Overview of AutoLISP Automatic Loading

AutoCAD loads the contents of three user-definable files automatically: acad.lsp, acadlisp.lsp, and the MNL file that accompanies your current customization file. By default, the acad.lsp file is loaded only once, when AutoCAD starts, whereas acadlisp.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 acadlisp.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. 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 acadlisp.lsp startup files are not provided with AutoCAD. It is up to the user to create and maintain these files.

Command Autoload

When you automatically load a command using the load or command functions, the command’s definition takes up memory whether or not you actually use the command. The AutoLISP autoload function makes a command available without loading the entire routine into memory. Adding the following code to your acadlisp.lsp file automatically loads the commands CMD1, CMD2, and CMD3 from the cmdls.lsp file and the newcmd command from the newcmd.lsp file.

```lisp
(autoload "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 autoautoload function for ObjectARX applications. See autoload and autoautoload in the AutoLISP Reference in the Help system. To display Developer Help, on the InfoCenter toolbar, to the right of the Help button, click the drop-down arrow. From the menu, click Additional Resources >> Developer Help.

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).

Quick Reference

- Commands
- System Variables
The ACAD.LSP File

You can create an acad.lsp file if you regularly use specific AutoLISP routines. When you start AutoCAD, 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 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 acadadd.lsp file.

The recommended functionality of acad.lsp and acadadd.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, database.synch.lsp, and drawingmanager.lsp are loaded every time you start AutoCAD.

```lisp
(load "mysessionapp1")
(load "database.synch")
(load "drawingmanager")
```

Warning Do not modify the reserved acad2011.lsp file. Autodesk provides the acad2011.lsp file, which contains AutoLISP defined functions that are required by AutoCAD. This file is loaded into memory immediately before the acad.lsp file is loaded.

Quick Reference

- [Commands](#)
- [System Variables](#)
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 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 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 acad2011doc.lsp file. Autodesk provides the acad2011doc.lsp file, which contains AutoLISP-defined functions that are required by AutoCAD. This file is loaded into memory immediately before the acaddoc.lsp file is loaded.

Quick Reference

- Commands
- System Variables
The MNL File for an AutoLISP Menu

When AutoCAD 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 loads the AutoLISP functions that are needed for proper operation of a menu.

This function ensures that AutoCAD loads the AutoLISP functions that are needed for proper operation of a menu. For example, the default AutoCAD customization file, acad.cui, relies on the file acad.mnl. This file defines numerous AutoLISP functions used by the menu. The MNL file is loaded after the acad.doc/sp file.

Note: If a customization file is loaded with the AutoLISP command function—with syntax similar to (command "menu" "new-menu")—the associated MNL file is not loaded until the entire AutoLISP routine has run.

In this example, calls to the prime function can be used to display status messages. The first use of prime displays the following at the command prompt:

```
Newmenu utilities... Loaded
```

The second call to prime exits the AutoLISP function. Without this second call to prime, the message would be displayed twice. As mentioned previously, you can include the onfailure argument with calls to the load function as an extra precaution.
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 library path generally cause errors. Therefore, you may want to use the onfailure argument with the load function. The following example uses the onfailure argument:

```
(princ (load "mydocapp1" "\MYDOCAPP1.LSP file not loaded."))
(princ (load "build" "\BUILD.LSP file not loaded."))
(princ (load "counter" "\COUNTER.LSP file not loaded."))
(princ)
```

If a call to the 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 onfailure argument. In the preceding example, the value returned by 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 loads the mydocapp1.lsp file, the princ function displays the following message and AutoCAD continues to load the two remaining files:

```
MYDOCAPP1.LSP file not loaded.
```

If you use the command function in an acad.lsp, acaddoc.lsp, or MNL file, 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.
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, acad.doc.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, acad.doc.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 LISP 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 acad.doc.lsp file that contains the following:

```lisp
(defun C:HATCH ()
  (alert "Using the BHATCH command!
"
  (command "BHATCH")
  (princ)
)
(defun OLDHATCH ()
  (command ".HATCH")
  (princ)
)
(defun q S::STARTUP ()
  (command "undefine "HATCH")
  (princ "\nundefined HATCH to BHATCH\n")
)
```

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 ...
)
```

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.
Select these tabs, or check boxes in the dialog box to prevent multiple instances of the Acad.lsp from opening.