr>
</tbody>
</table>
### Components Element

The Components element is used to specify the components that make up one version of the plug-in.

More than one Components elements can be used to identify the components for a plug-in; each Components element can identify one or more components. Platform and product information for a Components element is defined by the RuntimeRequirements element (page 95).

For example, one Components element might define the 32-bit version of the plug-in, while another might define the 64-bit Windows or Mac OS X version.

Along with the RuntimeRequirements element, the Components element might contain one or more ComponentEntry elements. The ComponentEntry element (page 96) is used to define the individual files that need to be loaded by AutoCAD when the plug-in is ran.
RuntimeRequirements Element

The RuntimeRequirements element is recommended and is used to control which operating systems, platforms, releases, and languages the components can be used with.

It is also used to define AutoCAD support paths that are required for components in that Components section. You can specify as many sets of components as needed. If not included, it is assumed that all components are compatible with all AutoCAD and AutoCAD-based products, releases, and operating systems.

NOTE
Although this element is optional, it is possible that the plug-in might be installed on Mac OS X or another system that the plug-in was not originally tested on. Therefore, it is recommended that the element is used to control when the plug-in can be loaded.

A RuntimeRequirements element can have any of the following attributes attached to it:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Target operating system. Supported values are ‘Mac’, ‘Win32’, or ‘Win64’. If omitted, it is assumed the plug-in supports all operating systems. Multiple operating systems can be specified by separating the values with the ‘</td>
</tr>
</tbody>
</table>

NOTE
AutoLISP applications can be used across multiple operating systems.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Target AutoCAD or AutoCAD-based products. Should be used when using APIs specific to one of the AutoCAD-based products that might not available in AutoCAD or other AutoCAD-based products. Multiple AutoCAD platforms can be specified by separating the values with the ‘</td>
</tr>
</tbody>
</table>
The initial implementation of plug-ins cannot differentiate AutoCAD from an AutoCAD-based product, so the plug-in should support AutoCAD and all AutoCAD-based products.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeriesMin</td>
<td>Defines the minimum AutoCAD release number the set of components supports. The value can be a major version number (R18) or a specific version (R18.2). The AutoCAD version number can found in the Windows Registry or obtained with the ACADVER system variable. If this attribute and SeriesMax are not specified, it is assumed all components are compatible with all AutoCAD releases. If you omit this value, any version before that specified by the SeriesMax attribute is allowed.</td>
</tr>
<tr>
<td>SeriesMax</td>
<td>Defines the maximum AutoCAD release number the set of components supports. If you omit this value, any version after that specified by the SeriesMin attribute is allowed.</td>
</tr>
<tr>
<td>SupportPath</td>
<td>List of support paths used by this set of components separated by a semicolon. The support paths should be relative to the plug-in bundle. Localized support paths can be specified by combining SupportPath with a locale code. See Supported Locale Codes (page 100) for a full list of supported locale codes.</td>
</tr>
</tbody>
</table>

**ComponentEntry Element**

The ComponentEntry element is required and is used to specify details about each individual component in the Components element.

You can specify as many ComponentEntry elements as needed. Component types can be one of the following file formats:

- AutoLISP (LSP)
A ComponentEntry element may contain a **Commands element** (page 100) if the LoadReasons attribute is set to LoadOnCommandInvocation.

**NOTE**

ComponentEntry elements are loaded in the order they are listed, but from the bottom up. Therefore, any files that other components are dependent on must be lower down the list. For example, if an ObjectARX module is dependent on an ObjectDBX module, then the ObjectARX module will need to appear above the ObjectDBX module in the list.

A ComponentEntry element can have any of the following attributes attached to it:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppName</strong></td>
<td>Optional for AutoLISP; Required for ObjectARX - Component name; same as AppName in the ObjectARX API AcadAppInfo class.</td>
</tr>
<tr>
<td><strong>AppDescription</strong></td>
<td>Component description; same as AppDescription in the ObjectARX API AcadAppInfo class.</td>
</tr>
</tbody>
</table>
| **AppType**     | Component type; overrides the type derived from the file extension provided in the ModuleName attribute. The component type can be one of the following:  
- **Arx** – ObjectARX  
- **Dbx** – ObjectDBX  
- **Lisp** or **CompiledLisp** – AutoLISP  
- **Bundle** – Bundle package |
| **ModuleName**  | Relative path to component within the bundle; same as ModuleName in the ObjectARX API AcadAppInfo class. The component type is determined from the file extension:  
- **.arx** – ObjectARX |
It your application will handle multiple languages, different versions of a specific component to support different languages can be specified by combining ModuleName with a locale code. See Supported Locale Codes (page 100) for a full list of supported locale codes.

**NOTE** All path specifiers are ‘/’ and not ‘\’, and paths are relative to the root .bundle folder.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.dbx</td>
<td>.dbx – ObjectDBX</td>
</tr>
<tr>
<td>.lsp, .fas</td>
<td>.lsp, .fas, or – AutoLISP</td>
</tr>
</tbody>
</table>

AutoLISP only - When True, the AutoLISP file is loaded once per document. Default is True.

Multiple values can be specified - Defines the load behavior parameters for the component with LoadReasons and the exception of the LoadOnCommandInvocation parameter.

- LoadOnAutoCADStartup
- LoadOnAppearance
- LoadOnProxy

By default, LoadOnAutoCADStartup, LoadOnAppearance and LoadOnProxy are enabled (set to True) if LoadReasons is not specified. If parameters need to be disabled (set to False), the LoadReasons element must be specified along with the parameters set to False.

By default, LoadOnCommandInvocation is disabled, enabling it will disable LoadOnAutoCADStartup and LoadOnAppearance unless they are explicitly enabled.

See the ObjectARX Reference Guide for full details on AcadAppInfo LoadReasons.

Valid parameter values:

- **True** – Parameter is enabled
- **False** – Parameter is disabled

The following parameters are available:

- **LoadOnCommandInvocation**. Load only when a custom commands is invoked. When using this parameter, a ‘Commands’ element must be included. If LoadOnCommandInvocation is enabled, LoadOnAutoCADStartup and LoadOnAppearance are as-
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadOnAutoCADStartup - (Only applies to ObjectARX)</td>
<td>Load when AutoCAD for Mac starts up. When specified, this parameter has precedence over all other parameters. It is recommended only to use LoadOnAutoCADStartup when none of the other parameters are suitable, disable it (set it to False) whenever possible. If the LoadOnAutoCADStartup parameter is omitted, then it defaults to enabled (set to True) unless LoadOnCommandInvocation is enabled, in which case LoadOnAutoCADStartup defaults to False.</td>
</tr>
<tr>
<td>LoadOnProxyDetection</td>
<td>Load when a proxy for a custom entity is detected. By default, this parameter is enabled unless explicitly disabled (set to False). When enabled (set to True), LoadOnAutoCADStartup should be disabled. Only applies to ObjectDBX files.</td>
</tr>
<tr>
<td>LoadOnAppearance</td>
<td>Load when the product detects the application bundle in one of the ApplicationPlugins folders, thereby supporting instant load on installation with no need to restart AutoCAD. The parameter behaves the same way as LoadOnAutoCADStartup except the load context is relevant to when an application is installed while the product is running.</td>
</tr>
</tbody>
</table>

sumed to be disabled unless explicitly enabled. Only applies to ObjectARX.

NOTE
For AutoCAD startup performance reasons, it is very important to use this option when your components define commands.
Commands and Command Element

The Commands element is optional unless the LoadOnCommandInvocation parameter is enabled for the LoadReasons attribute. Used to specify which commands to register for LoadOnCommandInvocation.

You can specify more than one Command element as needed.

A Commands element can have the following attribute attached to it:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupName</td>
<td>Name used to organize related commands.</td>
</tr>
</tbody>
</table>

Command Element

Specifies the global and local names for each command.

A Command element can have any of the following attributes attached to it:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>Global command name.</td>
</tr>
<tr>
<td>Local</td>
<td>Local command name. Commands can be defined for multiple languages by combining Local with a locale code. See Supported Locale Codes (page 100) for a full list of supported locale codes.</td>
</tr>
</tbody>
</table>

Supported Locale Codes

AutoCAD for Mac is localized in a wide range of languages and the structure of the PackageContents.xml file supports these different languages with locale codes.

Many of the attributes in the PackageContents.xml file support localized languages. Append a locale code to the end of an attribute name to define a localized version of the attribute.
For example, to define a Spanish version of a description in the ApplicationPackage element you would create an attribute named DescriptionEsp.

The following is a full list of all supported locale codes:

- Chs - Chinese (PRC)
- Cht - Chinese (Taiwan)
- Csy - Czech
- Deu - German
- Enu - English
- Esp - Spanish
- Fra - French
- Hun - Hungarian
- Ita - Italian
- Jpn - Japanese
- Kor - Korean
- Plk - Polish
- Rus - Russian

**Example of Basic .bundle Folder Structure**

Basic example of what a package for a plug-in might contain and how it is structured on disk.

*.bundle* is not a file, but a folder name with a BUNDLE extension. The following is an example of a plug-in that contains a LSP file as the main program and a DWG support file. The following plug-in example is named OfficeSymbols and its folder structure might look something like; folders are in bold:

**OfficeSymbols.bundle**

- PackageContents.xml
- OfficeSymbolsUtilities.lsp
- OfficeSymbols.dwg
- OfficeSymbols.ico
### File name | Description
---|---
*OfficeSymbols.bundle* | The folder containing the files for a plug-in and has the BUNDLE extension.
*PackageContents.xml* | XML file that contains metadata about the plug-in.
*OfficeSymbolsUtilities.lsp* | Example of a custom application file that might define the behavior of the plug-in. An application file can be an AutoLISP, ObjectARX, or .NET assembly file.
*OfficeSymbols.dwg* | DWG file that contains symbols used by the functionality defined in *OfficeSymbolsUtilities.lsp*.
*OfficeSymbols.ico* | Icon used by the Apps tab on Autodesk Exchange.
*OfficeSymbols.htm* | Help documentation for the plug-in. Can be a redirect to where the documentation might be stored on the local drive or an online location.

### Definition of the PackageContents.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ApplicationPackage SchemaVersion="1.0" AppVersion="1.0"
    ProductCode="[Add Unique Plug-in GUID Here]"
    Name="Office Symbols"
    Icon="OfficeSymbols.ico"
    Helpfile="OfficeSymbols.htm"
>

<CompanyDetails
    Name="ABC Indoor CAD, Inc."
    Email="support@abcindoorcad.com"
>
Example of Using Folders to Organize Components

Example of what a package for a plug-in might look like using folders to organize components.

It is recommended to use an organized folder structure for larger applications as this can significantly speed up load times. The following is an example of a plug-in that contains multiple LSP files and resource files. The plug-in in this example is named OfficeSymbols and its folder structure might look something like:

OfficeSymbols.bundle
  |- PackageContents.xml
  |- Contents
    |- OfficeSymbolsMain.lsp
    |- OfficeSymbolsUtilities.lsp
  |- Resources
    |- OfficeSymbols.dwg
    |- OfficeSymbols.cuix
    |- OfficeSymbols.ico
  |- Help
    |- OfficeSymbols.htm
Definition of the PackageContents.xml

<?xml version="1.0" encoding="utf-8" ?>
<ApplicationPackage
  SchemaVersion="1.0"
  AppVersion="1.0"
  Author="ABC Indoor CAD, Inc."
  ProductCode="[Add Unique Plug-in GUID Here]"
  Name="Office Symbols (contains Full version)"
  Icon="./Contents/Resources/OfficeSymbols.ico"
  Helpfile="/Contents/Help/OfficeSymbols.htm"
>
  <CompanyDetails
    Name="ABC Indoor CAD, Inc."
    Phone="1 (555)-415-1234"
    PhoneEsp="34 5554 151234"
    Url="www.abcindoorcad.com"
    UrlEsp="www.abcindoorcad.es"
    Email="support@abcindoorcad.com"
  />

  <Components>
    <RuntimeRequirements SupportPath="/Contents/Support"/>
    <ComponentEntry
      AppName="MainLISP"
      ModuleName="/Contents/OfficeSymbolsMain.lsp"
      PerDocument="True"
    />
    <ComponentEntry
      AppName="UtilitiesLISP"
      ModuleName="/Contents/OfficeSymbolsUtilities.lsp"
      PerDocument="True"
    />
    <ComponentEntry
      ModuleName="/Contents/Resources/OfficeSymbols.cuix"
    />
  </Components>
</ApplicationPackage>
Overview of Shape Files

Shapes are objects that you use like blocks. First you use the LOAD command to load the compiled shape file containing the shape definition. Then you use the SHAPE command to insert shapes from the file into your drawing. You can specify the scale and rotation to use for each shape as you add it. AutoCAD for Mac SHP fonts are a special type of shape file, and are defined in the same way as shape files.

Blocks are more versatile and easier to use and apply than shapes. However, shapes are more efficient for AutoCAD for Mac to store and draw. User-defined shapes are helpful when you must insert a simple part many times and when speed is important.

Compile Shape/Font Files

You enter the description of shapes in a specially formatted text file with a file extension of .shp. To create the file, use a text editor or word processor that enables you to save in ASCII format, and then compile the ASCII file. Compiling a shape definition file (SHP) generates a compiled shape file (SHX).

The compiled file has the same name as the shape definition file but with a file type of SHX. If the shape definition file defines a font, you use the STYLE command to define a text style. Then, you use one of the text placement commands (TEXT or MTEXT) to place the characters in the drawing. If the shape definition file defines shapes, you use the LOAD command to load the shape file into the drawing. Then, you use the SHAPE command to place the individual shapes in the drawing (similar in concept to the INSERT command).
Compile PostScript Fonts

To use a Type 1 PostScript font in AutoCAD for Mac, you must first compile it into an AutoCAD for Mac shape file. The COMPILE command accepts both SHP and PFB files as input and generates an SHX file. Compiled versions of PostScript fonts can take a lot of disk space, so compile only those fonts you use frequently.

AutoCAD for Mac cannot compile and load every Type 1 font. The PostScript font facilities in AutoCAD for Mac are intended to process a subset of Adobe fonts. If you receive an error while compiling a PostScript font, the resulting SHX file (if one is generated) may not load into AutoCAD for Mac.

For more information on the Adobe Type 1 font format, refer to Adobe Type1 Font Format Version 1.1. When you’ve purchased and installed these fonts, you can begin using them with AutoCAD for Mac.

 NOTE Make sure you understand any copyright that accompanies the PostScript fonts you use. The same copyright restrictions generally apply to the SHX form of fonts you’ve compiled.

Overview of Shape Files

To compile a shape or font file

- At the Command prompt, enter compile and press Enter.

In the Select Shape or Font File dialog box, you can select a shape definition file (SHP) or PostScript font file (PFB). After you select the file name, compiling begins. If AutoCAD for Mac finds an error in the shape descriptions, a message is displayed telling you the type of error and the line number. When compiling is complete, the following messages are displayed:

Compilation successful.
Output file name.shx contains nnn bytes.

Create Shape Definition Files

AutoCAD for Mac font and shape files (SHX) are compiled from shape definition files (SHP). You can create or modify shape definition files with a text editor or word processor that saves files in ASCII format.
Shape Descriptions

AutoCAD for Mac font and shape files (SHX) are compiled from shape definition files (SHP). You can create or modify shape definition files with a text editor or word processor that saves files in ASCII format.

The syntax of the shape description for each shape or character is the same regardless of the final use (shape or font) for that shape description. If a shape definition file is to be used as a font file, the first entry in the file describes the font itself rather than a shape within the file. If this initial entry describes a shape, the file is used as a shape file.

Being able to create your own shape definitions is a valuable skill. Keep in mind, however, that this is a very complex subject to learn and requires patience.

Each line in a shape definition file can contain up to 128 characters. Longer lines cannot be compiled. Because AutoCAD for Mac ignores blank lines and text to the right of a semicolon, you can embed comments in shape definition files.

Each shape description has a header line of the following form and is followed by one or more lines containing specification bytes, separated by commas and terminated by a 0.

```
*shapenumber,defbytes,shapename

specbyte1,specbyte2,specbyte3,...,0
```

The following list describes the fields of a shape description:

**shapenumber** A number, unique to the file, between 1 and 258 (and up to 32768 for Unicode fonts), and preceded by an asterisk (*). Non-Unicode font files use the shape numbers 256, 257, and 258 for the symbolic identifiers Degree_Sign, Plus_or_Minus_Sign, and Diameter_Symbol. For Unicode fonts these glyphs appear at the U+00B0, U+00B1, and U+2205 shape numbers and are part of the “Latin Extended-A” subset.

Text fonts (files containing shape definitions for each character) require specific numbers corresponding to the value of each character in the ASCII code; other shapes can be assigned any numbers.

**defbytes** The number of data bytes (specbytes) required to describe the shape, including the terminating 0. The limit is 2,000 bytes per shape.

**shapenname** The shape name. Shape names must be uppercase to be recognized. Names with lowercase characters are ignored and are usually used to label font shape definitions.
specbyte A shape specification byte. Each specification byte is a code that defines either a vector length and direction or one of a number of special codes. A specification byte can be expressed in the shape definition file as either a decimal or hexadecimal value. This section uses both decimal and hexadecimal specification byte values for its examples (as do many of the shape definition files). If the first character of a specification byte is a 0 (zero), the two characters that follow are interpreted as hexadecimal values.

**Vector Length and Direction Code**

A simple shape specification byte contains vector length and direction encoded into one specification byte.

A simple shape specification byte contains vector length and direction encoded into one specification byte (one `specbyte` field). Each vector length and direction code is a string of three characters. The first character must be a 0, which indicates to AutoCAD for Mac that the next two characters are interpreted as hexadecimal values. The second character specifies the length of the vector in units. Valid hexadecimal values range from 1 (one unit long) through F (15 units long). The third character specifies the direction of the vector. The following figure illustrates the direction codes.

![Vector direction codes](image)

Vector direction codes

All the vectors in the preceding figure were drawn with the same length specification. Diagonal vectors stretch to match the X or Y displacement of
the closest orthogonal vector. This is similar to the action of the snap grid in AutoCAD for Mac.

The following example constructs a shape named DBOX with an arbitrarily assigned shape number of 230.

*230,6,DBOX
014,010,01C,018,012,0

The preceding sequence of specification bytes defines a box one unit high by one unit wide, with a diagonal line running from the lower left to the upper right. After saving the file as dbox.shp, use the COMPILE command to generate the dbox.shx file. Use the LOAD command to load the shape file containing this definition, and then use the SHAPE command as follows:

Command: shape
Enter shape name or [?] : dbox
Specify insertion point: 1,1
Specify height <current> : 2
Specify rotation angle <current> : 0

The resulting shape is shown in the following illustration.

Special Codes

Special codes can be used to create additional geometric forms and specify certain actions.
Use Special Codes

The second character of the three-character string (the vector length specification) must be 0, or you can specify the special code number. For example, 008 and 8 are both valid specifications.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>End of shape definition</td>
</tr>
<tr>
<td>001</td>
<td>Activate Draw mode (pen down)</td>
</tr>
<tr>
<td>002</td>
<td>Deactivate Draw mode (pen up)</td>
</tr>
<tr>
<td>003</td>
<td>Divide vector lengths by next byte</td>
</tr>
<tr>
<td>004</td>
<td>Multiply vector lengths by next byte</td>
</tr>
<tr>
<td>005</td>
<td>Push current location onto stack</td>
</tr>
<tr>
<td>006</td>
<td>Pop current location from stack</td>
</tr>
<tr>
<td>007</td>
<td>Draw subshape number given by next byte</td>
</tr>
<tr>
<td>008</td>
<td>X-Y displacement given by next two bytes</td>
</tr>
<tr>
<td>009</td>
<td>Multiple X-Y displacements, terminated (0,0)</td>
</tr>
<tr>
<td>00A</td>
<td>Octant arc defined by next two bytes</td>
</tr>
<tr>
<td>00B</td>
<td>Fractional arc defined by next five bytes</td>
</tr>
<tr>
<td>00C</td>
<td>Arc defined by X-Y displacement and bulge</td>
</tr>
<tr>
<td>00D</td>
<td>Multiple bulge-specified arcs</td>
</tr>
</tbody>
</table>
Specification byte codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00E</td>
<td>Process next command only if vertical text</td>
</tr>
</tbody>
</table>

**Codes 0, 1, and 2: End of Shape and Draw Mode Control**

Code 0 marks the end of the shape definition, and Codes 1 and 2 control Draw mode.

Draw is activated at the start of each shape. When Draw mode is turned on (code 1), the vectors cause lines to be drawn. When Draw mode is turned off (code 2), the vectors move to a new location without drawing.

**Codes 3 and 4: Size Control**

Codes 3 and 4 control the relative size of each vector.

The height specified with the SHAPE command is initially considered the length of a single orthogonal vector (direction 0, 4, 8, or C). Code 3 divides vector lengths by the next byte. Code 4 multiplies vector lengths by the next byte. Codes 3 and 4 are followed by a specification byte containing an integer scale factor (1 through 255).

If you want the shape height to specify the size of the entire shape, and you use 10 vector lengths to draw it, you can use 3,10 to scale the height specification. The scale factor is cumulative within a shape; that is, multiplying by 2 and again by 6 results in a scale factor of 12. Usually, you should reverse the effect of your scale factors at the end of the shape, especially for subshapes and text font shapes. AutoCAD for Mac does not reset the scale factor for you.

**Codes 5 and 6: Location Save/Restore**

Code 5 pushes (saves) and code 6 pops (restores) the current coordinate position while drawing a shape so that you can return to it from a later point in the shape.
You must pop everything you push. The position stack is only four locations deep. If the stack overflows because of too many pushes or too many missing pops, the following message is displayed when the shape is drawn.

Position stack overflow in shape mnn

Similarly, if you try to pop more locations than have been pushed onto the stack, the following message is displayed when the shape is drawn.

Position stack underflow in shape mnn

**Code 7: Subshape**

Code 7 draws the subshape number given by the next byte.

For a non-Unicode font the specification byte following code 7 is a shape number from 1 to 255. For a Unicode font, code 7 is followed by a Unicode shape number from 1 to 65535. Unicode shape numbers should be counted as two bytes (for specific information about the differences between Unicode and non-Unicode fonts, see Unicode Font Descriptions (page 176)).

The shape with that number (in the same shape file) is drawn at this time. Draw mode is not reset for the new shape. When the subshape is complete, drawing the current shape resumes.

**Codes 8 and 9: X-Y Displacements**

With codes 8 and 9 you can draw nonstandard vectors using X-Y displacements.

Normal vector specification bytes draw only in the 16 predefined directions, and the longest length is 15. These restrictions help make shape definitions efficient but are sometimes limiting. Code 8 specifies the X-Y displacement given by the next two bytes. Code 8 must be followed by two specification bytes in the format:

8, X-displacement, Y-displacement

The X-Y displacements can range from -128 to +127. A leading + is optional, and you can use parentheses to improve readability. The following example results in a vector that draws (or moves) 10 units to the left and three units up.

8, (-10, 3)
Following the two displacement specification bytes, the shape returns to Normal Vector mode.

You can use code 9 to draw a sequence of nonstandard vectors. Code 9 specifies any number of X-Y displacement pairs. The code sequence is terminated by a (0,0) pair. The following example draws three nonstandard vectors and returns to Normal Vector mode.

9, (3,1), (3,2), (2,-3), (0,0)

You must terminate the sequence of X-Y displacement pairs with a (0,0) pair in order for AutoCAD for Mac to recognize any Normal Vectors or special codes that follow.

**Code 00A: Octant Arc**

Special code 00A (or 10) uses the next two specification bytes to define an arc.

This is called an *octant arc* because it spans one or more 45-degree *octants*, starting and ending on an octant boundary. Octants are numbered counterclockwise from the 3 o’clock position, as shown in the following illustration.

![Octant Diagram]

The arc specification is

10, radius, (-)0sC

The radius can be any value from 1 through 255. The second specification byte indicates the direction of the arc (counterclockwise if positive, and clockwise if negative), its starting octant (s, a value from 0 through 7), and the number of octants it spans (c, a value from 0 through 7, in which 0 equals eight octants, or a full circle). You can use parentheses to improve readability. For example, consider the following fragment of a shape definition:

...012, 10, (1, -032), 01E,...
This code draws a one-unit vector up and to the right, a clockwise arc from octant 3 (with a radius of one unit for two octants), and then a one-unit vector down and to the right, as shown in the following illustration.

**Code 00B: Fractional Arc**

Special code 00B (11) draws an arc that doesn’t necessarily start and end on an octant boundary. The definition uses five specification bytes.

11, start_offset, end_offset, high_radius, radius, (-)0SC

The `start_offset` and `end_offset` represent how far from an octant boundary the arc begins or ends. The `high_radius` represents the most significant eight bits of the radius; the high radius will be 0 unless the `radius` is greater than 255 units. Multiply the `high_radius` value by 256 and add that value to the `radius` value to generate an arc radius greater than 255. The `radius` and ending specification byte are the same as for the octant arc specification (code 00A, described previously).

You determine the `start_offset` by calculating the difference in degrees between the starting octant’s boundary (a multiple of 45 degrees) and the start of the arc. Then, you multiply this difference by 256 and divide by 45. If the arc starts on an octant boundary, its `start_offset` is 0.

The `end_offset` is calculated in a similar fashion, but you use the number of degrees from the last octant boundary crossed to the end of the arc. If the arc ends on an octant boundary, its `end_offset` is 0.

For example, a fractional arc from 55 degrees to 95 degrees with a 3 unit radius would be coded as follows:

11, (56, 28, 0, 3, 012)

Here is the explanation:
start_offset = 56 because \((55 - 45) \times 256 / 45\) = 56
end_offset = 28 because \((95 - 90) \times 256 / 45\) = 28
high_radius = 0 because \(radius < 255\)
radius = 3
starting octant = 1 because arc starts in the 45 degree octant
ending octant = 2 because arc ends in the 90 degree octant

**Codes 00C and 00D: Bulge-Specified Arcs**

Special codes 00C and 00D (12 and 13) provide another mechanism for including arc segments in shape descriptions.

They are similar to codes 8 and 9 in that you can use them to specify X-Y displacements. However, codes 00C and 00D draw arcs by applying a bulge factor to the displacement vector. Code 00C draws one arc segment, while code 00D draws multiple arc segments (polyarcs) until it is terminated by a (0,0) displacement.

Code 00C must be followed by three bytes describing the arc:

\[ \text{0C, X-displacement, Y-displacement, Bulge} \]

Both the X and Y displacement and the bulge, which specifies the curvature of the arc, can range from -127 to +127. If the line segment specified by the displacement has length \(D\), and the perpendicular distance from the midpoint of that segment has height \(H\), the magnitude of the bulge is \((2 \times H / D) \times 127\). The sign is negative if the arc from the current location to the new location is clockwise.

![Diagram showing the bulge factor](image)

A semicircle has bulge 127 (or -127) and is the greatest arc that can be represented as a single-arc segment using these codes (use two consecutive arc segments for larger arcs). A bulge specification of 0 is valid and represents a straight-line segment. Note, however, that using code 8 for a straight-line segment saves a byte in the shape description.
The polyarc code (00D, or 13) is followed by 0 or by more arc segment triples, and is terminated by a (0,0) displacement. Note that no bulge is specified after the final displacement. For example, the letter S might be defined by the following sequence:

\[ 13, (0,5,127), (0,5,-127), (0,0) \]

Zero bulge segments are useful within polyarcs to represent straight segments; they are more efficient than terminating the polyarc, inserting one straight segment, and then starting another polyarc.

The number -128 cannot be used in arc segment and polyarc definitions.

**Code 00E: Flag Vertical Text Command**

Special code 00E (14) is used only in dual-orientation text font descriptions, where the font is used in both horizontal and vertical orientations.

When this special code is encountered in a character definition, the next code is either processed or skipped, depending on orientation. If the orientation is vertical, the next code is processed; if it is horizontal, the next code is skipped.

In horizontal text, the start point for each character is the left end of the baseline. In vertical text, the start point is assumed to be the top center of the character. At the end of each character, a pen-up segment is normally drawn to position to the next character's start point. For horizontal text, it is to the right; for vertical text, it is downward. The special 00E (14) code is used primarily to adjust for differences in start points and endpoints, permitting the same character shape definition to be used both horizontally and vertically. For instance, the following definition of an uppercase D could be used in either horizontal or vertical text.

\[ \ast 68,22,ucd \\
2,14,8,(-2, 6),1,030,012,044,016,038,2,010,1,06C,2,050,14,8,(-4,-3),0 \]
Text Font Descriptions

Text fonts must include a special shape number 0 that conveys information about the font itself.

AutoCAD for Mac is packaged with numerous text fonts. You can use the STYLE command to apply expansion, compression, or obliquing to any of these fonts, thereby tailoring the characters to your needs. You can draw text of any height, at any baseline angle, and with either horizontal or vertical orientation using these fonts.

AutoCAD for Mac text fonts are files of shape definitions with shape numbers corresponding to the ASCII code for each character. Codes 1 through 31 are for control characters, only one of which is used in AutoCAD for Mac text fonts:

10 (LF) The line feed (LF) must drop down one line without drawing. This is used for repeated TEXT commands, to place succeeding lines below the first one.

*10,5,lf
2,8,(0,-10),0

You can modify the spacing of lines by adjusting the downward movement specified by the LF shape definition.

Text fonts must include a special shape number 0 that conveys information about the font itself. The format has the following syntax:

*0,4,font-name
above,below,modes,0
The above value specifies the number of vector lengths above the baseline that the uppercase letters extend, and below indicates how far the lowercase letters descend below the baseline. The baseline is similar in concept to the lines on writing paper. These values define the basic character size and are used as scale factors for the height specified in the TEXT command.

The modes byte should be 0 for a horizontally oriented font and 2 for a dual-orientation (horizontal or vertical) font. The special 00E (14) command code is honored only when modes is set to 2.

The standard fonts supplied with AutoCAD for Mac include a few additional characters required for the AutoCAD for Mac dimensioning feature.

%%d Degree symbol (°)
%%p Plus/minus tolerance symbol ( )
%%c Circle diameter dimensioning symbol

You can use these and the %%nnn control sequences, as described under TEXT in the Command Reference.

**NOTE** AutoCAD for Mac draws text characters by their ASCII codes (shape numbers) and not by name. To save memory, specify the shape name portion of each text shape definition in lowercase as shown in the following example. (Lowercase names are not saved in memory.)

```
*65,11,uca
 024,043,04d,02c,2,047,1,040,2,02e,0
```

Because the shape name uca contains lowercase letters, AutoCAD for Mac does not save the name in memory. However, you can use the name for reference when editing the font definition file. In this example, uca stands for uppercase A.

## Sample Files

This topic contains sample files that help extend the font characters provided with AutoCAD for Mac.

### Extended Simplex Roman

```
;;
;; romans.shp - Extended Simplex Roman
```
2, 8, (17, -3), 14, 6, 14, 8, (10, -13), 0

*00025, 64, kpc
2, 14, 8, (-12, -21), 14, 5, 8, (21, 21), 1, 8, (-18, -21), 2, 8, (5, 21), 1, 02E, 02C, 02B, 029, 028, 026, 024, 023, 021, 020, 02F, 8, (3, -1), 030, 8, (3, 1), 021, 2, 8, (-4, -14), 1, 029, 02B, 02C, 02E, 020, 021, 023, 024, 026, 028, 2, 8, (7, -7), 14, 6, 14, 8, (12, -9), 0

*00026, 67, kand
2, 14, 8, (-13, -21), 14, 5, 8, (23, 12), 1, 014, 016, 018, 01A, 02B, 8, (-2, -5), 8, (-2, -3), 02A, 029, 048, 027, 016, 025, 024, 023, 012, 8, (7, 4), 012, 023, 024, 025, 027, 029, 02B, 02C, 8, (1, -3), 8, (2, -3), 8, (5, -7), 02E, 02F, 020, 012, 014, 2, 8, (3, -2), 14, 6, 14, 8, (13, -9), 0

*00027, 29, kapos
2, 14, 8, (-5, -25), 14, 5, 8, (6, 24), 1, 01A, 016, 012, 01E, 02C, 02B, 01A, 2, 8, (6, -19), 14, 6, 14, 8, (5, -9), 0

*00028, 39, klp
2, 14, 8, (-7, -25), 14, 5, 8, (11, 25), 1, 02A, 8, (-2, -3), 04B, 8, (-1, -5), 04C, 8, (1, -5), 04D, 8, (2, -3), 02E, 2, 8, (3, 7), 14, 6, 14, 8, (7, -16), 0

*00029, 39, krp
2, 14, 8, (-7, -25), 14, 5, 8, (3, 25), 1, 02E, 8, (2, -3), 04D, 8, (1, -5), 04C, 8, (-1, -5), 04B, 8, (-2, -3), 02A, 2, 8, (11, 7), 14, 6, 14, 8, (7, -16), 0

*0002A, 37, kas
2, 14, 8, (-8, -21), 14, 5, 8, (8, 21), 1, 0CC, 2, 8, (-5, 9), 1, 8, (10, -6), 2, 064, 1, 8, (-10, -6), 2, 8, (13, -12), 14, 6, 14, 8, (8, -9), 0

*0002B, 31, kplis
2, 14, 8, (-13, -18), 14, 5, 8, (13, 18), 1, 8, (0, -18), 2, 096, 1, 8, (18, 0), 2, 8, (4, -9), 14, 6, 14, 8, (13, -9), 0

*0002C, 29, kcmca
2, 14, 8, (-5, -2), 14, 5, 8, (6, 1), 1, 01A, 016, 012, 01E, 02C, 02B, 01A, 2, 8, (6, 4), 14, 6, 14, 8, (5, -13), 0

*0002D, 25, ksub
2, 14, 8, (-13, -9), 14, 5, 8, (4, 9), 1, 8, (18, 0), 2, 8, (4, -9), 14, 6, 14, 8, (13, -9), 0

*0002E, 26, kper
2, 14, 8, (-5, -2), 14, 5, 8, (5, 2), 1, 01A, 01E, 012, 016, 02B, 2, 8, (5, -9), 0

*0002F, 25, kdiv
2, 14, 8, (-11, -25), 14, 5, 8, (20, 25), 1, 8, (-18, -32), 2, 8, (20, 7), 14, 6, 14, 8, (11, -16), 0

*00030, 62, n0
2, 14, 8, (-10, -21), 14, 5, 8, (9, 21), 1, 8, (-3, -1), 8, (-2, -3), 8, (-1, -5),
2, 14, 8, (-5, -14), 14, 5, 8, (5, 14), 1, 01A, 01E, 012, 016, 2, 8, (1, -13), 1, 01A, 016, 012, 01E, 02C, 02B, 01A, 2, 8, (6, 4), 14, 6, 14, 8, (5, -13), 0

*0003C, 28, klt
2, 14, 8, (-12, -18), 14, 5, 8, (20, 18), 1, 8, (-16, -9), 8, (16, -9), 2, 8, (4, 0), 14, 6, 14, 8, (12, -9), 0

*0003D, 33, keq
2, 14, 8, (-13, -12), 14, 5, 8, (4, 12), 1, 8, (18, 0), 2, 8, (-18, -6), 1, 8, (18, 0), 2, 8, (4, -6), 14, 6, 14, 8, (13, -9), 0

*0003E, 28, kgt
2, 14, 8, (-12, -18), 14, 5, 8, (4, 18), 1, 8, (16, -9), 2, 8, (20, 0), 14, 6, 14, 8, (12, -9), 0

*0003F, 42, kqm
2, 14, 8, (-9, -21), 14, 5, 8, (3, 16), 1, 014, 023, 012, 021, 040, 02F, 01E, 02D, 02C, 02B, 01A, 049, 03C, 2, 05C, 1, 01A, 01E, 012, 016, 2, 8, (9, -2), 14, 6, 14, 8, (9, -9), 0

*00040, 93, kea
2, 14, 3, 2, 14, 8, (-27, -42), 14, 4, 2, 14, 5, 8, (18, 13), 1, 025, 027, 038, 029, 01A, 02B, 03C, 02D, 01E, 02F, 030, 021, 023, 2, 084, 1, 0AC, 01E, 020, 022, 8, (1, 3), 024, 8, (-1, 3), 025, 026, 027, 8, (-3, 1), 038, 8, (-3, -1), 029, 02A, 02B, 8, (-1, -3), 03C, 8, (1, -3), 02D, 02E, 02F, 8, (3, -1), 030, 8, (3, 1), 021, 012, 2, 8, (6, -3), 14, 6, 14, 3, 2, 14, 8, (27, -18), 14, 6, 4, 2, 0

*00041, 39, uca
2, 14, 8, (-9, -21), 14, 5, 8, (9, 21), 1, 8, (-8, -21), 2, 8, (8, 21), 1, 8, (8, -21), 2, 8, (-13, 7), 1, 0A0, 2, 8, (4, -7), 14, 6, 14, 8, (9, -9), 0

*00042, 70, ucb
2, 14, 3, 2, 14, 8, (-21, -42), 14, 4, 2, 14, 5, 8, (4, 21), 1, 8, (0, -21), 2, 8, (0, 21), 1, 090, 8, (3, -1), 01E, 02D, 02C, 02B, 01A, 8, (-3, -1), 2, 098, 1, 090, 8, (3, -1), 01E, 02D, 03C, 02B, 01A, 8, (-3, -1), 098, 2, 8, (17, 0), 14, 6, 14, 3, 2, 14, 8, (21, -18), 14, 4, 2, 0

*00043, 55, ucc
2, 14, 3, 2, 14, 8, (-21, -42), 14, 4, 2, 14, 5, 8, (18, 16), 1, 025, 026, 027, 048, 029, 02A, 02B, 8, (-1, -3), 05C, 8, (1, -3), 02D, 02E, 02F, 040, 021, 022, 023, 2, 8, (3, -5), 14, 6, 14, 3, 2, 14, 8, (21, -18), 14, 4, 2, 0

*00044, 61, ucd
2, 14, 3, 2, 14, 8, (-21, -42), 14, 4, 2, 14, 5, 8, (4, 21), 1, 8, (0, -21), 2, 8, (0, 21), 1, 070, 8, (3, -1), 02E, 02D, 8, (1, -3), 05C, 8, (-1, -3), 02B, 02A, 8, (-3, -1), 078, 2, 8, (17, 0), 14, 6, 14, 3, 2, 14, 8, (21, -18), 14, 4, 2, 0

*00045, 55, uce
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (4, 21), 1, 8, (0, -21), 2,
2, 14, 3, 2, 14, 8, (-21, -42), 1, 8, (0, -21), 2, 8, (0, 21), 1, 090, 8, (3, -1), 01E, 02D, 03C, 02B, 01A, 8, (-3, -1), 098, 2, 8, (17, -10), 14, 6, 14, 3, 2, 14, 8, (21, -18), 14, 4, 2, 0
2, 14, 8, (-11, -21), 1, 029, 02A, 02B, 01A, 8, (-3, -1), 098, 2, 070, 1, 8, (3, 0), 14, 6, 14, 3, 2, 14, 8, (21, -18), 14, 4, 2, 0
2, 14, 8, (-10, -21), 14, 5, 8, (17, 18), 1, 026, 8, (-3, 1), 048, 8, (-3, -1), 048, 8, (8, 21), 1, 0E0, 2, 8, (1, -21), 14, 6, 14, 8, (10, -9), 0
2, 14, 8, (-8, -21), 14, 5, 8, (8, 21), 1, 8, (0, -21), 2, 8, (-7, 21), 1, 0E0, 2, 8, (1, -21), 14, 6, 14, 8, (8, -9), 0
2, 14, 8, (-11, -21), 14, 5, 8, (4, 21), 1, 0E0, 8, (1, -3), 02E, 8, (3, -1), 020, 8, (3, 1), 022, 8, (1, 3), 0F4, 2, 8, (4, -21), 14, 6, 14, 8, (11, -9), 0
2, 14, 8, (-9, -21), 14, 5, 8, (1, 21), 1, 8, (8, -21), 2, 8, (8, 21), 1, 8, (-8, -21), 2, 8, (9, 0), 14, 6, 14, 8, (9, -9), 0
2, 14, 8, (-12, -21), 14, 5, 8, (2, 21), 1, 8, (5, -21), 2, 8, (5, 21), 1, 8, (-5, -21), 2, 8, (5, 21), 1, 8, (-5, -21), 2, 8, (7, 0), 14, 6, 14, 8, (12, -9), 0
2, 14, 8, (-10, -21), 14, 5, 8, (3, 21), 1, 8, (14, -21), 2, 8, (0, 21), 1, 8, (-14, -21), 2, 8, (17, 0), 14, 6, 14, 8, (10, -9), 0
2, 14, 8, (-9, -21), 14, 5, 8, (1, 21), 1, 8, (8, -10), 0BC, 2, 8, (8, 21), 1, 8, (-8, -10), 2, 8, (9, -11), 14, 6, 14, 8, (9, -9), 0
2, 14, 8, (-10, -21), 14, 5, 8, (17, 21), 1, 8, (-14, -21), 2, 8, (0, 21), 1, 0E0, 2, 8, (-14, -21), 1, 0E0, 2, 8, (3, 0), 14, 6, 14, 8, (10, -9), 0
2, 14, 8, (-7, -25), 14, 5, 8, (4, 25), 1, 8, (0, -32), 2, 8, (0, 32), 1, 070, 2, 8, (-7, -32), 1, 070, 2, 8, (3, 7), 14, 6, 14, 8, (7, -16), 0

124 | Chapter 8 Shapes and Shape Fonts
Create Shape Definition Files | 125
Chapter 8  Shapes and Shape Fonts
Chapter 8  Shapes and Shape Fonts
130 | Chapter 8  Shapes and Shape Fonts
2, 8, (-6, -21), 1, 8, (16, 21), 2, 8, (3, -21), 14, 8, (-11, -9), 0
*000D9, 43, uc^*
2, 14, 8, (-11, -25), 14, 5, 8, (15, 21), 1, 087, 2, 06B, 1, 0DC, 8, (1, -3), 02E, 8, (3, -1), 020, 8, (3, 1), 022, 8, (1, 3), 0D4, 2, 8, (4, -19), 14, 6, 14, 8, (11, -9), 0
*000DA, 45, uc^*
2, 14, 8, (-11, -25), 14, 5, 8, (15, 25), 1, 089, 2, 8, (-3, -2), 1, 0DC, 8, (1, -3), 02E, 8, (3, -1), 020, 8, (3, 1), 022, 8, (1, 3), 0D4, 2, 8, (4, -19), 14, 6, 14, 8, (11, -9), 0
*000DB, 46, uc^*
2, 14, 8, (-11, -25), 14, 5, 8, (15, 23), 1, 047, 049, 2, 8, (-3, -2), 1, 0FC, 8, (1, -3), 02E, 8, (3, -1), 020, 8, (3, 1), 022, 8, (1, 3), 0F4, 2, 8, (4, -21), 14, 6, 14, 8, (11, -9), 0
*000DC, 55, uc^*
2, 14, 8, (-11, -25), 14, 5, 8, (14, 24), 1, 01E, 012, 016, 01A, 2, 088, 1, 01E, 012, 016, 01A, 2, 8, (-2, -3), 1, 0FC, 8, (1, -3), 02E, 8, (3, -1), 020, 8, (3, 1), 022, 8, (1, 3), 0F4, 2, 8, (4, -21), 14, 6, 14, 8, (11, -9), 0
*000DD, 38, uc^*
2, 14, 8, (-9, -25), 14, 5, 8, (13, 25), 1, 089, 2, 049, 1, 8, (8, -9), 0AC, 2, 8, (8, 19), 1, 8, (-8, -9), 2, 8, (9, -10), 14, 6, 14, 8, (9, -9), 0
*000DF, 53, kgers
2, 14, 8, (-9, -21), 14, 5, 030, 1, 012, 8, (0, 16), 023, 012, 021, 020, 02F, 01E, 02D, 02C, 02B, 01A, 029, 028, 2, 020, 8, (3, -1), 01E, 02D, 03C, 02B, 01A, 029, 028, 027, 016, 012, 01E, 2, 8, (10, -2), 14, 6, 14, 8, (9, -9), 0
*000E0, 63, lc...
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (5, 21), 1, 8, (8, -4), 2, 8, (2, -3), 1, 0EC, 2, 0B4, 1, 026, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (4, -3), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000E1, 63, lc
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (5, 17), 1, 8, (8, 4), 2, 8, (2, -7), 1, 0EC, 2, 0B4, 1, 026, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (4, -3), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000E2, 64, lc^*
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (5, 18), 1, 8, (4, 3), 8, (4, -3), 2, 04D, 1, 0EC, 2, 0B4, 1, 026, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (4, -3), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000E3, 63, lcf
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (4, 18), 1, 032, 010, 03E, 010,
032, 2, 07C, 1, 0EC, 2, 0B4, 1, 026, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (4, -3), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000E4, 71, 1c+
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (4, 20), 1, 01E, 012, 016, 01A, 2, 090, 1, 01E, 012, 016, 01A, 2, 8, (2, -6), 1, 0EC, 2, 0B4, 1, 026, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (4, -3), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000E5, 63, 1c†
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (7, 19), 1, 10, (2, 64), 2, 8, (8, -5), 1, 0EC, 2, 0B4, 1, 026, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (4, -3), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000E6, 51, 1c
2, 14, 3, 2, 14, 8, (10, 64), 1, 070, 014, 8, (10, 8), 1, 070, 014, 8, (-1, 3), 026, 028, 02A, 026, 028, 02A, 8, (-1, -3), 04C, 8, (1, -3), 02E, 020, 022, 02E, 020, 021, 023, 2, 8, (-7, 11), 1, 0EC, 2, 0A0, 14, 6, 14, 8, (10, -9), 0
*000E7, 49, 1c†
2, 14, 8, (-9, -14), 14, 5, 8, (15, 11), 1, 026, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (-8, -9), 1, 01E, 030, 012, 024, 016, 028, 034, 2, 090, 14, 6, 14, 8, (9, -16), 0
*000E8, 48, 1c§
2, 14, 8, (-9, -21), 14, 5, 8, (5, 21), 1, 08F, 2, 8, (-10, -9), 1, 0C0, 024, 025, 016, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (3, -3), 14, 6, 14, 8, (9, -9), 0
*000E9, 48, 1c
2, 14, 8, (-9, -21), 14, 5, 8, (5, 17), 1, 081, 2, 8, (-10, -13), 1, 0CD, 024, 025, 016, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (3, -3), 14, 6, 14, 8, (9, -9), 0
*000EA, 51, 1c−
2, 14, 8, (-9, -21), 14, 5, 8, (5, 18), 1, 8, (4, 3), 8, (4, -3), 2, 0AA, 1, 0C0, 024, 025, 016, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (3, -3), 14, 6, 14, 8, (9, -9), 0
*000EB, 58, 1c%
2, 14, 8, (-9, -21), 14, 5, 8, (4, 20), 1, 01E, 012, 016, 01A, 2, 080, 1, 01E, 012, 016, 01A, 2, 8, -12, 1, 0C0, 024, 025, 016, 027, 038, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 2, 8, (3, -3), 14, 6, 14, 8, (9, -9), 0
*000EC, 27, 1c−
2, 14, 8, (-7, -21), 14, 5, 8, (3, 21), 1, 08F, 2, 04A, 1, 0DC, 2, 8, (4, 0), 14, 6,

Create Shape Definition Files | 133
14, 8, (7, -9), 0
*000ED, 27, lc
2, 14, 8, (-7, -21), 14, 5, 8, (3, 17), 1, 081, 2, 08B, 1, 0DC, 2, 8, (4, 0), 14, 6, 14, 8, (7, -9), 0
*000EE, 34, lce
2, 14, 8, (-7, -21), 14, 5, 8, (3, 18), 1, 8, (4, 3), 8, (4, -3), 2, 8, (-4, -5), 1, 0DC, 2, 8, (4, 0), 14, 6, 14, 8, (7, -9), 0
*000EF, 39, lc<
2, 14, 8, (-7, -21), 14, 5, 8, (3, 20), 1, 01E, 012, 016, 01A, 2, 060, 1, 01E, 012, 016, 01A.
016, 01A, 2, 8, (-2, -7), 1, 0DC, 2, 8, (4, 0), 14, 6, 14, 8, (7, -9), 0
*000F1, 56, lc
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (4, 18), 1, 032, 010, 03E, 010, 032, 2, 8, (-11, -7), 1, 0EC, 2, 0A4, 1, 032, 021, 030, 02F, 8, (1, -3), 0AC, 8, (4, 0), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000F2, 64, lc
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (5, 21), 1, 8, (9, -4), 2, 069, 1, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 8, (-1, -3), 024, 8, (11, -14), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000F3, 66, lc
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (5, 17), 1, 8, (9, 4), 2, 8, (-6, -7), 1, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 8, (-1, -3), 024, 8, (11, -14), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000F4, 73, lc
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (5, 18), 3, 2, 1, 8, (9, 6), 8, (9, -6), 2, 4, 2, 8, (-6, -4), 1, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 8, (1, -3), 024, 8, (-1, -3), 026, 027, 038, 2, 8, (11, -14), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000F5, 68, lc^*000F6, 41, kto
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (4, 18), 1, 032, 010, 03E, 010, 032, 2, 8, (-7, -9), 1, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 8, (1, -3), 024, 8, (11, -14), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000F7, 74, lc
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (4, 20), 1, 01E, 012, 016, 01A, 2, 090, 1, 01E, 012, 016, 01A, 2, 8, (-5, -6), 1, 029, 02A, 8, (-1, -3), 02C, 8, (1, -3), 02E, 02F, 030, 021, 022, 8, (1, 3), 024, 8, (-1, -3), 026, 027, 038, 2, 8, (11, -14), 14, 6, 14, 3, 2, 14, 8, (19, -18), 14, 4, 2, 0
*000F8, 41, kto
2, 14, 3, 2, 14, 8, (-19, -42), 14, 4, 2, 14, 5, 8, (8, 13), 1, 01E, 012, 016, 01A, 2, 8, (-5, -6), 1,
Chapter 8  Shapes and Shape Fonts
Chapter 8  Shapes and Shape Fonts

8, (2, 4), 1, 9, (3, 3), (1, 2), (1, 4), (0, 12), (11, 0), (0, -21), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-12, -13), 0
*00415, 40,

2, 14, 8, (-9, -21), 2, 8, (17, 21), 1, 9, (-13, 0), (0, -21), (13, 0), (0, 0), 2, 8, (-1, 11), 1, 9, (-12, 0), (0, 0), 2, 8, (15, -11), 1, 2, 14, 8, (-10, -9), 0
*00416, 66,

2, 14, 8, (-12, -21), 2, 8, (1, 0), 1, 9, (9, 12), (0, 0), 2, 8, (2, 9), 1, 9, (0, -21), (0, 0), 2, 8, (11, 21), 1, 9, (11, -11), (0, 0), 2, 8, (11, -10), 1, 9, (-9, 12), (0, 0), 2, 8, (10, -12), 1, 2, 14, 8, (-12, -9), 0
*00417, 68,

2, 14, 8, (-9, -21), 2, 8, (3, 20), 1, 9, (4, 1), (3, 0), (3, -1), (1, -2), (0, -2), (-1, -2), (-3, -2), (3, -1), (2, -2), (1, -2), (0, -2), (-1, -2), (-2, -2), (-3, -1), (-3, 1), (-3, -2), (0, 0), 2, 8, (9, 9), 1, 9, (-9, 4), (0, 0), 2, 8, (12, -12), 1, 2, 14, 8, (-9, -9), 0
*00418, 30,

2, 14, 8, (-11, -21), 2, 8, (4, 21), 1, 9, (0, -21), (14, 21), (0, -21), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-11, -9), 0
*00419, 48,

2, 14, 8, (-11, -21), 2, 8, (4, 21), 1, 9, (9, 0, -21), (14, 21), (0, -21), (0, 0), 2, 8, (-11, 26), 1, 9, (1, -1), (2, -1), (2, 0), (2, 1), (1, 1), (0, 0), 2, 8, (7, -26), 1, 2, 14, 8, (-11, -9), 0
*0041A, 46,

2, 14, 8, (-10, -21), 2, 8, (18, 0), 1, 9, (-9, 12), (0, 0), 2, 8, (9, 9), 1, 9, (-14, -14), (0, 0), 2, 8, (0, 14), 1, 9, (0, -21), (0, 0), 2, 8, (16, 0), 1, 2, 14, 8, (-10, -9), 0
*0041B, 36,

2, 14, 8, (-10, -21), 2, 8, (16, 0), 1, 9, (0, 21), (-11, 0), (0, -17), (-1, -2), (-1, -1), (-2, -1), (0, 0), 2, 8, (19, 0), 1, 2, 14, 8, (-10, -9), 0
*0041C, 32,

2, 14, 8, (-12, -21), 2, 8, (20, 0), 1, 9, (0, 21), (-8, -15), (-8, 15), (0, -21), (0, 0), 2, 8, (20, 0), 1, 2, 14, 8, (-12, -9), 0
*0041D, 46,

2, 14, 8, (-11, -21), 2, 8, (4, 21), 1, 9, (0, -21), (0, 0), 2, 8, (0, 11), 1, 9, (14, 0), (0, 0), 2, 8, (0, -11), 1, 9, (0, 21), (0, 0), 2, 8, (4, -21), 1, 2, 14, 8, (-11, -9), 0
*0041E, 64,

2, 14, 8, (-11, -21), 2, 8, (9, 21), 1, 9, (4, 0), (2, -1), (2, -2), (1, -2), (1, -3), (0, -5), (-1, -3), (-1, -2), (-2, -2), (-2, -1), (-4, 0), (-2, 1), (-2, 2), (-1, 2), (-1, 3), (0, 5), (1, 3), (1, 2), (2, 2), (2, 1), (0, 0), 2,
Create Shape Definition Files | 141

<table>
<thead>
<tr>
<th>Transformation Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>8, (13, -21), 1, 2, 14, 8, (-11, -9), 0</td>
</tr>
<tr>
<td><strong>0.041F, 30, _</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-11, -21), 2, 8, (4, 0), 1, 9, (0, 21), (14, 0), (0, -21), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-11, -9), 0</td>
</tr>
<tr>
<td><strong>0.0420, 44, _</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-10, -21), 2, 8, (4, 0), 1, 9, (0, 21), (9, 0), (3, -1), (1, -1), (1, -2), (0, -3), (-1, -2), (-1, -1), (-3, -1), (-9, 0), (0, 0), 2, 8, (16, -10), 1, 2, 14, 8, (-10, -9), 0</td>
</tr>
<tr>
<td><strong>0.0421, 62, '</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-10, -21), 2, 8, (18, 16), 1, 9, (-1, 2), (-1, 1), (-1, -1), (-2, 1), (-4, 0), (-2, -1), (-1, -1), (-1, -3), (-1, -2), (2, -2), (2, -1), (4, 0), (1, 2), (1, 2), (0, 0), 2, 8, (2, -5), 1</td>
</tr>
<tr>
<td><strong>0.0422, 36, '</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-8, -21), 2, 8, (8, 21), 1, 9, (0, -21), (0, 0), 2, 8, (-7, 21), 1, 9, (14, 0), (0, 0), 2, 8, (1, -21), 1, 2, 14, 8, (-8, -9), 0</td>
</tr>
<tr>
<td><strong>0.0423, 44, '</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-8, -21), 2, 8, (15, 21), 1, 9, (-7, -17), (-1, -2), (-1, -1), (-2, -1), (-1, -1), (0, 0), 2, 8, (-2, 21), 1, 9, (7, -17), (0, 0), 2, 8, (8, -4), 1, 2, 14, 8, (-8, -9), 0</td>
</tr>
<tr>
<td><strong>0.0424, 74, '</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-13, -21), 2, 8, (11, 19), 1, 9, (4, 0), (3, -1), (2, -1), (2, -2), (1, -2), (0, -4), (-1, -2), (-2, -2), (-2, -1), (-3, -1), (-4, 0), (-3, 1), (-2, 1), (-2, 2), (1, 2), (0, 4), (1, 2), (2, 2), (2, 1), (3, 1), (0, 0), 2, 8, (2, 2), 1, 9, (0, -21), (0, 0), 2, 8, (13, 0), 1, 2, 14, 8, (-13, -9), 0</td>
</tr>
<tr>
<td><strong>0.0425, 36, '•</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-8, -21), 2, 8, (15, 21), 1, 9, (-14, -21), (0, 0), 2, 8, (0, 21), 1, 9, (14, -21), (0, 0), 2, 8, (1, 0), 1, 2, 14, 8, (-8, -9), 0</td>
</tr>
<tr>
<td><strong>0.0426, 40, '-'</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-11, -21), 2, 8, (4, 21), 1, 9, (0, -21), (16, 0), (0, -4), (0, 0), 2, 8, (-2, 25), 1, 9, (0, -21), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-11, -13), 0</td>
</tr>
<tr>
<td><strong>0.0427, 44, '-'</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-10, -21), 2, 8, (3, 21), 1, 9, (0, -8), (1, -3), (1, -1), (3, -1), (9, 0), (0, 0), 2, 8, (0, 13), 1, 9, (0, -21), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-11, -9), 0</td>
</tr>
<tr>
<td><strong>0.0428, 40, '-'</strong></td>
</tr>
<tr>
<td>2, 14, 8, (-14, -21), 2, 8, (4, 21), 1, 9, (0, -21), (21, 0), (0, 21), (0, 0), 2, 8, (-10, 0), 1, 9, (0, -21), (0, 0), 2, 8, (14, 0), 1, 2, 14, 8, (-15, -9), 0</td>
</tr>
<tr>
<td><strong>0.0429, 50, '™</strong></td>
</tr>
</tbody>
</table>
2,14,8, (-14, -21), 2, 8, (4, 21), 1, 9, (0, -21), (23, 0), (0, -4), (0, 0), 2, 8, (-12, -25), 1, 9, (0, -21), (0, 0), 2, 8, (10, 21), 1, 9, (0, -21), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-15, -13), 0
*0042A, 48,
2, 14, 8, (-10, -21), 2, 8, (1, 21), 1, 9, (4, 0), (0, -21), (8, 0), (2, 0), (2, 1), (1, 1), (1, 2), (0, 4), (-1, 2), (-1, 1), (-2, 1), (-10, 0), (0, 0), 2, 8, (16, -12), 1, 2, 14, 8, (-11, -9), 0
*0042B, 54,
2, 14, 8, (-12, -21), 2, 8, (4, 21), 1, 9, (0, -21), (9, 0), (2, 1), (1, 1), (1, 2), (0, 4), (-1, 2), (-1, 1), (-2, 1), (-9, 0), (0, 0), 2, 8, (16, 9), 1, 9, (0, -21), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-12, -9), 0
*0042C, 44,
2, 14, 8, (-10, -21), 2, 8, (4, 21), 1, 9, (0, -21), (10, 0), (2, 1), (1, 1), (1, 2), (0, 4), (-1, 2), (-1, 1), (-2, 1), (-10, 0), (0, 0), 2, 8, (16, -12), 1, 2, 14, 8, (-10, -9), 0
*0042D, 64,
2, 14, 8, (-9, -21), 2, 8, (6, 11), 1, 9, (10, 0), (0, 0), 2, 8, (-14, 7), 1, 9, (2, 2), (2, 1), (4, 0), (2, -1), (2, -2), (1, -2), (1, -3), (0, -5), (-1, -3), (-1, -2), (-2, -2), (-2, -1), (-4, 0), (-2, 1), (0, 0), 2, 8, (17, -3), 1, 2, 14, 8, (-10, -9), 0
*0042E, 76,
2, 14, 8, (-13, -21), 2, 8, (14, 21), 1, 9, (4, 0), (2, -1), (2, -3), (1, -4), (0, -5), (-1, -4), (-2, -3), (-2, -1), (-4, 0), (-2, 1), (-2, 3), (-1, 4), (0, 5), (1, 4), (2, 3), (2, 1), (0, 0), 2, 8, (-10, -10), 1, 9, (-5, 0), (0, 0), 2, 8, (16, -12), 1, 2, 14, 8, (-13, -9), 0
*0042F, 54,
2, 14, 8, (-10, -21), 2, 8, (2, 0), 1, 9, (7, 11), (0, 0), 2, 8, (7, -11), 1, 9, (0, 21), (-9, 0), (-3, -1), (-1, -1), (-1, -2), (0, -2), (1, -2), (1, -1), (3, -1), (9, 0), (0, 0), 2, 8, (4, -11), 1, 2, 14, 8, (-10, -9), 0
*00430, 62,
2, 14, 8, (-9, -14), 2, 8, (15, 3), 1, 9, (-2, -2), (-2, -1), (-3, 0), (-2, 1), (-1, 1), (-1, 3), (0, 2), (1, 3), (2, 2), (2, 1), (3, 0), (2, -1), (2, -2), (0, 0), 2, 8, (0, 3), 1, 9, (0, -14), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-10, -9), 0
*00431, 64,
2, 14, 8, (-9, -21), 2, 8, (14, 21), 1, 9, (-2, -1), (-5, -1), (-2, -1), (-1, -2), (0, -12), (1, -2), (1, -1), (2, -1), (3, 0), (2, 1), (2, 2), (1, 3), (0, 2), (-1, 3), (-1, 1), (-1, 1), (-2, 1), (-3, 0), (-2, -1), (-2, -2), (0, 0), 2, 8, (15, -11), 1, 2, 14, 8, (-10, -9), 0
*00432, 68,
Create Shape Definition Files | 143
(-2, 0), (0, 0), 2, 8, (11, 14), 1, 9, (0, -14), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-9, -9), 0
*0043C, 32,
2, 14, 8, (-11, -14), 2, 8, (18, 0), 1, 9, (0, 14), (-7, -11), (-7, 11), (0, -14),
(0, 0), 2, 8, (18, 0), 1, 2, 14, 8, (-11, -9), 0
*0043D, 46,
2, 14, 8, (-9, -14), 2, 8, (4, 7), 1, 9, (11, 0), (0, 0), 2, 8, (0, 7), 1, 9, (0, -14), (0, 0), 2, 8, (-11, 14), 1, 9, (0, -14), (0, 0), 2, 8, (15, 0), 1, 2, 14, 8, (-10, -9), 0
*0043E, 56,
2, 14, 8, (-9, -14), 2, 8, (8, 14), 1, 9, (3, 0), (2, -1), (2, -2), (1, -3), (0, -2), (-1, -3), (-2, -2), (-2, -1), (-3, 0), (-2, 1), (-1, -3), (0, 2), (1, 3), (2, 2), (2, 1), (0, 0), 2, 8, (11, -14), 1, 2, 14, 8, (-10, -9), 0
*0043F, 46,
2, 14, 8, (-9, -14), 2, 8, (4, 14), 1, 9, (0, -14), (0, 0), 2, 8, (0, 10), 1, 9, (3, 3), (2, 1), (3, 0), (2, -1), (1, -2), (0, -11), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-10, -9), 0
*00440, 60,
2, 14, 8, (-9, -14), 2, 8, (4, 14), 1, 9, (0, -21), (0, 0), 2, 8, (0, 18), 1, 9, (2, 2), (2, 1), (3, 0), (2, -1), (2, -2), (1, -3), (0, -2), (-1, -3), (-2, -2), (-2, -1), (-3, 0), (-2, 1), (-2, 2), (0, 0), 2, 8, (15, -3), 1, 2, 14, 8, (-10, -16), 0
*00441, 50,
2, 14, 8, (-8, -14), 2, 8, (15, 11), 1, 9, (-2, 2), (-2, 1), (-3, 0), (-2, -1), (-2, -2), (-1, -3), (0, -2), (1, -3), (2, -2), (2, -1), (3, 0), (2, 1), (2, 2), (0, 0), 2, 8, (2, -3), 1, 2, 14, 8, (-9, -9), 0
*00442, 66,
2, 14, 8, (-14, -14), 2, 8, (4, 14), 1, 9, (0, -14), (0, 0), 2, 8, (0, 10), 1, 9, (3, 3), (2, 1), (2, 0), (2, -1), (1, -3), (0, -10), (0, 0), 2, 8, (0, 10), 1, 9, (3, 3), (2, 1), (2, 0), (2, -1), (1, -3), (0, -10), (0, 0), 2, 8, (4, 0), 1, 2, 14, 8, (-14, -9), 0
*00443, 44,
2, 14, 8, (-7, -14), 2, 8, (1, 14), 1, 9, (6, -14), (0, 0), 2, 8, (6, 14), 1, 9, (-6, -14), (-2, -4), (-2, -2), (-2, -1), (-1, 0), (0, 0), 2, 8, (14, 7), 1, 2, 14, 8, (-7, -16), 0
*00444, 66,
2, 14, 8, (-11, -14), 2, 8, (14, 14), 1, 9, (-6, 0), (-2, -1), (-2, -2), (-1, -3), (0, -2), (-1, -3), (2, -2), (2, -1), (6, 0), (2, 1), (2, 2), (1, 3), (0, 2), (-1, 3), (-2, 2), (-2, 1), (0, 0), 2, 8, (-3, 0), 1, 9, (0, -21), (0, 0), 2, 8, (11, 7), 1, 2,
Create Shape Definition Files | 145
Chapter 8  Shapes and Shape Fonts
Extended Standard Font for UNICODE

;;
;; txt.shp - Extended Standard Font for UNICODE
;;
;; Copyright 1997 by Autodesk, Inc.
;;
;; Permission to use, copy, modify, and distribute this software for
;; any purpose and without fee is hereby granted, provided that the
;; above copyright notice appears in all copies and that the
;; rights notice below appear in all supporting documentation.
;;
;; Use, duplication, or disclosure by the U.S. Government
;; is subject
;; to restrictions set forth in FAR 52.227-19 (Commercial
;; Computer
;; Software - Restricted Rights) and DFAR
;; 252.227-7013(c)(1)(ii)
;; (Rights in Technical Data and Computer Software), as
;; applicable.
;;
*UNIFONT,6,TXT Copyright 1997 by Autodesk, Inc.
6,2,2,0,0,0
*00000A,7,1f
2,0AC,14,8,(9,10),0
*00020,7,spc
2,060,14,8,(-6,-8),0
*00021,17,kexc
2,14,06C,1,014,2,014,1,044,2,020,06C,14,8,(-2,-3),0
*00022,20,kdblqt
2,14,8,(-1,-6),044,1,023,2,010,1,02B,2,04C,030,14,8,(-3,1),0
*00023,27,kns
2,14,8,(-2,-6),024,1,040,2,024,1,048,2,023,1,06C,2,020,1,064,2,
06D,14,8,(-4,-3),0
*00024,25,kds
2,14,8,(-2,-6),014,1,030,012,016,028,016,012,030,2,027,1,06C,2,
040,14,8,(-4,-3),0
2,14,8,(-2,-6),041,1,028,018,016,024,012,010,02E,2,044,1,06C,2,020,14,8,(-4,-3),0
*00065,20,lce
2,14,04B,024,1,030,012,016,028,01A,02C,01E,020,2,030,14,8,(-4,-3),0
*00066,22,lcf
2,14,8,(-2,-6),034,1,030,2,023,1,016,018,01A,05C,2,050,14,8,(-4,-3),0
*00067,22,lcg
2,14,04B,01C,1,01E,020,012,044,016,028,01A,02C,01E,030,2,020,14,8,(-4,-5),0
*00068,21,lch
2,14,8,(-2,-6),1,064,2,04C,1,022,010,01E,03C,2,020,14,8,(-4,-3),0
*00069,17,lci
2,14,06C,1,044,2,014,1,014,2,020,06C,14,8,(-2,-3),0
*0006A,35,lcj
2,14,3,2,14,8,(-3,-12),14,4,2,01C,1,01E,010,012,054,2,014,1,014,2,020,06C,14,3,2,14,8,(-7,-10),14,4,2,0
*0006B,23,lck
2,14,8,(-2,-6),1,064,2,04C,1,020,022,2,02A,1,02E,2,020,14,8,(-4,-3),0
*0006C,28,lcl
2,14,3,2,14,8,(-1,-12),14,4,2,064,1,05C,01E,2,020,14,3,2,14,8,(-5,-6),14,4,2,0
*0006D,24,lcm
2,14,04B,1,044,2,01C,1,012,01E,01C,2,014,1,012,01E,03C,2,020,14,8,(-4,-3),0
*0006E,21,lcn
2,14,8,(-2,-6),1,044,2,02C,1,022,010,01E,03C,2,020,14,8,(-4,-3),0
*0006F,20,lco
2,14,04B,030,1,028,016,024,012,020,01E,02C,01A,2,030,14,8,(-4,-3),0
*00070,22,lcp
2,14,04B,02C,1,064,2,01C,1,012,020,01E,02C,01A,038,2,060,14,8,(-4,-5),0
*00071,22,lcq
2,14,04B,04F,1,064,2,01C,1,016,028,01A,02C,01E,030,2,020,14,8,(-4,-5),0
*00072,19,lcr
2,14,04B,1,044,2,02C,1,022,010,01E,2,020,03C,14,8,(-4,-3),0
Chapter 8   Shapes and Shape Fonts
2,14,8, (-1,-6), 044, 1, 020, 2, 014, 1, 016, 01A, 2, 01E, 1, 04C, 2, 018, 1, 020, 2, 020, 14, 03A, 0
*000CF, 29, uc
2,14,8, (-1,-6), 044, 1, 020, 2, 014, 1, 014, 2, 028, 1, 01C, 2, 01E, 1, 04C, 2, 018, 1, 020, 2, 020, 14, 03A, 0
*000D0, 25, uc
2,14,8, (-2,-6), 1, 064, 030, 01E, 04C, 01A, 038, 2, 024, 015, 1, 020, 2, 01D, 04F, 14, 8, (-4,-3), 0
*000D1, 25, uc
2,14,8, (-2,-6), 1, 044, 04E, 044, 2, 048, 014, 1, 012, 02F, 012, 2, 8, (2,-6), 14, 8, (-4,-3), 0
*000D2, 25, uc
2,14,8, (-2,-6), 1, 044, 044, 2, 048, 014, 1, 012, 02F, 012, 2, 8, (2,-6), 14, 8, (-4,-3), 0
*000D3, 25, uc
2,14,8, (-2,-6), 1, 044, 040, 2, 016, 1, 027, 2, 02C, 030, 1, 04C, 048, 2, 060, 14, 8, (-4,-3), 0
*000D4, 26, uc
2,14,8, (-2,-6), 1, 044, 040, 2, 025, 1, 029, 2, 01C, 030, 1, 04C, 048, 2, 060, 14, 8, (-4,-3), 0
*000D5, 27, uc
2,14,8, (-2,-6), 1, 044, 040, 2, 024, 1, 01A, 027, 01A, 2, 01C, 040, 1, 04C, 048, 2, 060, 14, 8, (-4,-3), 0
*000D6, 31, uc
2,14,8, (-2,-6), 1, 044, 02A, 2, 014, 1, 012, 02B, 1, 040, 2, 016, 1, 014, 2, 02D, 1, 04C, 048, 2, 060, 14, 8, (-4,-3), 0
*000D8, 29, ucd
2,14,8, (-2,-6), 1, 044, 02A, 2, 014, 1, 012, 02B, 1, 040, 2, 016, 1, 014, 2, 02D, 1, 04C, 048, 2, 060, 14, 8, (-4,-3), 0
*000D9, 24, uc
2,14,8, (-2,-6), 064, 1, 05C, 01E, 020, 012, 054, 2, 01A, 1, 027, 2, 050, 06C, 14, 8, (-4,-3), 0
*000DA, 24, uc
2,14,8, (-2,-6), 064, 1, 05C, 01E, 020, 012, 054, 2, 01A, 1, 027, 2, 050, 06C, 14, 8, (-4,-3), 0
*000DB, 25, uc
2,14,8, (-2,-6), 064, 1, 05C, 01E, 020, 012, 054, 2, 01A, 1, 016, 01A, 2, 050, 05C, 14, 8, (-4,-3), 0
*000DC, 27, uc
2,14,8, (-2,-6), 064, 1, 05C, 01E, 020, 012, 054, 2, 01A, 1, 014, 2, 028, 1, 01C,
2, 05E, 14, 8, (-4, -3), 0
*000EA, 27, 1c*
2, 14, 8, (-2, -6), 024, 1, 030, 012, 016, 028, 01A, 02C, 01E, 020, 2, 054, 1, 016,
01A, 2, 05E, 14, 8, (-4, -3), 0
*000EB, 31, 1c*
2, 14, 8, (-2, -6), 024, 1, 030, 012, 016, 028, 01A, 02C, 01E, 020, 2, 064, 010, 1,
01C, 2, 048, 1, 014, 2, 06E, 14, 8, (-4, -3), 0
*000EC, 18, 1c_
2, 14, 8, (-1, -6), 010, 1, 044, 2, 025, 1, 02F, 2, 020, 05C, 14, 03A, 0
*000ED, 18, 1c
2, 14, 8, (-1, -6), 010, 1, 044, 2, 016, 1, 021, 2, 020, 06C, 14, 03A, 0
*000EE, 19, 1cE
2, 14, 8, (-1, -6), 010, 1, 044, 2, 016, 1, 012, 01E, 2, 020, 05C, 14, 03A, 0
*000EF, 22, 1c
2, 14, 8, (-1, -6), 010, 1, 044, 2, 016, 1, 014, 2, 020, 1, 01C, 2, 020, 05C,
14, 03A, 0
*000F0, 30, 1c
2, 14, 8, (-2, -6), 8, (3, 4), 1, 028, 01A, 02C, 01E, 020, 012, 024, 036, 2, 01C, 1,
021, 2, 8, (3, -6), 14, 8, (-4, -3), 0
*000F1, 27, 1c
2, 14, 8, (-2, -6), 1, 044, 2, 014, 1, 012, 02E, 012, 2, 04A, 1, 022, 010, 01E, 03C,
2, 020, 14, 8, (-4, -3), 0
*000F2, 27, 1c_
2, 14, 8, (-2, -6), 030, 1, 028, 016, 024, 012, 020, 01E, 02C, 01A, 2, 054, 1, 027,
2, 050, 06C, 14, 8, (-4, -3), 0
*000F3, 26, 1c
2, 14, 8, (-2, -6), 030, 1, 028, 016, 024, 012, 020, 01E, 02C, 01A, 2, 064, 1, 029,
2, 05E, 14, 8, (-4, -3), 0
*000F4, 27, 1c”
2, 14, 8, (-2, -6), 030, 1, 028, 016, 024, 012, 020, 01E, 02C, 01A, 2, 054, 1, 016,
01A, 2, 05E, 14, 8, (-4, -3), 0
*000F5, 32, 1c”
2, 14, 8, (-2, -6), 030, 1, 028, 016, 024, 012, 020, 01E, 02C, 01A, 2, 8, (-3, 5),
1, 012, 02E, 012, 2, 8, (-2, 6), 14, 8, (-4, -3), 0
*000F6, 35, 1c”
2, 14, 8, (-2, -6), 030, 1, 028, 016, 024, 2, 034, 1, 01C, 2, 040, 1, 014, 2,
8, (-4, -3), 1, 012, 020, 01E, 02C, 01A, 2, 030, 14, 8, (-4, -3), 0
*000F7, 23, kto
2, 14, 8, (-2, -5), 021, 1, 014, 2, 021, 1, 048, 2, 022, 1, 01C, 2, 04E,
14, 8, (-4, -2), 0
*000F8, 24, 1cd”
Chapter 8  Shapes and Shape Fonts
1, 024, 2, 07C, 050, 14, 8, (-4, -3), 0
*00158, 33, c252
2, 14, 8, (-2, -8), 1, 064, 030, 01E, 01C, 01A, 038, 2, 010, 1, 03E, 2, 028, 074, 1,
012, 2, 01A, 1, 016, 2, 050, 08C, 14, 8, (-4, -3), 0
*00159, 29, c253
2, 14, 8, (-2, -6), 1, 044, 2, 02C, 1, 022, 010, 01E, 2, 026, 1, 012, 2, 01A, 1, 016,
2, 050, 06C, 14, 8, (-4, -3), 0
*0015A, 27, c151
2, 14, 8, (-2, -6), 014, 1, 01E, 020, 012, 046, 012, 020, 01E, 2, 024, 038, 1, 021,
2, 08C, 030, 14, 8, (-4, -3), 0
*0015B, 31, c152
2, 14, 04B, 1, 030, 012, 016, 028, 016, 012, 030, 2, 3, 2, 8, (-5, 2), 1, 021, 2,
029, 8, (9, -10), 4, 2, 14, 8, (-4, -3), 0
*00160, 30, c230
2, 14, 8, (-2, -8), 014, 1, 01E, 020, 012, 046, 012, 020, 01E, 2, 026, 1, 012, 2,
01A, 1, 016, 2, 050, 08C, 14, 8, (-4, -3), 0
*00161, 29, c231
2, 14, 8, (-2, -6), 1, 030, 012, 016, 028, 016, 012, 030, 2, 027, 1, 012, 2, 01A, 1,
016, 2, 050, 06C, 14, 8, (-4, -3), 0
*00164, 28, c155
2, 14, 8, (-2, -8), 064, 1, 040, 2, 028, 1, 06C, 2, 074, 1, 012, 2, 01A, 1, 016, 2,
050, 08C, 14, 8, (-4, -3), 0
*00165, 26, c156
2, 14, 8, (-2, -6), 044, 1, 040, 2, 026, 1, 05C, 01E, 012, 2, 054, 1, 01A, 2, 030,
05C, 14, 8, (-4, -3), 0
*00166, 27, c222
2, 14, 8, (-2, -9), 064, 1, 05C, 01E, 020, 012, 054, 2, 027, 1, 012, 016, 01A, 01E,
2, 040, 07C, 14, 8, (-4, -3), 0
*0016F, 31, c133
2, 14, 8, (-2, -7), 044, 1, 03C, 01E, 010, 022, 2, 024, 1, 04C, 2, 028, 054, 1, 012,
016, 01A, 01E, 2, 040, 05C, 14, 8, (-4, -3), 0
*00170, 28, c235
2, 14, 8, (-2, -6), 064, 1, 05C, 01E, 020, 012, 054, 2, 016, 1, 03C, 2, 028, 1, 034,
2, 07C, 050, 14, 8, (-4, -3), 0
*00171, 30, ccue
2, 14, 04B, 044, 1, 03C, 01E, 010, 022, 2, 024, 1, 04C, 2, 074, 018, 1, 02C, 2, 028,
1, 024, 2, 07C, 050, 14, 8, (-4, -3), 0
*00179, 25, c141
2, 14, 8, (-2, -6), 064, 1, 040, 8, (-4, -6), 040, 2, 038, 074, 1, 021, 2, 08C, 030,
14, 8, (-4, -3), 0
*0017A, 28, c171
Chapter 8  Shapes and Shape Fonts
Create Shape Definition Files | 163
060,01C,14,8,(-4,-3),0
*00438,17,1cri
2,14,8,(-2,-6),044,1,04C,042,04C,2,020,14,8,(-4,-3),0
*00439,23,1crii
2,14,8,(-2,-6),044,1,04C,042,04C,2,044,018,1,028,2,050,04C,14,8,(-4,-3),0
*0043A,19,1crk
2,14,8,(-2,-6),1,044,02C,020,022,02A,02E,2,020,14,8,(-4,-3),0
*0043B,16,1crl
2,14,8,(-2,-6),1,043,020,04C,2,020,14,8,(-4,-3),0
*0043C,17,1crm
2,14,8,(-2,-6),1,044,02E,022,04C,2,020,14,8,(-4,-3),0
*0043D,18,1crn
2,14,8,(-2,-6),1,044,02C,040,024,04C,2,020,14,8,(-4,-3),0
*0043E,25,1cro
2,14,04B,14,8,(0,-2),014,1,024,012,02O,01E,02C,01A,028,016,2,060,01C,14,8,(-4,-3),0
*0043F,16,1crp
2,14,8,(-2,-6),1,044,04O,04C,2,020,14,8,(-4,-3),0
*00440,20,1crr
2,14,8,(-2,-6),1,044,030,01E,01C,01A,038,2,060,01C,14,8,(-4,-3),0
*00441,23,1crr
2,14,8,(-2,-6),040,014,1,01A,028,016,024,012,02O,01E,2,020,03C,14,8,(-4,-3),0
*00442,18,1crt
2,14,8,(-2,-6),02O,1,044,028,04O,2,02O,04C,14,8,(-4,-3),0
*00443,22,1cru
2,14,8,(-2,-6),014,1,01E,02O,020,012,034,02C,028,026,2,060,04C,14,8,(-4,-3),0
*00444,25,1crf
2,14,8,(-2,-6),02O,1,044,018,01C,01E,02O,012,014,016,018,2,040,04C,14,8,(-4,-3),0
*00445,20,1crh
2,14,04B,14,8,(0,-2),1,042,2,048,1,04E,2,020,14,8,(-4,-3),0
*00446,21,1crn
2,14,8,(-2,-6),044,1,04C,030,044,04C,010,01C,2,014,020,14,8,(-4,-3),0
Chapter 8  Shapes and Shape Fonts
Big Font Descriptions

Some languages, such as Japanese, use text fonts with thousands of non-ASCII characters. In order for drawings to contain such text, AutoCAD for Mac supports a special form of shape definition file called a Big Font file.

Define a Big Font

Special codes in the first line of a Big Font file specify how to read two-byte hexadecimal codes.

A font with hundreds or thousands of characters must be handled differently from a font containing the ASCII set of up to 256 characters. In addition to using more complicated techniques for searching the file, AutoCAD for Mac needs a way to represent characters with two-byte codes as well as one-byte codes. Both situations are addressed by the use of special codes at the beginning of a Big Font file.

The first line of a Big Font shape definition file must be as follows:

```
*BIGFONT nchars, nranges, b1, e1, b2, e2, ...
```

where `nchars` is the approximate number of character definitions in this set; if it is off by more than about 10 percent, either speed or file size suffers. You can use the rest of the line to name special character codes (escape codes) that signify the start of a two-byte code. For example, on Japanese computers, Kanji characters start with hexadecimal codes in the range 90-AF or E0-FF. When the operating system sees one of these codes, it reads the next byte and combines the two bytes into a code for one Kanji character. In the *BIGFONT line, `nranges` tells how many contiguous ranges of numbers are used as escape codes; `b1, e1, b2, e2`, and so on, define the beginning and ending codes in each range. Therefore, the header for a Japanese Big Font file might look like this:

```
*BIGFONT 4000, 2, 090, 0AF, 0E0, 0FF
```
After the *BIGFONT line, the font definition is just like a regular AutoCAD for Mac text font, except that character codes (shape numbers) can have values up to 65535.

**Define an Extended Big Font File**

To reduce the size of composite Kanji characters, you can define an extended Big Font file. Extended big fonts use the subshape code, followed immediately by a 0.

The first line of an extended Big Font file is the same as the regular Big Font file. This is the format for the remaining lines of the file:

```
*0,5,font-namecharacter-height, 0, modes, character-width,0
. .
*shape-number, defbytes, shape-name
 . code,0,primitive#, basepoint-x, basepoint-y, width, height,
 . .
 . code,0,primitive#, basepoint-x, basepoint-y, width, height,
 .
terminator
```

The following list describes the fields of a Big Font definition file:

**character height** Used along with character width to indicate the number of units that define the font characters.

**character width** Used along with character height to indicate the number of units that define the font characters. The character-height and character-width values are used to scale the primitives of the font. In this context, primitives are the points, lines, polygons, or character strings of the font geometrically oriented in two-dimensional space. A Kanji character consists of several primitives used repeatedly in different scales and combinations.

**modes** The modes byte should be 0 for a horizontally oriented font and 2 for a dual-orientation (horizontal or vertical) font. The special 00E (14) command code is honored only when modes is set to 2.

**shape-number** Character code.
**defbytes** Byte size. It is always 2 bytes, consisting of a hexadecimal or a combination of decimal and hexadecimal codes.

**shape-name** Character name.

**code** Shape description special code. It is always 7 so that it can use the subshape feature.

**primitive#** Reference to the subshape number. It is always 2 bytes.

**basepoint-x** X origin of the primitive.

**basepoint-y** Y origin of the primitive.

**width** Scale of the width of the primitive.

**height** Scale of the height of the primitive.

**terminator** End-of-file indicator for the shape definition. It is always 0.

To arrive at the scale factor, AutoCAD for Mac scales down the primitive to a square unit and then multiplies it by the height and width to get the shape of the character. Character codes (shape numbers) in the Big Font shape definition file can have values up to 65535. The following table describes the fields of the extended Big Font file.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Byte size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>shape-number</strong></td>
<td>xxxx</td>
<td>2 bytes</td>
<td>Character code</td>
</tr>
<tr>
<td><strong>code</strong></td>
<td>7,0</td>
<td>2 bytes</td>
<td>Extended font definition</td>
</tr>
<tr>
<td><strong>primitive#</strong></td>
<td>xxxx</td>
<td>2 bytes</td>
<td>Refer to subshape number</td>
</tr>
<tr>
<td><strong>basepoint-x</strong></td>
<td></td>
<td>1 byte</td>
<td>Primitive X origin</td>
</tr>
<tr>
<td><strong>basepoint-y</strong></td>
<td></td>
<td>1 byte</td>
<td>Primitive Y origin</td>
</tr>
<tr>
<td><strong>width</strong></td>
<td></td>
<td>1 byte</td>
<td>Scale of primitive width</td>
</tr>
<tr>
<td><strong>height</strong></td>
<td></td>
<td>1 byte</td>
<td>Scale of primitive height</td>
</tr>
</tbody>
</table>
Fields of the extended Big Font file

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Byte size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminator</td>
<td>0</td>
<td>1 byte</td>
<td>End of shape definition</td>
</tr>
</tbody>
</table>

The following figure is an example of a 16 x 16 dot matrix that you could use to design an extended Big Font, such as a Kanji character. In the example, the distance between each dot is one unit. The callout points to a square unit.

A square matrix for a Kanji character

The following figure shows examples of Kanji characters. Each character occupies an M×N matrix (matrices don’t have to be square), similar to the one shown in the previous figure. The numbers above each figure are the associated shape numbers.
Examples of Kanji characters
The following figure shows Kanji primitives.
Examples of Kanji primitives

NOTE
Not all fonts are defined in a square matrix; some are defined in rectangular matrices.

Example: Shape Definition File for an Extended Big Font

*BIGFONT 50,1,080,09e
*0,5,Extended Font
15,0,2,15,0
*08D91,31,unspecified
2,0e,8,-7,-15,
7,0,08cfb,0,0,16,16,7,0,08bca,2,3,12,9,
2,8,18,0,2,0e,8,-11,-3,0
*08CD8,31,unspecified
2,0e,8,-7,-15,
7,0,08be0,0,0,16,16,7,0,08cc3,8,0,8,16,
2,8,18,0,2,0e,8,-11,-3,0
*08ADF,31,unspecified
2,0e,8,-7,-15,
7,0,089a4,0,0,8,16,7,0,08cb3,8,0,8,16,
2,8,18,0,2,0e,8,-11,-3,0
*08CE8,39,unspecified
2,0e,8,-7,-15,
7,0,089a4,0,1,5,14,7,0,08cc3,5,2,5,14,7,0,08c8e,9,0,7,
16,2,8,18,0,2,0e,8,-11,-3,0
Create Shape Definition Files | 173
Use Big Font Text in a Drawing

To use a Big Font for drawing text, you set up a text style and then specify the name of the Big Font file.

To use a Big Font for drawing text, you must set up a text style by using the STYLE command and then specify the name of the Big Font file. The same text style can use a normal ASCII font as well; enter only the two file names, separated by a comma. The following example uses the command prompt version of the STYLE command.

Command: -style
Enter name of text style or [?] <current>: style_name
Specify full font name or font file name (TTF or SHX): txt,greek

AutoCAD for Mac assumes that the first name is the normal font and that the second is the big font.

If you enter only one name, AutoCAD for Mac assumes it is the normal font and removes any associated Big Font.

By using leading or trailing commas when specifying the font file names, you can change one font without affecting the other, as shown in the following table.

<table>
<thead>
<tr>
<th>Input for changing fonts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
</tr>
<tr>
<td>normal, big</td>
</tr>
<tr>
<td>normal,</td>
</tr>
<tr>
<td>,big</td>
</tr>
<tr>
<td>normal</td>
</tr>
<tr>
<td>ENTER (null response)</td>
</tr>
</tbody>
</table>

When you use the -STYLE command to list styles or to revise an existing style, AutoCAD for Mac displays the normal font file, a comma, and the Big Font
file. If the style has only a Big Font file, it is displayed with a leading comma: ,greek.

For each character in a text string, AutoCAD for Mac searches the Big Font file first. If the character is not found there, the normal font file is searched.

To enable Big Fonts from the Text Style dialog box, choose the Big Font file you want to use from the Asian Set list.

**Use a Big Font to Extend a Font**

To include special symbols in text strings, you can use a Big Font instead of extending a standard text font.

In some drafting disciplines, many special symbols can appear in text strings. The AutoCAD for Mac standard text fonts can be extended to include special symbols. However, extending standard text fonts has several limitations:

- The number of shapes is 255 per font file.
- Standard character set uses almost half the available shape numbers. Only codes 1 through 9, 11 through 31, and 130 through 255 are available.
- Multiple text fonts require duplication of the symbol definitions in each font.
- Special symbols require that you enter `%%nnn`, where `nnn` is the symbol’s shape number.

The Big Font mechanism avoids these problems. You can select one or more seldom-used characters, such as the tilde (˜) or the vertical bar (|), as an escape code, and use the next character to select the appropriate special symbol. For instance, you can use the following Big Font file to draw Greek letters by entering a vertical bar (|, ASCII code 124) followed by the equivalent Roman letter. Because the first byte of each character is 124, the character codes are biased by 124 x 256, or 31744.

```
*BIGFONT 60,1,124,124
*0,4,Greek
above, below, modes, 0
*31809,n,uca
...
uppercase Alpha definition, invoked by "|A"
*31810,n,ucb
...
uppercase Beta definition, invoked by "|B"
*31841,n,1ca
```
lowercase Alpha definition, invoked by "|a"
*31842,n,lcb

lowercase Beta definition, invoked by "|b"
*31868,n,vbar

vertical bar definition, invoked by "||"

Unicode Font Descriptions

A single Unicode font, due to its large character set, is capable of supporting all languages and platforms. Unicode shape definition files are virtually identical in format and syntax to regular AutoCAD for Mac shape definition files.

The main difference is in the syntax of the font header as shown in the following code:

```
*UNIFONT,6,font-name
above,below,modes,encoding,type,0
```

The `font-name`, `above`, `below`, and `modes` parameters are the same as in regular fonts. The remaining two parameters are defined as follows:

- **encoding** Font encoding. Uses one of the following integer values.
  - 0 Unicode
  - 1 Packed multibyte 1
  - 2 Shape file

- **type** Font embedding information. Specifies whether the font is licensed. Licensed fonts must not be modified or exchanged. Bitcoded values can be added.
  - 0 Font can be embedded
  - 1 Font cannot be embedded
  - 2 Embedding is read-only

Another important difference is the handling of the code 7 subshape reference. If a shape description includes a code 7 subshape reference, the data following the code 7 is interpreted as a two-byte value. This affects the total number of data bytes (defbytes) in the shape description header. For example, the following shape description is found in the `romans.shp` file:
The second field in the header represents the total number of bytes in the shape description. If you are not used to working with Unicode font descriptions, you may be inclined to use three bytes rather than four, but this would cause an error during the compiling of the SHP file. This is true even if the shape number you are referencing is not in the two-byte range (below 255); the compiler always uses two bytes for this value, so you must account for that in the header.

The only other difference between Unifont shape definitions and regular shape definitions is the shape numbers. The Unifont shape definitions that AutoCAD for Mac provides use hexadecimal shape numbers as opposed to decimal values. Although hexadecimal numbers are not required, their use makes it easier to cross-reference the shape numbers with the \U control character values.

**Superscripts and Subscripts in SHX Files**

You can modify shape definition files to improve their ability to display superscripts and subscripts.

The AutoCAD for Mac SHX fonts have limited superscript and subscript capabilities. However, it is relatively easy to modify shape definition files to improve superscript and subscript capability.

Creating superscripts and subscripts requires two steps. First, the “imaginary pen” that is creating the text, vector by vector, on your screen needs to be shifted up or down. Then, the font “scale” needs to be reduced. In addition, the reverse process has to take place to return to the normal font. The font needs to recognize four new keys: two for superscripts and two for subscripts. To avoid altering the existing font definitions, you can access these with the numeric keypad on your keyboard.

**Superscripts and Subscripts in SHX Files**

To add superscript and subscript definitions to a font

This example procedure is based on the AutoCAD for Mac Romans font file, although a similar method applies to any AutoCAD for Mac font. This procedure adds four new shape definitions to a font: super_on, super_off, sub_on, and sub_off, which control the position and size of the characters.
that follow. For simplicity, this example replaces the left- and right-bracket characters ([and]) and the left and right curly brace characters ({and}) with the new characters. You may choose to replace other characters or use a shape number in the extended range (ASCII codes 128 through 256). If you use an extended shape number, you need to use the \%\%nnn method (where nnn is the ASCII value of the character) for placing the new characters.

1. Edit your SHP file with an ASCII text editor.

2. Search for the shape definitions of the characters you are replacing. To comment out those definitions so the new definitions can take their place, insert a semicolon in front of each line of the shape definition. The shape definition may continue for a number of lines.
   
   The left- and right-bracket characters have ASCII values of 91 and 93 (05B and 05D hex values, if the font is Unicode). The left and right curly brace characters have ASCII values of 123 and 125 (07B and 07D hex).

3. Add the first and second values on the second line of the definition, and divide the total by 2 as shown in the following example:

   *UNIFONT,6,Extended Simplex Roman for UNICODE
   21,7,2,0 21 + 7 = 28, then 28 / 2 = 14. This number is used later.

4. Add the following lines to the end of the SHP file:

   *91,8,super_on
   2,8,(0,14),003,2,1,0

   *93,8,super_off
   2,004,2,8,(0,-14),1,0

   *123,8,sub_on
   2,8,(0,-14),003,2,1,0

   *125,8,sub_off
   2,004,2,8,(0,14),1,0

   Notice the 14 and -14 values in the preceding lines. They are Y axis offsets for the imaginary pen. The value 14 is half the maximum height of a character in this font, which is the correct approximation for superscripts and subscripts. This value needs to be calculated for each font file, but you can modify it any way you want.

5. Save the file.

6. Use the COMPILE command to compile the SHP file.
   Once the shape is compiled and an appropriate style is defined, you can access the new pen-up and pen-down commands by entering the [ , ] ,
and } characters. The [ character initiates superscript and the ] character returns from superscript to normal. The { character initiates subscript and the } character returns from subscript to normal.
Index

A

acad.lsp file 81
acaddoc.lsp file 82
addition DIESEL function 64
aliases
  creating for commands 13
alignment
  linetype patterns 17
alternate font mapping files 9
and DIESEL function 66
angtos DIESEL function 67
ApplicationPackage element 91
applications
  AutoLISP 78
  ObjectARX 86
  plug-in applications 87
  Visual LISP 78
AutoCAD Runtime Extension 86
AutoLISP (LSP) applications
  about 78
  acad.lsp file 81
  acaddoc.lsp file 82
  DIESEL expressions and 62
  errors when loading 83
  loading 78, 80
  loading ObjectARX applications with 86
  macros and 38, 46, 61
  MNL files 83
  S STARTUP function 84
  system variables and 78

B

batch files 6
Big Fonts
  about 167
  defining 167
  examples 170
  extended Big Font files 168
  extending text fonts with 175
  text styles and 174
  borders (menu items) 48
  BUNDLE packages 87, 101

C

canceling commands 39
caret syntax in macros 40
CFG files 9
characters
  Big Fonts 167
  including in linetypes 19
  macro control characters 38, 40
  shape fonts
  codes (shape specification bytes) 110
Command element 100
command line
  switches 75
command line switches
  running scripts and 75
command scripts 73
commands
  about 34
  aliases 13
  canceling 39
  creating 35
  customizing 34
  editing 35
  Help topics about 49
  images for 50
  macros and 38, 39, 44
  menus 50
  name validity 4
  pausing for input 42
  properties 34
  repeating 44
  reusing 35
  scripts 73
  submenus and 51
  tool sets and tool groups 54

181 | Index
<table>
<thead>
<tr>
<th>tooltips 49</th>
</tr>
</thead>
<tbody>
<tr>
<td>transparent commands 42</td>
</tr>
<tr>
<td>Commands element 100</td>
</tr>
<tr>
<td>comments</td>
</tr>
<tr>
<td>AutoLISP commenting 78</td>
</tr>
<tr>
<td>script commenting 74</td>
</tr>
<tr>
<td>CompanyDetails element 93</td>
</tr>
<tr>
<td>compiling shape or font files 105</td>
</tr>
<tr>
<td>ComponentEntry element 96</td>
</tr>
<tr>
<td>Components element 94</td>
</tr>
<tr>
<td>conditional expressions in macros 45</td>
</tr>
<tr>
<td>configuration files (plotters) locating 9</td>
</tr>
<tr>
<td>configuration files (pointing devices) locating 9</td>
</tr>
<tr>
<td>control characters in macros 40</td>
</tr>
<tr>
<td>copyright restrictions on fonts 106</td>
</tr>
<tr>
<td>custom icon files 9</td>
</tr>
<tr>
<td>customization about 1</td>
</tr>
<tr>
<td>customization files (CUI) DIESEL expressions in 61</td>
</tr>
<tr>
<td>directory structure and 4</td>
</tr>
<tr>
<td>Customize dialog box about 33</td>
</tr>
<tr>
<td>customizing about customization 33</td>
</tr>
<tr>
<td>commands 34</td>
</tr>
<tr>
<td>Customize dialog box 33</td>
</tr>
<tr>
<td>functions and features 1</td>
</tr>
<tr>
<td>glossary of terms for 34</td>
</tr>
<tr>
<td>hatch patterns 25</td>
</tr>
<tr>
<td>hatches 25</td>
</tr>
<tr>
<td>linetypes 15</td>
</tr>
<tr>
<td>menus 50</td>
</tr>
<tr>
<td>tool groups 54</td>
</tr>
<tr>
<td>tool sets 54</td>
</tr>
<tr>
<td>user interface 33</td>
</tr>
</tbody>
</table>

| DCL files (dialog control language) 9, 78 |
| DIESEL expressions about 61 |
| error messages 72 |
| macros and 38, 45, 61 |
| pull-down menu labels 63 |
| DIESEL functions addition 64 |
| and 66 |
| angtos 67 |
| catalog 64 |
| division 65 |
| edtime 67 |
| eq 69 |
| equal to 65 |
| eval 69 |
| fix 69 |
| getenv 69 |
| getvar 70 |
| greater than 66 |
| greater than or equal to 66 |
| if 70 |
| index 70 |
| less than 65 |
| less than or equal to 66 |
| multiplication 65 |
| not equal to 66 |
| nth 70 |
| or 71 |
| rtos 71 |
| strlen 71 |
| substr 71 |
| subtraction 65 |
| upper 71 |
| xor 72 |

| dimensions |
| font characters for 118 |
| direction |
| shape specification codes 108 |
| directories 3 |
| directory paths |
| AutoLISP files 79 |
| program and support files 3 |
| division DIESEL function 65 |
| dots in linetype definitions 16 |
drawing files (DWG)
  folders and maintenance  5
drawing templates
  locating files  12
drop-downs
  tool groups  56
DWT files
  locating  12

E
  echoes in macros  40
  editing
    commands  35
    edtime DIESEL function  67
  eq DIESEL function  69
  equal to DIESEL function  65
  error messages
    AutoLISP  84
    DIESEL  72
  errors
    AutoLISP errors  83
  eval DIESEL function  69
  extended Big Font files  168
  extended help  49

F
  FAS files  78
  fields (font definition files)
    Big Font files  168
    shape font files  107
  files
    locations  3
    maintenance  5
    roaming profiles and  7
  fix DIESEL functions  69
  flyouts
    tool groups  56
  FMP files (font mapping)  9
  folders
    BUNDLE packages  101
    file locations  3
    plug-in application organization  103
  font mapping files  9
  fonts
    compiling  105
    creating  117
    dimensioning characters in  118
    including in linetypes  19
    shape fonts
    special symbols in  175
    superscripts and subscripts in  177

G
  getenv DIESEL function  69
  getvar DIESEL function  70
  graphics for commands  50
  graying out menu item labels  48
  greater than DIESEL function  66
  greater than or equal to DIESEL function  66

H
  hatch patterns
    about  25
    creating  25
    examples  25, 27, 29
    multiple lines  29
  hatches
    creating  25
    customizing  25
  Help
    locating Help files  9
    tooltip command Help  49

I
  icons
    customizations  9
  if DIESEL function  70
  images for commands  50
  index DIESEL function  70
  input
    pausing macros for  42
    prompting for  47
  installation
    plug-in applications  87
interface
  Customize dialog box  33
  customizing  33
terminology  34
interface elements
defined  34
interface items  34
international character sets  44, 167, 168

J
Japanese fonts  167, 168

K
Kanji fonts  167, 168
keyboard input in macros  42

L
languages
  Big Fonts  168
  macro translations  44
  special character codes  167
less than DIESEL function  65
less than or equal to DIESEL function  66
library search path  3
LIN files (linetype library)  9, 15
linetypes
  about  15
  complex  22
  creating  16
  customizing  15
  examples  16
  including shapes in  22
  simple  16
  text characters in  19
loading
  AutoLISP applications  78, 80, 81, 82, 83
  ObjectARX applications  86
  plug-in applications  88
localization  100
LSP files (AutoLISP)  78

M
macros
  about  38
  AutoLISP expressions in  46, 61
  commands in  44
  conditional expressions in  45
defined  34
  DIESEL expressions in  45, 61
  international language support  44
  pausing for user input  40, 42
  prompting for user input  47
  repeating commands  44
  scripts  73
  selecting objects with  45
  special character codes for commands  40
  suppressing echoes and prompts  40
  syntax  38
  terminating  39
MDE (Multiple Design Environment)  78
menu items
  DIESEL expressions and  63
  disabling  48
  graying out  48
menus
  AutoLISP MNL files  83
  creating  50, 51
  creating sub-menu  51
  customizing  50
  international language support  44
  menu bar  51
  submenus  51
MNL files (menu LISP)
  loading  80, 83
  macros and  46
modifying
  commands  35
Multiple Design Environment  78
multiple-line hatch patterns  29
multiplication DIESEL function  65

N
namespaces in Visual LISP  78
naming
  linetypes 16
  shapes 107
not equal to DIESEL function 66
nth DIESEL function 70

ObjectARX applications
  about 86
  loading 86
or DIESEL function 71

package folder structure 101
package installation 87
PackageContents.xml file 88
palettes
  defined 34
PAT files 9
patterns
  linetypes 17
pausing macros 40, 42
PFB files (printer font binary) 106
PGP files (program parameters) 9
plot styles
  locating files 7
plotter configuration files
  locating files 8
plug-in applications 87
PostScript fonts 106
prefixes for commands 14
profiles 7
program file locations 3
programming applications
  AutoLISP 78
  ObjectARX 86
  plug-in applications 87
  Visual LISP 78
prompts
  suppressing 40
  user input in macros 47
properties
  commands 34
PSF files 9

rearranging
  tools 55
repeating commands 44
ribbon
  command images 50
roaming profiles 7
rtos DIESEL function 71
RuntimeRequirements element 95

S
S STARTUP AutoLISP function 84
SCR files (scripts) 73
scripts
  about 73
  file names and 74, 75
  running at startup 75
  switches and 75
search paths
  default file locations 3
selecting
  objects with macros 45
shape definition files
  about 105
  Big Font files 167
  creating 105, 106
  samples 118, 147
  superscripts and subscripts 177
  text fonts 117
  Unicode fonts and 176
shape fonts
  about
    Big Fonts 167
    sample files 118, 147
    superscripts and subscripts 177
    Unicode shape definition files 176
shape specification bytes 110
shapes
  about
    descriptions 107
    including in linetypes 22
    shape specification bytes 108
    special codes 110
SHP fonts (shape definition) 105, 106
SHX fonts 9, 105, 106, 177
spaces in linetypes 16
special characters
  commands and 51
  macros and 38, 40, 51
  shape fonts
startup AutoLISP errors 83
startup scripts 75
strlen DIESEL function 71
submenus 51
subscripts in fonts 177
substr DIESEL function 71
subtraction DIESEL function 65
superscripts in fonts 177
support files
  locations 3
  suppressing echoes and prompts 40
switches
  running scripts and 75
symbols
  commands and 51
  macros and 38, 40, 51
  shape fonts
syntax
  macros 38
  shape fonts 107
  Unicode shape definition files 176
system variables
  AutoLISP functions and 78
  toggling values with macros 46

T
templates
  locating files 12
terminating macros 39
text
  including in linetypes 19
  shape fonts
text styles
  Big Fonts and 174
textures
  locating files 13
time
  edtime DIESEL function 67
  tool groups
    customizing 54
    defined 34
    drop-downs on 56
    expanding 55
    flyouts on 56
  tool sets
    creating 55
    customizing 54
    defined 34
    drop-downs 56
    flyouts 56
  Tool Sets palette 55
toolbars
  command images 50
tooltips
  command Help in 49
  translating macros 44
  translating plug-in applications 100
  transparent commands 42
  Type 1 PostScript fonts 106

U
Unicode fonts 147, 176
Unicode shape definition files 176
UNT files 9
upper DIESEL function 71
user input
  pausing macros 42
user interface
  Customize dialog box 33
  customizing 33
  terminology 34

V
vectors in shape specifications 108
Visual LISP 78
VLISP (Visual LISP) 78

X
xor DIESEL function 72