



Autodesk
University
2007

Migrating from MNU to the CUI

R. Robert Bell – Sparling

CM219-1P

The Customize User Interface (CUI) radically changes the customization of the user's environment. Along with the new dialog interface is the retirement of your MNS/MNU files. Do you struggle with the concepts of workspaces affecting the user interface; maintaining the enterprise CUI to support company standards; customization options from the main CUI; and the use of the CUI when editing the interface? If you answered "yes," this class will help you put all the pieces together.

About the Speaker:

Robert is the Design Technology Manager for Sparling, the largest specialty electrical engineering and technology consulting firm in the United States, located in Seattle Washington. He provides strategic direction, technical oversight, and high-level support for Sparling's enterprise design and production technology systems. He is instrumental in positioning Sparling as an industry and client leader in leveraging technology in virtual building and design. Robert has been writing AutoLISP[®] code since the release of AutoCAD v2.5, and VBA since introduced in R14. He has customized applications for the electrical/lighting, plumbing/piping, and HVAC disciplines. Robert has also developed applications for AutoCAD as a consultant. A former member of the Board of Directors for AUGI[®], he is active on AUGI forums and Autodesk discussion groups.
rbell@sparling.com



Autodesk
University
2007

Introduction

The Customize User Interface (CUI) is a radical approach to customizing AutoCAD. The changes are enough to overwhelm many CAD managers comfortable with the menu approach. This course is intended not only to introduce how to use the CUI Editor but to discuss the possibilities for organization and migration issues.

Some CAD managers may be tempted to ignore the CUI and continue to maintain their menu files, allowing the files to be converted to CUI files, and simply ignoring the CUI interface. This approach, although compelling in its simplicity, ignores the newer interface features at the expense of your users.

Terminology

CUI (Customize User Interface)

- 1: The commands and elements, including workspaces, that makes up the user's interface.
- 2: The extension for the XML-based file that contains the information used to present the user interface.

Main

The user customizable CUI file defined in the Options dialog box.

Enterprise

The CUI file defined in the Options dialog box that is not user customizable.

Partial CUI

A CUI file that is loaded into the main or enterprise CUI file.

Core CUI

The CUI file that supports the application out of the box, e.g. Acad.cui.

Office CUI

A term used in this course to describe the CUI file that holds office standard customizations.

Nodes

Major sections of a CUI file, such as Toolbars or Mouse Buttons.

Workspace

Defines visible elements of the user interface and their location.

Commands

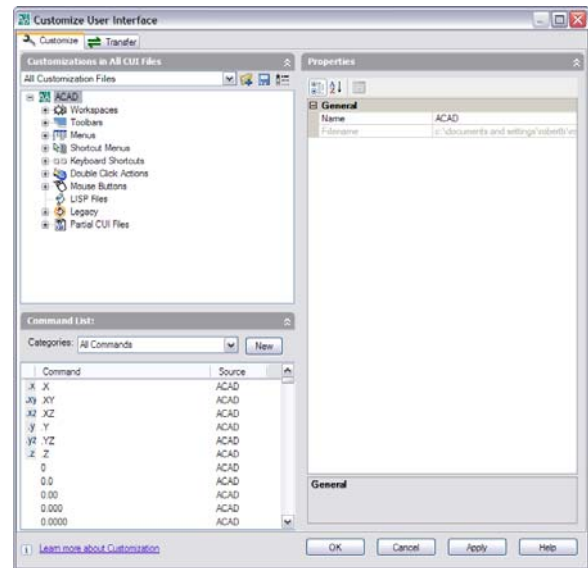
Macros that define an action and may be used throughout the nodes.

OOTB

Out of the box.

Vertical

A version of AutoCAD customized by Autodesk to support a specific industry, such as AutoCAD Architectural or AutoCAD MEP.



Migration Basics

Migration occurs automatically when a menu file is loaded. If you have kept your menu files separate from the core menu files, migration can be that simple. The migration will be more difficult for firms that have modified the core menu files directly. The issue is that there will be many duplicated commands between the core CUI and the office CUI files. The best option in this case is to convert your legacy menu into a temporary CUI and then transfer the needed elements into a new CUI file.

The CUI file dialog boxes can be changed to show legacy menu files. You load the menu file and it is converted into a CUI file. The CUI file will be used after the conversion, not the original menu file. You

may want to keep the menu files in an archive, but they will rapidly become out-of-date as you utilize the CUI.

Some preparation of your menu files before you attempt to convert them can reduce the potential for a bad migration. Also, the latest versions of AutoCAD have corrected many issues that occurred in the first versions of the CUI.

Correcting Migration Issues

The following items are common issues that can be quickly corrected. Take a few minutes with your menu files and profile to verify that none of these issues will affect you.

- You need to make sure the icons used by the legacy menu are found on AutoCAD's search path before you migrate the menu.
- A very common error is to have an incorrect menu group name. The menu group name is not intended to duplicate the filename of the menu. A menu group name is limited to 32 characters and cannot contain spaces or punctuation marks.

Incorrect: `***MENUGROUP=C:\Acad Customizations\Menu\My Special Menu.mnu`

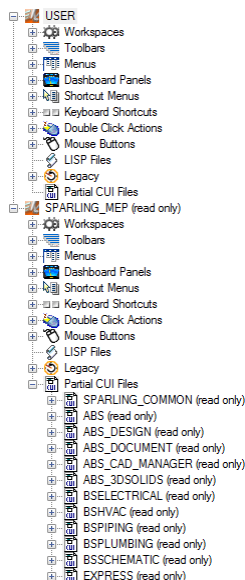
Correct: `***MENUGROUP=MyMenu`

- Migration of legacy files will work well if your menu files do not contain errors in them. One way to reduce errors is to use the `.mns` file to perform the migration instead of the `.mnu` file. If a migration fails, examine the source file for errors.
- You cannot have different CUI files with the same name (menu group name) loaded at the same time.
- Verify that the same ID is *not* used for different menu macros.

CUI Organization

The CUI introduced the concept of main and enterprise elements of the user interface. This split in the interface has confused many CAD managers. On the surface, it may seem possible to ignore the enterprise feature and simply use the main CUI and its partials in a manner similar (and let's admit it, familiar) to the legacy menu system. However, this approach will quickly reveal limitations that are not immediately obvious. This class will cut thru the confusion and demonstrate a structure that many firms use.

Before the CUI organization is discussed in detail, take a few minutes to understand the differences and similarities between the types of CUI files.



The differences between Main, Enterprise, and Partial

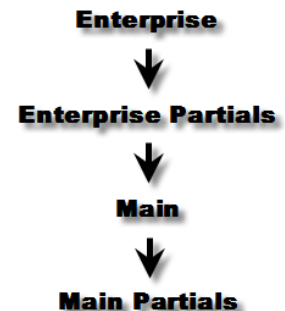
- A partial CUI file is one that is loaded into either the main or enterprise CUI files. These may be loaded from the CUI Editor or with the `CUILoad` command.
- A partial CUI file can actually be a main or enterprise CUI under a different profile.
- A partial CUI file may have many nodes populated, or only a few.
- The main CUI file, and any partial CUI files loaded into it, are the only ones that you can modify.

- The enterprise CUI file, and any partial CUI files loaded into it, are read-only in the current profile. When the same CUI file is loaded as a partial to both the main and enterprise CUI files it will be read-only.
- An enterprise CUI file can be edited by modifying the Main Customization File entry in the Files tab of the Options dialog box.
- Workspaces in the main and enterprise CUI files are available, but workspaces defined in partial CUI files are not.

Load Order and Precedence

Enterprise loads first. Any partial files in the enterprise CUI file will load next. Main then loads and any partial CUI files contained by it will load last. The order displayed in the Partial CUI Files node is the order of loading.

The last loaded definition of keyboard shortcuts, whether shortcut keys or temporary overrides, will be the one that takes effect. Double-click actions also work in the same manner. However, for all other CUI elements such as menus and mouse buttons the instance loaded first will be the definition used.



What file should be used for the main CUI?

The approach you take will depend on what you want to accomplish and what is important to you. However, it is assumed that you don't want your users modifying the core or office CUI files. Therefore, it becomes obvious that you don't want the core or office CUI files to be the main CUI file.

Of key importance is the fact that the enterprise CUI file, and any partial CUI files loaded into it, is read-only to the user. Please note that although the CUI Editor makes the enterprise CUI file read-only, all the user would need to do to edit it is change their options. Therefore, you still need to have network permissions assigned to protect the enterprise CUI file. You need to provide a main CUI file in any case, which means the user can modify it.

Workspaces also play a large role. Workspaces in the main CUI file can be modified by the user. Workspaces in the enterprise CUI are available to the user, but they cannot be edited. Workspaces that are defined in CUI files that are partial to either main or enterprise are not available.

What file should be used for the enterprise CUI?

There are several options for the enterprise CUI file. In fact, there are so many options that this is what often overwhelms the CAD manager. Consider the following items to help you make a decision.

- Does your firm have standard tools that have been created for your users, that work the same regardless of the vertical being used?
- Does your firm need to support multiple verticals?
- Does your firm have customizations that are specific to a vertical?
- Does your firm need to prevent users modifying the company standard tools or the core files?

Given the above criteria the choice becomes clear. You have three options: the core CUI file, an office standard CUI file, or an office vertical CUI file. When you need to support multiple verticals there is really only one choice: an office vertical CUI file.

Why use an office vertical CUI file?

Workspaces are what drive this decision. Workspaces that are in the main or enterprise CUI file are the only workspaces available. If you elect to use the core CUI file as enterprise, this would mean that you need to modify the core CUI file to store the workspaces. This, of course, violates the idea of keeping the core CUI file inviolate. It also does not make sense to use a “vanilla” office CUI file to store the workspaces, since one vertical’s workspaces will not be sensible to another vertical.

Unfortunately, this means that your CUI structure will be more fractured than you may be used to, compared to the legacy menu structure. Here is the basic picture.

- You have a vertical-flavored enterprise CUI file which has, as partials:
 - The core CUI file
 - The office common CUI file
- The user will have their own CUI file as main:
 - Be it a blank CUI file, or
 - One that is migrated from a previous version
 - Their main CUI file will be vertically-flavored

If you only need to support one “flavor” of AutoCAD, consider yourself fortunate. This means you can make your office CUI file the enterprise CUI file and just the core CUI files as partials to it.

File Location

Now that you have a better idea of what files you need, it is time to determine the best location of the files. Obviously, the office CUI files need to be located on the network to make it easy to update them. There is no issue in locating them local, if you have some mechanism to push the updates to the local machines.

The OOTB CUI files can be located either in their OOTB installed location or up on the network. When they are located locally there are fewer files that need to be opened on the network. However, clever users might find ways to edit the OOTB content. That is not as large an issue as it may seem, since the OOTB files are easily reset to their original content (see below). Locating the OOTB CUI files on the network can help to protect them if you run into frequent issues with the OOTB content getting messed up by your users.

As long as the location of the OOTH CUI files is on the support paths for AutoCAD, the CUI files attached as partial *do not include path information*. So there is no problem in using the Acad.cui file located in your user profile when creating your structure since the path information will not be stored.

The main CUI file, if it is the user’s file, needs to be located either in a roamable folder in the user’s profile (the default Support folder meets the criteria) or on the network in a location where they have rights to modify files. If you do not have roaming enabled, there is a distinct advantage to locating the user’s files on the network. This enables them to log in on any computer on the network and have their usual user interface. *Note: supporting login from any workstation involves careful planning and testing beyond simply locating files on the network. Issues such as logging into a computer for the first time need to be addressed.*

Backup

On occasion, you may find that you need to reset the original CUI files that were installed with AutoCAD. The original CUI files are available in the `C:\Program Files\AutoCAD`

Version>\UserDataCache\Support folder. *UserDataCache* is a hidden folder. So there is no need to reinstall AutoCAD simply to reset the original CUI files. AutoCAD 2007 introduced a couple of options under the CUI editor's shortcut menu to make it easier to restore a backup or reset a core file. The first option restores the automatic backup of the CUI file saved in the same location as the CUI file, and the second option resets (or replaces) the CUI file with the originally installed CUI file, if applicable. If the file in question cannot be reset, that option will be disabled. Similarly, if a .bak file does not exist, the Restore option will be disabled.

It is a good idea to have your own backup of your CUI files before you edit them. One way to do this is to simply keep a zip file in the same folder as the CUI files and drop the CUI files into the zip file before editing. You should endeavor to keep a historical series of backups, in case a problem doesn't manifest itself immediately.

Create a workspace to store the interface so that you can retrieve it if needed. This will be covered in more detail in the Workspaces section.

The Editor

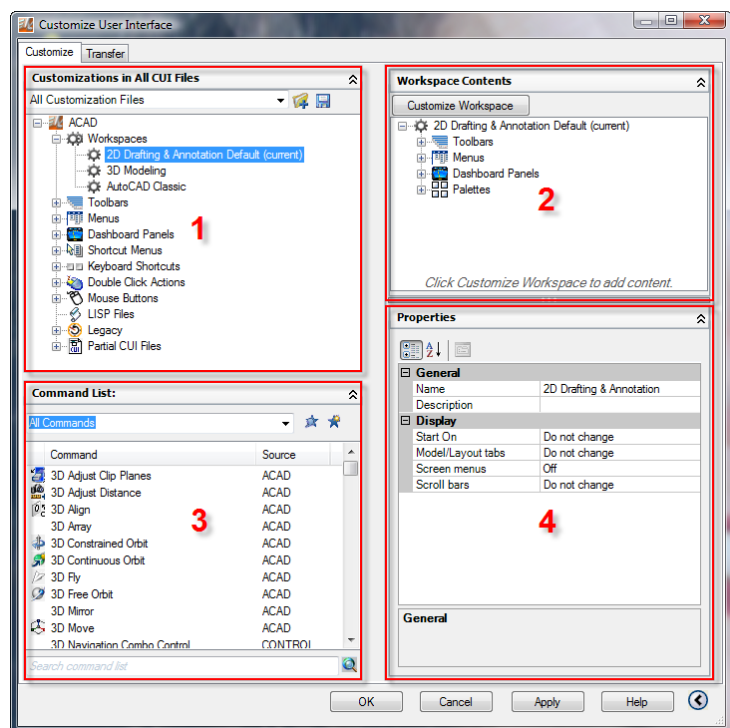
The editor contains two tabs, the *Customize* tab and the *Transfer* tab. You will spend the majority of your time in the *Customize* tab, but the *Transfer* tab is very useful on occasions. Take a few moments to identify the four major panes in the *Customize* tab.

The pane in the upper left is the *Customization In Pane*. This pane displays the nodes of the CUI selected in the pane's combo box. Usually this combo box is displaying all customization files, but you may select the main CUI structure, the enterprise CUI structure, or a specific partial CUI file.

The pane in the upper right is called the *Dynamic Display Pane*. This pane changes depending on the element selected in the *Customization In Pane*. New to AutoCAD 2008 is the ability to select buttons on the toolbar preview when you have a toolbar element selected. When you select a command in the *Command List Pane* the *Dynamic Display Pane* closes.

The *Command List Pane* displays all the loaded commands. This list is filtered by the selection in the *Customization In Pane*'s combo box, and furthered filtered by its own combo box. All the commands you create will be lumped into the Custom commands category, so it can become necessary to filter the list with the *Customization In Pane*'s combo box as your list of commands grows (as long as the commands are in separate CUI files, obviously).

Finally we come to the *Properties Pane*. This pane will display the properties of the selected element or command.



Find and Replace

There is a find and replace feature available in the CUI editor. However, it takes a right-click to make the feature evident. Alas, the feature was obviously written by a programmer. There is far too much “tech-talk” in the results. However, the following information might make the obscure language easier to interpret.

For example, do the following:

- Open the Acad.cui file in the CUI editor
- Right-click on either of the left panes in the Customize tab
- Select the Find... item. The “Find and Replace” dialog box displays.
- Select the “Find what” combo box and type “etransmit”.
- Make sure the “Ignore case” check box is checked and clear the “Restrict Search to” check box.
- Select the “Find Next” button.

The panes will change to the location in the tree of the item found and/or the command and its properties.

When the found item is something located in the structure of the CUI, as opposed to the command only, the “Find and Replace” dialog box displays text similar to this: Search string found in tree node ‘eTransmit...’ property ‘Command Name’ at position 0 (1/9).

“Tree node” means that the item found belongs in the *Customization In Pane* and is part of the CUI structure. Also, the command used by the item is highlighted in the *Command List Pane*. Usually it is the command that feeds the item in the tree, but a bug occurs when you loop the same search back to the beginning of the matches. In that case, an incorrect command is displayed for items found in the tree.

When a command only is the item found the result reads “command” instead of “tree node”.

“Property” refers to the *Properties Pane* and the name of the property is given.

“At position” is talking about where in the property’s data the string is located. Unhappily for mere mortals, the number given is the character count, but where the first character is 0, the second character is 1, and so on. So position 36 means the string starts at character 37 in the property. (This is called a zero-based index, popular with programmers, and nobody else.)

Finally, the numbers in the parenthesis indicate the current match and the total number of matches, e.g. (1/9) is the first match of 9 different matches.

Working with Workspaces

Although you can modify the CUI without using workspaces, you *must not* ignore them. At the very least, you should create a workspace of your CUI as a default. As you edit the CUI, it is far too easy to alter the interface unintentionally such as removing menus from the menu bar. Migrating from a legacy menu that used menu swapping will display all the pull-down menus on the menu bar, so a workspace is needed to control the visible menus. In addition, menus added at later periods in time will often not display automatically on the menu bar. Therefore, you need to use workspaces to manage those menus.

Workspaces may be stored in any CUI file, however, only the main and enterprise CUI file’s workspaces are available. The enterprise CUI file is the perfect place to create workspaces that need to be available on every user’s workstation.

Right click on the Workspaces node and select New ▶ Workspace. A new workspace has some unfamiliar properties.

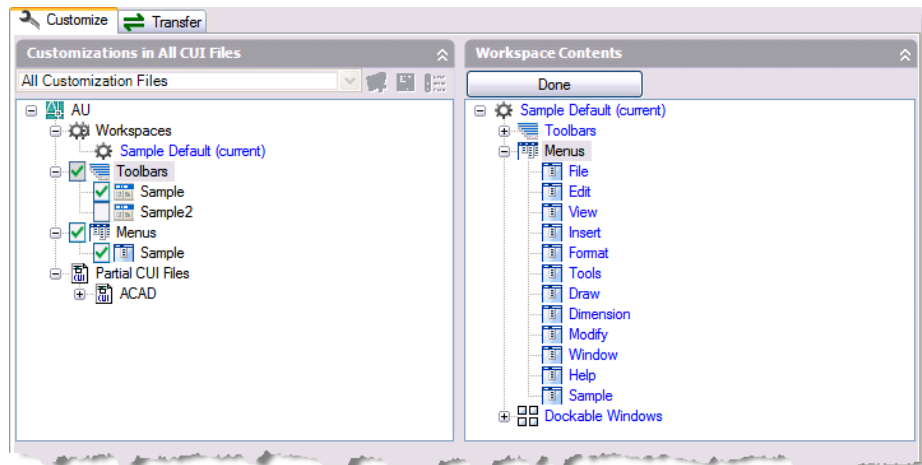
- Start On: This permits you to switch the active layout when you select the workspace.
- Model/Layout tabs: Choose to display or hide the layout tabs.
- Screen menus: Forget it. Lose screen menus.
- Scroll bars: Why does anyone use scroll bars when we have real-time pan/zooms?

Workspaces also control Dockable Windows, such as the Properties palette. Many of properties for these windows can be set to “Do not change”. This is perfect in the case of workspaces assigned by the enterprise CUI file. When you set the properties to that option the workspace will not change that property from what the user has set. It is a good idea to dock toolbars that are controlled by an enterprise workspace. If they are set to float, the user will get floating toolbars every time that workspace is made active. In addition, floating toolbars depend on the resolution of the screen, rather than the application window size.

Positioning and locking toolbars with both a main and enterprise CUI file is still problematic.

Note: If workspaces are only defined in the enterprise CUI file initially, there can be an error when the user attempts to save their first workspace to the main CUI file. You have an option to avoid the error: create the workspace in the CUI Editor.

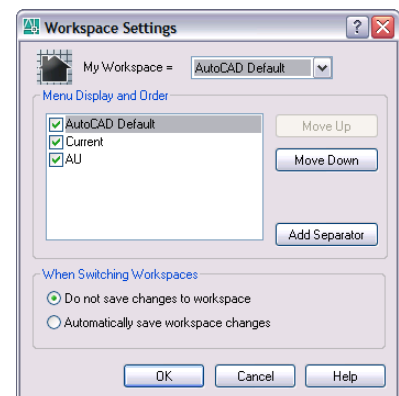
You change the elements in the workspace using the Customize Workspace button in the Workspace Contents pane. The elements in the pane turn blue. The elements in the Customizations pane display with check boxes. Use those check boxes to add elements to the workspace. Change the display order of the chosen elements in the Workspace Contents pane using drag-and-drop.



The Workspace toolbar provides access to all your workspaces and the general settings for workspaces (the left button). This toolbar provides an excellent mechanism to switch the interface instead of using a pull-down menu to swap other pull-down menus.

Unhappily, the Transfer tab is not useful for copying just workspaces. The issue is that the items that the workspace uses are also transferred. This means that, even if the source CUI is partial to the target CUI, all the items are duplicated in the target CUI file. This will lead to odd behavior in some of the vertical applications. When a vertical application swaps pull-down menus for discipline-specific menus, you will wind up with duplicated menus on the menu bar.

So the best approach is to take a screen shot of the source CUI’s user interface with the workspace active, and create a new workspace in the target CUI file that duplicates the desired user interface.



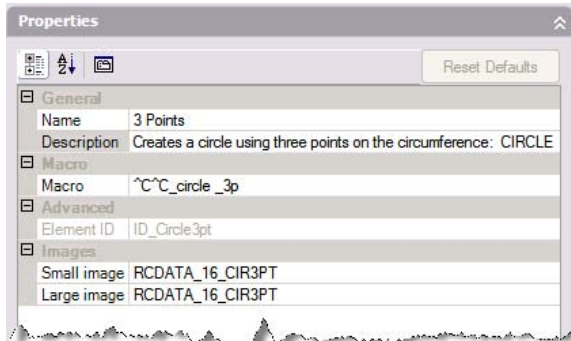
It all starts with Commands

One of the key differences of the CUI approach is how commands are obviously used in multiple places. A single command might be used by a toolbar, a menu item, and even a shortcut key. Commands are ultimately identified by their element ID. It is possible to have different commands with the same name, even in the same CUI file. Therefore, the element ID is important to uniquely identify different commands.

One command, multiple locations

A sticking point for some migrated legacy menus arises because the one customizing the new CUI file believes that they can edit the same command used in multiple locations without affecting the other instances.

*One single command (element ID) may be used in **multiple** locations and editing the command in **one** location will **affect all** the other locations*



You can filter the list of commands. First, select just the main CUI file in the Customizations pane. Next, select Custom Commands in the Categories drop-down list in the Command List pane. Control elements may still appear in the command list. If so, simply select Custom Commands again.

New commands will be created in the CUI file selected in the Customization In Pane combo box. It is possible to create your commands in any of the partial CUI files attached to the main CUI file.

Note that images now also display on pull-down menus. Remember, when you modify a command, all the items are updated that use that command.

A command's name and description are not necessarily limited in length. However, it would be best to be reasonable. The description is not only visible in the CUI Editor but displays in the status bar.

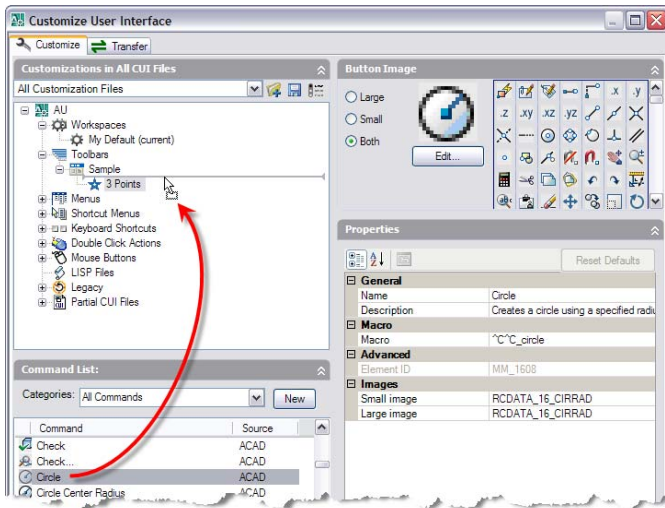
The macro itself is unchanged from the legacy menu macro. All the control characters still work as they always have. For more information on control characters, see the Customization Guide.

Images are attached to the command itself. This makes them available in the pull-down menus in addition to the toolbars. The source files for the icons may be individual BMP files or stored collectively in a resource .dll file. The icon files must be placed on AutoCAD's search path. The size of the bitmap for a small image is 16x16 pixels. A large image needs to be 32x32 pixels.

Remember, if you want to create a new command in a partial CUI file you must filter to just the partial CUI file in the *Customization In Pane* combo box. There is a way to copy commands by using a quirk in the editor's interface. Create a scratch CUI file and attach it as partial to the main CUI file. Create an element to store the commands, such as a toolbar or menu. Now, drag multiple instances of the desired command to the new element, one after the other, until you have the desired number of copies. Now select OK to dismiss the editor. Immediately run the CUI command again. Select the scratch CUI file and note that you will have separate instances of the command (notice that the element ID is different for each). You may now edit those copies independently of each other.

Standard Toolbars

Creating a new toolbar is as simple as right clicking on the Toolbars node and selecting New ♦ Toolbar from the menu. The new toolbar is automatically given an element ID that cannot be changed. Note that toolbars may have more than one alias. Aliases are useful for programs that need to affect the display of the toolbar. However, as you will later see, workspaces dispense with much of that effort.



Once the toolbar has been created, it is simply a matter of drag-and-drop to place commands on the toolbar. Select a command from the

Tip
 Drag the command in a wide arc to the right and come in from the side to the location in the structure. This helps avoid the structure scrolling on you.

Command List pane. Drag the selected command up to the desired toolbar. If the toolbar is currently empty, drop the command on to the toolbar name. Otherwise, drop the command in the desired location amongst the current commands on the toolbar. Reorganize the order of the icons on a toolbar by dragging-and-dropping the commands into the desired order.

The natural inclination, when you select a command on the toolbar list, is that you can modify the command for just that instance on the toolbar. Once again, however, you must recognize that the command may be used in multiple locations.

If you drag a command from a partial or enterprise CUI file a copy of the command is created in the target CUI file. In order to see the copied command you need to exit the editor and reenter the editor. This issue will be resolved in the future.

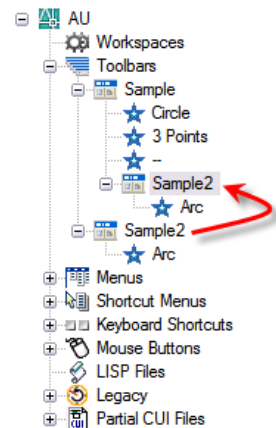
Flyout Toolbars/Buttons

The terms “flyout toolbar” and “flyout button” are essentially interchangeable, however, newer documentation seem to use the term flyout toolbar more often. This course will also use that term. This is because the item for the flyout is represented as a nested toolbar.

The nested toolbar is like an XRef of a source toolbar. Edits made to the nested toolbar are actually changing the source toolbar.

There are two ways to create a flyout toolbar.

To create a flyout toolbar based on an existing toolbar you drag the source toolbar into the desired location on *an expanded* toolbar.



You may also create a flyout and new source toolbar in one action. Select an existing toolbar. Right-click and select New ▶ Flyout from the menu. This will place a new toolbar in the root of the node. A “referenced” toolbar is placed within the selected toolbar.

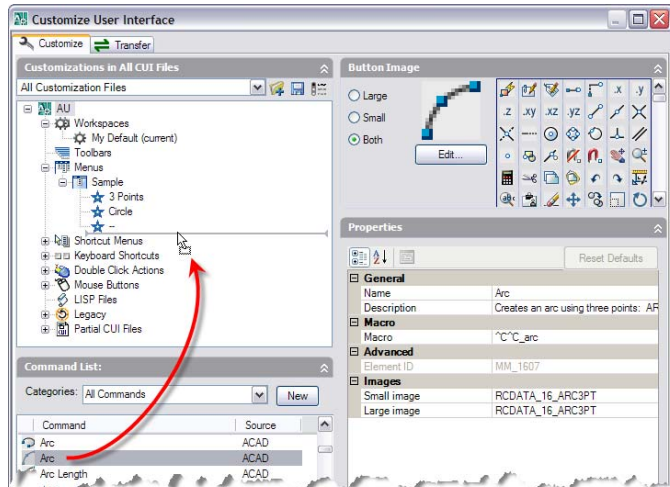
Toolbar aliases
 Never change the initial alias given the new source toolbar for the flyout. If you do, the button on the toolbar where the flyout appears will not be able to find that source toolbar.

A flyout toolbar has a unique property: Use Own Icon. This is how you instruct the flyout either to use the icon of the last selected tool or always use the same icon. If you choose to use its own icon, note that the button preview of the icon does not update until you select image files. If you do not select image files, the icon will become the “missing question mark” icon, regardless of what you see in the button image area.

If you change the icon for a flyout toolbar and only hit the Apply button, you can see the correct icon displayed on the actual toolbar, if it is visible.

Pull-down Menus

Create a new pull-down menu by right clicking on the Menus node and selecting New ♦ Menu. Give it a new name. Aliases are useful for control by programs.



Commands are added to the menu in the same fashion that they are added to toolbars, with a drag-and-drop operation. This is, of course, only a reference back to the actual command. Any changes you make to the apparent menu item are being made to the command. Therefore, if that command is referenced anywhere else, the changes affect all instances.

However, this does not mean that you cannot control the display of the menu items. You can still disable items and/or checkmark them.

To disable a command on the pull-down menus, place a tilde (~) at the beginning of the command name. This will not disable toolbar buttons where this command is used.

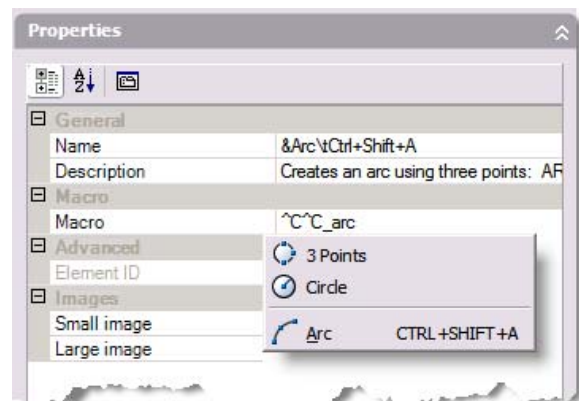
Menu commands can also be marked with a check mark or a border. A check mark will be displayed where a command does not have an image. Commands that do have an image will receive a border around the icon. Mark commands by placing an exclamation point and a period (!.) at the beginning of the command name.

Visual LISP can affect the menu item label thru the use of the *menucmd* function. DIESEL commands still operate correctly in a command name to affect the display of the pull-down items. DIESEL expressions are supported in the command name. For example, the following command name on the pull-down menu is grayed out when a command is active:

```
$(if, $(getvar, CmdActive), ~)3 Points
```

The menu access key is defined by placing an ampersand (&) in front of the character desired. This will give the user the ability to execute the menu item by using Alt+<character>.

You may also move part of the label to the right side of the menu by using “\t” in the command name.



Dashboard Panels

The Dashboard palette, introduced in AutoCAD 2007, is now customizable in AutoCAD 2008. The dashboard can contain both commands and controls. Each panel in the dashboard has a “button” associated with it. This “button” is simply an icon that is visible on the left side of the panel. Aside from that, the procedure of populating the panel is the same as populating a pull-down menu.

Shortcut Menus

Shortcut menus are a commonly edited feature. However, they can also be one of the trickiest parts of the CUI editing and setup. If your main (or enterprise) CUI file does not have shortcut menus then you need to copy *all* the shortcut menus to the main CUI file and “wake” AutoCAD to the new additions. The Transfer tab makes this easy.

For example, assume you want to add a shortcut menu for line objects. Create a new shortcut menu. Add the alias "OBJECT_LINE" to the new menu. This will cause the menu to be added to the Edit menu when a line object is gripped. If you want the menu to appear when multiple lines are selected use the alias "OBJECTS_LINE". Add whatever commands you desire to the new menu.

If you test your changes at this point, they would not appear. AutoCAD is still using the original context menus and will continue to do so even if you restart AutoCAD. To "wake" AutoCAD to the new context menu you need to detach the Acad.cui file and test your changes. They should appear at this time. Reattach the Acad.cui file and your edits will still be in effect.

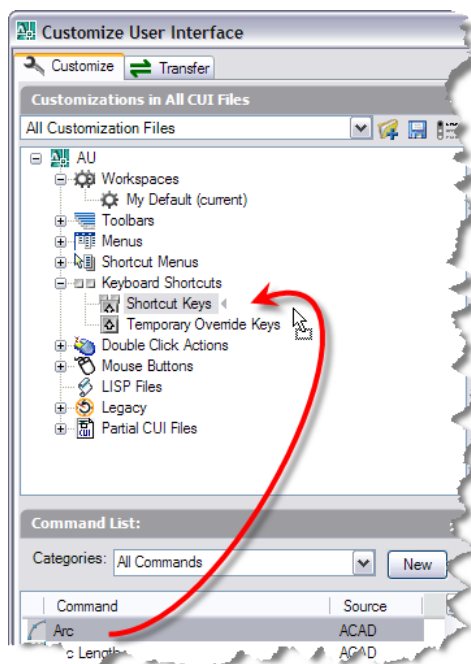
If you customize an OSnap menu, and you find that it is available only intermittently, you can use the following code in that CUI's .mnl file to make it more reliable. (Replace "AU" with the MenuGroup name of your CUI file, and "MyOSnaps" with the alias of your shortcut menu.)

(menucmd "P0=AU.MyOSnaps")

Keyboard Shortcuts

There are two types of keyboard shortcuts. The definitions in the Accelerator section from a legacy menu are now called Shortcut Keys. This is where you would make keystroke-based commands such as F4 using the EndP OSnap. Temporary Override Keys are a new feature since AutoCAD 2006. For example, you would use the Shift key during a command to temporarily toggle Ortho, or Shift+A to disable running OSnaps.

Shortcut Keys

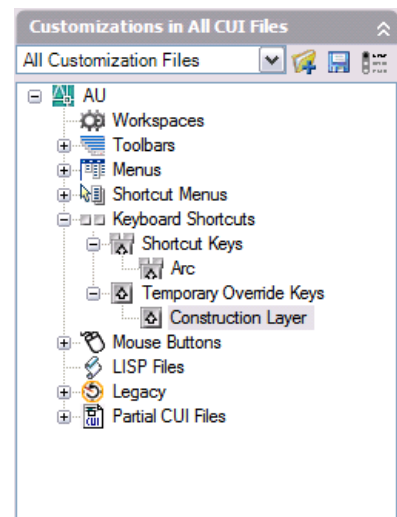


You *need* to use drag-and-drop to add a new shortcut key. If you right-click on the Keyboard Shortcuts or Shortcut Keys node and select New, shortcut keys are not an option. However, to create a new shortcut key all you need to do is drag-and-drop a command on to the Shortcut Keys node. You then edit the Key(s) property to assign a keystroke combination to the command. Take note of whether the keystroke combination already exists.

At times, you want to overwrite the current definition of a shortcut key. Remember that the last assignment of the keystroke combination is the one that takes precedence. Therefore, the CUI file that holds your overrides will need to load after the CUI files that contain the original definitions. This is an improvement over legacy menus, as the initial definition was the one that took precedence. Depending on the CUI file structure you have, you may decide to have a separate CUI file for just shortcut keys!

There are a few keystroke combinations that you are unable to enter in the CUI Editor, e.g. Ctrl+F1, Ctrl+F4, Ctrl+F6, and Ctrl+F10. However, a migrated .mns file with those accelerator keys defined will import correctly. Import a temporary .mns file and transfer the definitions if you want to make those definitions in the CUI Editor.

Tip: Turn Caps Lock off before assigning a keystroke combination, otherwise a Shift is added automatically. This allows you to create a new definition for F1!

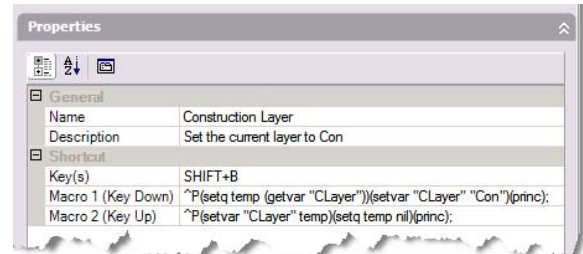


Temporary Override Keys

Temporary overrides provide a toggle while commands are active. While the keystroke combination is being pressed, the override is in effect. It is possible to make the override perform another macro when the keystroke combination is released.

Most of the typical overrides do not need the Key Up macro to reset the original condition overridden. For those cases where the override is not reset, the Key Up macro is useful.

Temporary override keys follow the same precedence rules as shortcut keys.



Double Click Actions

AutoCAD 2007 introduced the Double Click Actions node. This allows you to add or modify actions for specific types of objects. Even better, if you are familiar with Visual LISP you can write code that will examine the object and perform actions based on the object's properties.

```
(defun C: DetermineDuctEdit (/ ss myObj)
  (setq ss (ssget))
  (cond ((> (sslenght ss) 0)
    (setq myObj (ssname ss 0))
    (cond ((= (strcase (cdr (assoc 8 (entget myObj)))) "REF-DUCT-CL")
      (duct: Edit myObj))
      ((command ". _PEdit" myObj))))))
  (princ))
```

Each double-click action has a single command associated with it. Double-click actions are a good reason why it is time for you to move your office beyond the legacy menu system.

Mouse Buttons

Mouse buttons too are a drag-and-drop operation. Many of today's mice sport more than two buttons (not including the wheel). However, although the CUI Editor would seem to support more than two buttons, you will find that those extra button assignments are ignored. The issue is with the drivers for the mice. Most drivers are written so that when these extra buttons are pressed, no notification is sent to the operating system *unless the driver itself has been assigned a command*. For this reason you are better off making assignments to buttons with the mouse driver rather than the CUI Editor.

LISP Files

The .mnl files load automatically with the CUI file of the same name. However, you may also load any LISP-based files along with the CUI file. This can reduce the number of autoload statements you may have had in the .mnl file.

Legacy Elements

Some features of the user interface are being depreciated. These items are still supported by the CUI Editor, but it is obvious that these elements may not be around forever. You may still need to maintain these elements so the editor provides for this.

Tablet Menus

Tablet menus will usually be migrated from a legacy menu into the CUI. However, it is still possible to create a tablet menu directly in the CUI Editor. Tablet menu commands are added in the same manner as the rest of the CUI customizations.

WinTab Drivers

Note that your tablet driver *must* be WinTab compliant. One source for a WinTab driver is www.vtablet.com

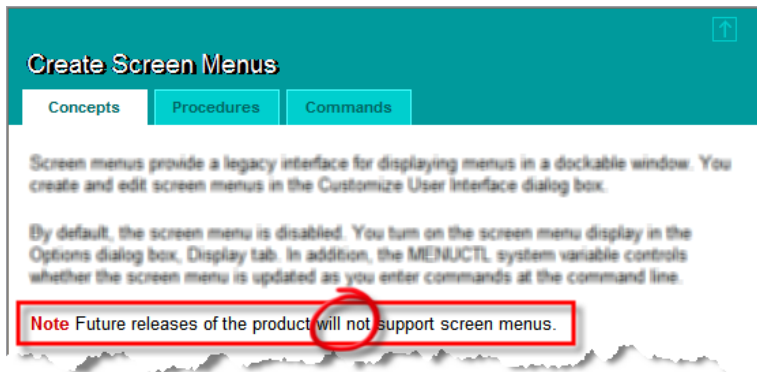
Tablet Buttons

Tablet buttons assignments are made in the same manner as mouse buttons. If you are having issues getting the buttons to work, be sure that the button has an alias assigned to it.

Screen Menus

If you are wondering what a screen menu is, this brief section does not apply to you. However, there are those that do know what they are, and they have staunch supporters. Therefore, this section is for those individuals.

Do not bother to continue using or maintaining screen menus. They are on their last legs. Note the following warning in the Customization Guide since AutoCAD 2006.



I hate to be the bearer of bad tidings. The death of screen menus has been rumored for years. Autodesk is well aware that there are firms and individuals that love to use screen menus. You may have one that you are very fond of, and have put a lot of time into making it work.

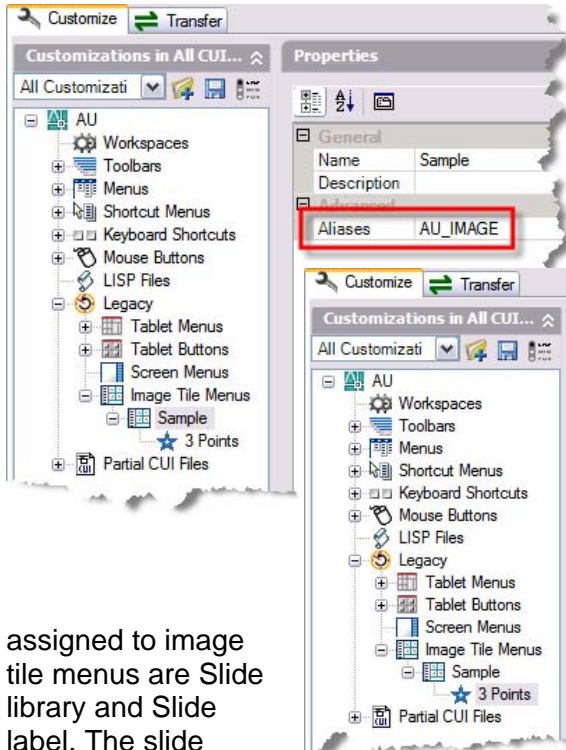
However, dynamic blocks and tool palettes go a long way in addressing the functionality that will be lost with the demise of screen menus. Now is the time to prepare. The effort will be worth it.

Image Tile Menu

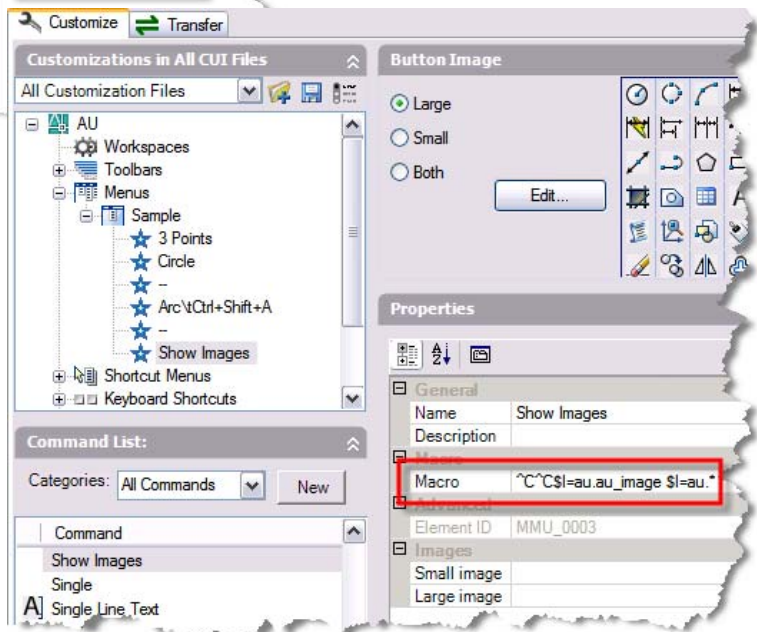
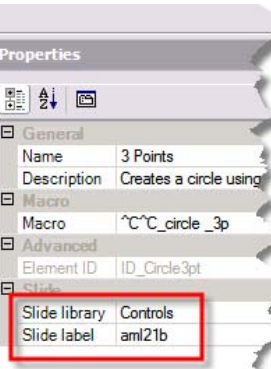
Image tile menus are still supported, but ought to be considered to be on life-support. If you have any significant effort required to maintain image menus, you might be better off using some of the newer interface feature instead, such as tool palettes.

Creating a new image tile also requires a command to display the image menu from a pull-down menu or toolbar. The macro should include the menu name, which you find in the CUI Editor at root node of a CUI file. A sample macro follows, which uses "au" as the menu name:

`^C^C$I=au. au_i mage; $I=au. *;`
 Unique properties of commands



assigned to image tile menus are Slide library and Slide label. The slide library or slide file used for image tile menus needs to be located in either AutoCAD's support path or the same location as the CUI file. You specify individual slide files in the Slide label property.



Bonus!

There may be cases where you want a list of the workspaces defined in a CUI file. Remember, a CUI file is simply an XML file. You can use Microsoft's XML interface to return a list of workspaces.

```
Private Function GetWorkspaces(CUIFileName As String) As Variant
    Dim myXML As DOMDocument
    Set myXML = New MSXML2.DOMDocument
    myXML.Load CUIFileName

    Dim myList As MSXML2.IXMLDOMNodeList
    Set myList = myXML.getElementsByTagName("WorkspaceConfig")

    Dim myCount As Long
    myCount = myList.Length - 1

    Dim resList As Variant
    ReDim resList(0 To myCount) As String

    Dim i As Long
    For i = 0 To myCount
        resList(i) = myList.Item(i).text
    Next i
    GetWorkspaces = resList
End Function
```



```
(defun i:GetWorkspaces (CUIFilename / acad xml nodes i wsList)
  (vl-load-com)
  (setq acad (vlax-Get-Acad-Object)
        xml (vla-GetInterfaceObject acad "MSXML2.DOMDocument"))
  (vlax-Invoke xml 'Load CUIFilename)
  (setq nodes (vlax-Invoke xml 'getElementByTagName "WorkspaceConfig")
        i (vlax-Get-Property nodes 'Length))
  (repeat i
    (setq i (1- i)
          wsList (cons (vlax-Get-Property (vlax-Get-Property nodes 'Item i) 'Text)
                      wsList)))
  (vlax-Release-Object nodes)
  (vlax-Release-Object xml)
  wsList)
```

Conclusion

The CUI will make the actual customization process a matter of drag-and-drop, as long as you understand how it works. “Do not over-think the CUI.” Careful planning of your CUI structure before deployment will yield rich benefits to your users.