Drawing Interchange and File Formats
Release 12

AutoCAD can be used by itself as a complete drawing editor. In some applications, however, other programs must examine drawings created by AutoCAD or generate drawings to be viewed, modified, or plotted with AutoCAD.

For example, if you've made an architectural drawing with AutoCAD, using inserted parts to represent windows, doors, and so on, you can process the drawing file and produce a bill of materials of all items used in the drawing, or even make energy-use calculations based on the area and the number and type of windows used. Another possible application is to use AutoCAD to describe structures and then send the descriptions to a more powerful computer for finite-element structural analysis. You can compute stresses and displacements and send back information to display the deformed structure as an AutoCAD drawing.

Since the AutoCAD drawing database (.dwg file) is written in a compact format that changes significantly as new features are added to AutoCAD, we do not document its format and do not recommend that you attempt to write programs to read it directly. To assist in interchanging drawings between AutoCAD and other programs, a Drawing Interchange file format (DXF) has been defined. All implementations of AutoCAD accept this format and are able to convert it to and from their internal drawing file representation.

AutoCAD also supports the Initial Graphics Exchange Specification (IGES) file format. The information comprising an AutoCAD drawing can be written out in IGES format, and IGES files can be read and converted to the AutoCAD internal format.

ASCII Drawing Interchange (DXF) Files

This section describes the AutoCAD DXF (drawing interchange file) format and the commands provided to read and write these files. DXF files are standard ASCII text files. They can easily be translated to the formats of other CAD systems or submitted to other programs for specialized analysis. AutoCAD can also produce or read a binary form of the full DXF file. This feature is described in detail later in this chapter.

DXFOUT Command - Writing a DXF File

You can generate a drawing interchange file from an existing drawing by means of the DXFOUT command:

```
Command: dxfout
```

When AutoCAD prompts you, respond with a filename or press 5 to accept the default.

The default name for the output file is the same as that of the current drawing, but with a file type of .dxf. If you specify an explicit filename, you do not need to include a file type; .dxf is assumed. If a file with the same name already exists, the existing file is deleted. If you specify the file using a file dialogue box, and a file with the same name already exists, AutoCAD tells you; allowing you to OK or cancel the deletion. Next, DXFOUT asks what precision you want for floating-point numbers and permits output of a partial DXF file containing only selected objects.

```
Enter decimal places of accuracy (0 to 16)/Entities/Binary <6>:
```

The Binary option is described later in this chapter.

If you respond with "entities" (or just "e"), DXFOUT asks you to select the objects you want written to the DXF file. Only the objects you select are included in the output file - symbol tables (including Block Definitions) will not be included. Once you've selected the desired objects, AutoCAD again prompts you for the numeric precision:

```
Enter decimal places of accuracy (0 to 16)/Binary <6>:
```

DXFIN Command - Loading a DXF File

A drawing interchange file can be converted into an AutoCAD drawing
by means of the DXFIN command:

```
Command: dxfin
```

When AutoCAD prompts you, respond with the name of the drawing interchange file to be loaded.

Full DXFIN

To load a complete DXF file, you must use DXFIN in an empty drawing, before any entities have been drawn and before any additional Block definitions, layers, linetypes, text styles, dimension styles, named views, named coordinate systems, or named viewport configurations have been created.

NOTE: If the drawing you are using as a prototype is not empty, you might find it helpful to open a new drawing using the No Prototype... button of the Create New Drawing dialogue box, as described in chapter 4 of the "AutoCAD Reference Manual." You should also be aware that some third-party applications include an acad.lsp or .mnl file that modifies your drawing upon startup.

If any errors are detected during the input, the new drawing is discarded. Otherwise, an automatic ZOOM All is performed to set the drawing extents.

Partial DXFIN

If the current drawing is not empty, DXFIN loads only the ENTITIES section of the DXF file, adding the entities found there to the current drawing. In this case, DXFIN displays the message:

```
Not a new drawing -- only ENTITIES section will be input.
```

If errors are detected during such partial DXF input, the drawing is returned to the state it was in before the DXFIN command. Otherwise, the newly added entities are drawn.

Auditing DXF Files

To ensure that corrupt data is not imported into your drawing, you can instruct AutoCAD to perform an audit after importing DXF files into your drawing with DXFIN. When you use DXFIN, the default action is to perform no automatic auditing. To activate automatic auditing, use the CONFIG command:

```
Command: config
```

Your current AutoCAD configuration appears. Press 5 to continue. From the Configuration menu select this option:

7. Configure operating parameters

From the Operating parameter menu select this option:

9. Automatic Audit after IGESIN, DXFIN, or DXBIN

Answer Y to this question:

```
Do you want an automatic audit after IGESIN, DXFIN, or DXBIN?
<N>: y
```

Return to the graphics screen by pressing 5 three times.

NOTE: This kind of audit only displays the errors AutoCAD finds; it does not correct them. To correct problems, use the AUDIT command on the drawing while you are in AutoCAD, or manually edit the DXF file.

DXF File Format

This section describes the format of a DXF file in detail. It contains technical information that you need only if you write your own programs to process DXF files or work with entity information obtained by certain AutoLISP and ADS functions.

It would probably be helpful to produce a DXF file from a small drawing, print it out, and refer to it occasionally while reading the information presented next.

General File Structure

A Drawing Interchange File is simply an ASCII text file with a file type of .dxf and specially formatted text. The overall organization of a DXF file is as follows:

1. **HEADER section** -General information about the drawing is found in this section of the DXF file. Each parameter has a variable name and an associated value (see table 11-3 for a list of the header variables).

2. **TABLES section -** This section contains definitions of named items.

   o Linetype table (LTYPE)
   o Layer table (LAYER)
   o Text Style table (STYLE)
   o View table (VIEW)
   o User Coordinate System table (UCS)
   o Viewport configuration table (VPORT)
   o Dimension Style table (DIMSTYLE)
   o Application Identification table (APPID)

3. **BLOCKS section -** This section contains Block Definition entities describing the entities that make up each Block in the drawing.

4. **ENTITIES section -** This section contains the drawing entities, including any Block References.

5. **END OF FILE**

If you use DXFOUT's Entities option, the resulting DXF file contains only the ENTITIES section and the END OF FILE marker, and the ENTITIES section reflects only the objects you select for output.

NOTE: If you select an INSERT entity, the corresponding Block definition is not included in the output file.

A DXF file is composed of many groups, each of which occupies two lines in the DXF file. The first line of a group is a group code, which is a positive nonzero integer output in FORTRAN I3 - that is, right-justified and blank filled in a three-character field (the exception to this is the four-digit extended entity data group codes, which are output in FORTRAN I4). The second line of the group is the group value, in a format that depends on the type of group specified by the group code. Although DXFOUT output has a fixed format, the DXFIN format is free.

The specific assignment of group codes depends on the item being described in the file. However, the type of the value this group supplies is derived from the group code in the following way:

Table 11-1. Group code ranges

Group code range	Following value
0 - 9	String
10 - 59	Floating-point
60 - 79	Integer
140 - 147	Floating-point
170 - 175	Integer
210 - 239	Floating-point
-999-	comment(string)
1010 - 1059	Floating-point
1060 - 1079	Integer
1000 - 1009	String

Thus a program can easily read the value following a group code without knowing the particular use of this group in an item in the file. The appearance of values in the DXF file is not affected by the setting of the UNITS command: coordinates are always represented as decimal (or possibly scientific notation if very large) numbers, and angles are always represented in decimal degrees with zero degrees to the east of origin.

Variables, table entries, and entities are described by a group that introduces the item, giving its type and/or name, followed by multiple groups that supply the values associated with the item. In addition, special groups are used for file separators such as markers for the beginning and end of sections, tables, and the file itself.

Entities, table entries, and file separators are always introduced with a 0 group code that is followed by a name describing the item.

NOTE: The maximum DXF file string length is 256 characters. If your AutoCAD drawing contains strings that exceed this number, those strings are truncated during DXFOUT. If your DXF file contains strings that exceed this number, DXFIN will fail.

Group Codes

Group codes are used both to indicate the type of the value of the group, as explained earlier, and to indicate the general use of the group. The specific function of the group code depends on the actual variable, table item, or entity description. This section indicates the general use of groups, noting as "(fixed)" any that always have the same function.

Table 11-2. AutoCAD entity group codes (by number)
+==

Group code	Value type
-0-	Identifies the start of an entity, table entry, or file separator The type of entity is given
-1-	The primary text value for an entity
-2-	A name: Attribute tag, Block name, and so on.
-2-	Also used to identify a DXF section or table name
- 3-4 -	Other textual or name values
- 5 -	Entity handle expressed as a hexadecimal string (fixed)
- 6 -	Line type name (fixed)
- 7 -	Text style name (fixed)
- 8 -	Layer name (fixed)
- 9 -	Variable name identifier (used only in HEADER section of the DXF file)
-10 -	Primary X coordinate (start point of a Line or Text entity, center of a Circle, etc.)
- 11-18 -	Other X coordinates
-20 -	Primary Y coordinate. 2n values always correspond to 1n values and immediately follow them in the file
- 21-28 -	Other Y coordinates
-30-	Primary Z coordinate. 3n values always correspond to 1n and 2n values and immediately follow them in the file
- 31-37 -	Other Z coordinates
-38 -	This entity's elevation if nonzero (fixed) Exists only in output from versions prior to R11
- 39 -	This entity's thickness if nonzero (fixed)
- 40-48 -	Floating-point values (text height, scale factors, etc.)
- 49 -	Repeated value - multiple 49 groups may appear in one entity for variable length tables (such as the dash lengths in the LTYPE table). A 7x group always appears before the first 49 group to specify the table length
- 50-58 -	Angles
- 62 -	Color number (fixed)
- 66 -	"Entities follow" flag (fixed)
- 67 -	Identifies whether entity is in model space or | paper space
- 68 -	Identifies whether viewport is on but fully off screen, is not active, or is off
- 69 -	Viewport identification number
- 70-78 -	Integer values such as repeat counts, flag bits, or modes
-210, 220, 230	X, Y, and Z components of extrusion direction (fixed)
- 999 -	Comments

Group code	Value type
-1000 -	An ASCII string (up to 255 bytes long) extended entity data
- 1001 -	Registered application name (ASCII string up to 31 bytes long) for XDATA (fixed)
- 1002 -	Extended entity data control string ("{" or "}") (fixed)
- 1003 -	Extended entity data Layer name
-1004 -	Chunk of bytes (up to 127 bytes long) in extended entity data
- 1005 -	Extended entity data database handle
- 1010, 1020 , 1030 -	Extended entity data X, Y, and Z coordinates
-1011, 1021, 1023 -	Extended entity data X, Y, and Z coordinates of 3D world space position
- 1012, 1022, 1032 -	Extended entity data X, Y, and Z components of 3D world space displacement
- 1013, 1023, 1033 -	Extended entity data X, Y, and Z components of 3D world space direction
-1040 -	Extended entity data Floating-point value
- 1041 -	Extended entity data distance value
- 1042 -	Extended entity data scale factor
- 1070 -	Extended entity data 16-bit signed integer

Comments

The 999 group code indicates that the following line is a comment string. DXFOUT does not currently include such groups in a DXF output file, but DXFIN honors them and ignores the comments. Thus, you can use the 999 group to include comments in a DXF file you've edited. For example:

```
999
This is a comment.
999
This is another comment.
```

File Sections

The DXF file is subdivided into four editable sections, plus the END OF FILE marker. File separator groups are used to delimit these file
sections. The following is an example of a void DXF file with only
the section markers and table headers present:

```
 0          (Begin HEADER section)
SECTION
 2
HEADER
            <<<<Header variable items go here>>>>
0
ENDSEC      (End HEADER section)
 0          (Begin TABLES section)
SECTION
 2
```

```
TABLES
 0
TABLE
 2
VPORT
 70
(viewport table maximum item count)
            <<<<viewport table items go here>>>>
0
ENDTAB
0
TABLE
2
APPID, DIMSTYLE, LTYPE, LAYER, STYLE, UCS, VIEW, or VPORT
70
(Table maximum item count)
            <<<<Table items go here>>>>
0
ENDTAB
0
ENDSEC       (End TABLES section)
0            (Begin BLOCKS section)
SECTION
2
BLOCKS
            <<<<Block definition entities go here>>>>
0
ENDSEC       (End BLOCKS section)
0            (Begin ENTITIES section)
SECTION
2
ENTITIES
            <<<<Drawing entities go here>>>>
0
ENDSEC       (End ENTITIES section)
0
EOF          (End of file)
```

HEADER Section

The HEADER section of the DXF file contains settings of variables associated with the drawing. These variables are set with various commands and are the type of information displayed by the STATUS command. Each variable is specified in the header section by a 9 group giving the variable's name, followed by groups that supply the variable's value. The following list shows the header variables and their meanings.

Although this list is very similar to the list of system variables in Appendix A of this manual, the two lists are not identical. Be sure you're referring to the proper list.

NOTE: $AXISMODE and $AXISUNIT are no longer functional in Release 12.

Table 11-3. DXF system variables

Variable	Type	Description
$ACADVER	1	The AutoCAD drawing database version number; AC1006 = R10, AC1009 = R11 and R12
$ANGBASE	50	Angle 0 direction
$ANGDIR	70	1 = clockwise angles, 0 = counterclockwise
$ATTDIA	70	Attribute entry dialogs, 1 = on, 0 = off
$ATTMODE	70	Attribute visibility: 0 = none, 1 = normal, 2 = all
$ATTREQ	70	Attribute prompting during INSERT, 1 = on, 0 = off
$AUNITS	70	Units format for angles
$AUPREC	70	Units precision for angles
$AXISMODE	70	Axis on if nonzero (not functional in Release 12)
$AXISUNIT	10, 20	Axis X and Y tick spacing (not functional in Release 12)
$BLIPMODE	70	Blip mode on if nonzero
$CECOLOR	62	Entity color number; 0 = BYBLOCK, 256 = BYLAYER
$CELTYPE	6	Entity linetype name, or BYBLOCK or BYLAYER
$CHAMFERA	40	First chamfer distance
$CHAMFERB	40	Second chamfer distance
$CLAYER	8	Current layer name
$COORDS	70	0 = static coordinate display, 1 = continuous update, 2 = "d<a" format
$DIMALT	70	Alternate unit dimensioning performed if nonzero
$DIMALTD	70	Alternate unit decimal places
$DIMALTF	40	Alternate unit scale factor

Variable	Type	Description
$DIMAPOST	1	Alternate dimensioning suffix
$DIMASO	70	1 = create associative dimensioning, 0 = draw individual entities
$DIMASZ	40	Dimensioning arrow size
$DIMBLK	2	Arrow block name
$DIMBLK1	1	First arrow block name
$DIMBLK2	1	Second arrow block name
$DIMCEN	40	Size of center mark/lines
$DIMCLRD	70	Dimension line color, range is 0 = BYBLOCK, 256 = BYLAYER
$DIMCLRE	70	Dimension extension line color, range is 0 = BYBLOCK, 256 = BYLAYER
$DIMCLRT	70	Dimension text color, range is 0 = BYBLOCK, 256 = BYLAYER
$DIMDLE	40	Dimension line extension
$DIMDLI	40	Dimension line increment
$DIMEXE	40	Extension line extension
$DIMEXO	40	Extension line offset
$DIMGAP	40	Dimension line gap
$DIMLFAC	40	Linear measurements scale factor
$DIMLIM	70	Dimension limits generated if nonzero
$DIMPOST	1	General dimensioning suffix
$DIMRND	40	Rounding value for dimension distances
$DIMSAH	70	Use separate arrow blocks if nonzero
$DIMSCALE	40	Overall dimensioning scale factor
$DIMSE1	70	First extension line suppressed if nonzero
$DIMSE2	70	Second extension line suppressed if nonzero
$DIMSHO	70	1 = Recompute dimensions while dragging, 0 = drag original image
$DIMSOXD	70	Suppress outside extensions dimension lines if nonzero
$DIMSTYLE	2	Dimension style name
$DIMTAD	70	Text above dimension line if nonzero
$DIMTFAC	40	Dimension tolerance display scale factor
$DIMTIH	70	Text inside horizontal if nonzero
$DIMTIX	70	Force text inside extensions if nonzero
$DIMTM	40	Minus tolerance
$DIMTOFL	70	If text outside extensions, force line extensions between extensions if nonzero
$DIMTOH	70	Text outside horizontal if nonzero

Variable	Type	Description
$DIMTOL	70	Dimension tolerances generated if nonzero
$DIMTP	40	Plus tolerance
$DIMTSZ	40	Dimensioning tick size: 0 = no ticks
$DIMTVP	40	Text vertical position
$DIMTXT	40	Dimensioning text height
$DIMZIN	70	Zero suppression for "feet & inch" dimensions
$DWGCODEPAGE	70	Drawing code page. Set to the system code page when a new drawing is created, but not otherwise maintained by AutoCAD
$DRAGMODE	70	0 = off, 1 = on, 2 = auto
$ELEVATION	40	Current elevation set by ELEV command
$EXTMAX	10, 20, 30	X, Y, and Z drawing extents upper right corner (in WCS)
$EXTMIN	10, 20, 30	X, Y, and Z drawing extents lower left corner (in WCS)
$FILLETRAD	40	Fillet radius
$FILLMODE	70	Fill mode on if nonzero
$HANDLING	70	Handles enabled if nonzero
$HANDSEED	5	Next available handle
$INSBASE	10, 20, 30	Insertion base set by BASE command (in WCS)
$LIMCHECK	70	Nonzero if limits checking is on
$LIMMAX	10, 20	XY drawing limits upper right corner (in WCS)
$LIMMIN	10, 20	XY drawing limits lower left corner (in WCS)
$LTSCALE	40	Global linetype scale
$LUNITS	70	Units format for coordinates and distances
$LUPREC	70	Units precision for coordinates and distances
$MAXACTVP	70	Sets maximum number of viewports to be regenerated
$MENU	1	Name of menu file
$MIRRTEXT	70	Mirror text if nonzero
$ORTHOMODE	70	Ortho mode on if nonzero
$OSMODE	70	Running object snap modes
$PDMODE	70	Point display mode
$PDSIZE	40	Point display size
$PELEVATION	40	Current paper space elevation
$PEXTMAX	10, 20, 30	Maximum X, Y, and Z extents for paper space
$PEXTMIN	10, 20,	Minimum X, Y, and Z extents for

Variable	Type	Description
	30	paper space
$PLIMCHECK	70	Limits checking in paper space when nonzero
$PLIMMAX	10, 20	Maximum X and Y limits in paper space
$PLIMMIN	10, 20	Minimum X and Y limits in paper space
$PLINEGEN	70	Governs the generation of linetype patterns around the vertices of a 2D Polyline 1 = linetype is generated in a continuous pattern around vertices of the Polyline 0 = each segment of the Polyline starts and ends with a dash
$PLINEWID	40	Default Polyline width
$PSLTSCALE	70	Controls paper space linetype scaling 1 = no special linetype scaling 0 = viewport scaling governs linetype scaling
$PUCSNAME	2	Current paper space UCS name
$PUCSORG	10, 20, 30	Current paper space UCS origin
$PUCSXDIR	10, 20, 30	Current paper space UCS X axis
$PUCSYDIR	10, 20, 30	Current paper space UCS Y axis
$QTEXTMODE	70	Quick text mode on if nonzero
$REGENMODE	70	REGENAUTO mode on if nonzero
$SHADEDGE	70	0 = faces shaded, edges not highlighted 1 = faces shaded, edges highlighted in black 2 = faces not filled, edges in entity color 3 = faces in entity color, edges in black
$SHADEDIF	70	Percent ambient/diffuse light, range 1 100, default 70
$SKETCHINC	40	Sketch record increment
$SKPOLY	70	0 = sketch lines, 1 = sketch polylines
$SPLFRAME	70	Spline control polygon display, 1 = on, 0 = off
$SPLINESEGS	70	Number of line segments per spline patch

Variable	Type	Description
$SPLINETYPE	70	Spline curve type for PEDIT Spline (See your AutoCAD Reference Manual)
$SURFTAB1	70	Number of mesh tabulations in first direction
$SURFTAB2	70	Number of mesh tabulations in second direction
$SURFTYPE	70	Surface type for PEDIT Smooth (See your AutoCAD Reference Manual)
$SURFU	70	Surface density (for PEDIT Smooth) in M direction
$SURFV	70	Surface density (for PEDIT Smooth) in N direction
$TDCREATE	40	Date/time of drawing creation
$TDINDWG	40	Cumulative editing time for this drawing
$TDUPDATE	40	Date/time of last drawing update
$TDUSRTIMER	40	User elapsed timer
$TEXTSIZE	40	Default text height
$TEXTSTYLE	7	Current text style name
$THICKNESS	40	Current thickness set by ELEV command
$TILEMODE	70	1 for previous release compatibility mode, 0 otherwise
$TRACEWID	40	Default Trace width
$UCSNAME	2	Name of current UCS
$UCSORG	10, 20, 30	Origin of current UCS (in WCS)
$UCSXDIR	10, 20, 30	Direction of current UCS's X axis (in World coordinates)
$UCSYDIR	10, 20, 30	Direction of current UCS's Y axis (in World coordinates)
$UNITMODE	70	Low bit set = display fractions, feet and inches, and surveyor's angles in input format
$USERI1 5	70	Five integer variables intended for use by third party developers
$USERR1 5	40	Five real variables intended for use by third party developers
$USRTIMER	70	0 = timer off, 1 = timer on
$VISRETAIN	70	0 = don't retain Xref dependent visibility settings, 1 = retain Xref dependent visibility settings
$WORLDVIEW	70	1 = set UCS to WCS during DVIEW/VPOINT, 0 = don't change UCS

The following header variables existed prior to AutoCAD Release 11 but now have independent settings for each active viewport. DXFIN honors these variables when read from DXF files, but if a VPORT symbol table with *ACTIVE entries is present (as is true for any DXF file produced by Release 11 or higher), the values in the VPORT table entries override the values of these header variables.

Table 11-4. Revised VPORT header variables

Variable	Type	Description
$FASTZOOM	70	Fast zoom enabled if nonzero
$GRIDMODE	70	Grid mode on if nonzero
$GRIDUNIT	10, 20	Grid X and Y spacing
$SNAPANG	50	Snap grid rotation angle
$SNAPBASE	10, 20	Snap grid base point (in UCS)
$SNAPISOPAIR	70	Isometric plane: 0 = left, 1 = top, 2 = right
$SNAPMODE	70	Snap mode on if nonzero
$SNAPSTYLE	70	Snap style: 0 = standard, 1 = isometric
$SNAPUNIT	10, 20	Snap grid X and Y spacing
$VIEWCTR	10, 20	XY center of current view on screen
$VIEWDIR	10, 20, 30	Viewing direction (direction from target, in WCS)
$VIEWSIZE	40	Height of view

The date/time variables ($TDCREATE and $TDUPDATE) are output as real numbers in the following format:

<Julian date>.<Fraction>

The elapsed time variables ($TDINDWG and $TDUSRTIMER) have a similar format:

<Number of days>.<Fraction>

The date and time variables are described on page 299.

TABLES Section

The TABLES section contains several tables, each of which contains a variable number of table entries.

The order of the tables may change, but the LTYPE table will always precede the LAYER table. Each table is introduced with a 0 group with the label TABLE. This is followed by a 2 group identifying the particular table (VPORT, LTYPE, LAYER, STYLE, VIEW, DIMSTYLE, UCS or APPID) and a 70 group that specifies the maximum number of table entries that may follow. Table names are always output in uppercase characters.

The tables in a drawing can contain deleted items, but these are not written to the DXF file. Thus, fewer table entries may follow the table header than are indicated by the 70 group, so don't use the count in the 70 group as an index to read in the table. This group is provided so that a program which reads DXF files can allocate an array large enough to hold all the table entries that follow.

Following this header for each table are the table entries. Each table item consists of a 0 group identifying the item type (same as table name, e.g., LTYPE or LAYER), a 2 group giving the name of the table entry, a 70 group specifying flags relevant to the table entry (defined for each following table), and additional groups that give the value of the table entry. The end of each table is indicated by a 0 group with the value ENDTAB.

The 70 group flag bit values that apply to all table entries are described in the following chart. Additional 70 group values that apply to LAYER, STYLE, and VIEW table entries are described in the appropriate sections below.

Table 11-5. Group 70 bit codes that apply to all table entries

Flag bit value	Meaning
16	If set, table entry is externally dependent on an Xref
32	If this bit and bit 16 are both set, the externally dependent Xref has been successfully resolved
64	If set, the table entry was referenced by at least one entity in the drawing the last time the drawing was edited. (This flag is for the benefit of AutoCAD commands; it can be ignored by most programs that read DXF files, and need not be set by programs that write DXF files)

The following are the groups used for each type of table item. All groups are present for each table item.

APPID 2 (user-supplied application name), 70 (standard flag values).These table entries maintain a set of names for all applications registered with a drawing.

DIMSTYLE 2 (dimension style name), 70 (standard flag values), and the following,described by dimension variable name:
3 (dimpost), 4 (dimapost), 5 (dimblk), 6 (dimblk1), 7 (dimblk2), 40 (dimscale), 41 (dimasz), 42 (dimexo), 43 (dimdli), 44 (dimexe), 45 (dimrnd), 46 (dimdle), 47 (dimtp), 48 (dimtm), 140 (dimtxt), 141 (dimcen), 142 (dimtsz), 143 (dimaltf), 144 (dimlfac), 145 (dimtvp), 146 (dimtfac), 147 (dimgap), 71 (dimtol), 72 (dimlim), 73 (dimtih), 74 (dimtoh), 75 (dimse1), 76 (dimse2), 77 (dimtad), 78 (dimzin), 170 (dimalt), 171 (dimaltd), 172 (dimtofl), 173 (dimsah), 174 (dimtix), 175 (dimsoxd), 176 (dimclrd), 177 (dimclre), 178 (dimclrt).

LTYPE 2 (linetype name), 70 (standard flag values), 3 (descriptive text for linetype), 72 (alignment code; value is always 65, the ASCII code for `A'), 73 (number of dash length items), 40 (total pattern length), and optionally: 49 (dash length 1), 49 (dash length 2), and so on.

LAYER 2 (layer name), 70 (standard flag values), 62 (color number, negative if layer is off), 6 (linetype name). In addition to the standard flags, the 70 group flag is bit coded as follows:

Table 11-6. Group 70 bit codes for LAYER table

Flag bit value	Meaning
1	If set, layer is frozen
2	If set, layer is frozen by default in new Viewports
4	If set, layer is locked

If no value (0) is set, the layer is on and thawed. The fourth bit (8) and the eighth bit (128) are not used Xref-dependent layers are output during DXFOUT. For these layers, the associated linetype name in the DXF file is always CONTINUOUS.

STYLE 2 (style name), 70 (standard flag values), 40 (fixed text height; 0 if not fixed), 41 (width factor), 50 (oblique angle), 71 (text generation flags), 42 (last height used), 3 (primary font filename), 4 (big-font file name; blank if none).
If the third bit (4) is set in the 70 group flags, this is a vertically oriented text style.

A STYLE table item is used to record shape file LOAD requests also. In this case the first bit (1) is set in the 70 group flags and only the 3 group (shape filename) is meaningful (all the other groups are output, however).

The text generation flags are a bit-coded field with the following bit meanings:

Table 11-7. Group 71 bit codes for STYLE table

Flag bit value	Meaning
2	Text is backward (mirrored in X)
4	Text is upside down (mirrored in Y)

UCS 2 (UCS name), 70 (standard flag values), 10, 20, 30(origin), 11, 21, 31 (X axis direction), 12, 22, 32 (Y axis direction). All in World coordinates.

VIEW 2 (name of view), 70 (standard flag values), 40 and 41 (view height and width, in DCS), 10 and 20 (view center point, in DCS), 11, 21, 31 (view direction from target, in WCS), 12, 22, 32 (target point, in WCS), 42 (lens length), 43 and 44 (front and back clipping planes - offsets from target point), 50 (twist angle), 71 view mode (see VIEWMODE system variable in appendix A).

If the first bit (1) is set in the 70 group flags, this is a paper space view.
(See chapter 2 of the "AutoLISP Programmer's Reference" for information on DCS, the Display Coordinate System.)

VPORT 2 (viewport name), 70 (standard flag values), 10 and 20 (lower-left corner of viewport; 0.0 to 1.0), 11 and 21 (upper-right corner), 12 and 22 (view center point, in WCS), 13 and 23 (snap base point), 14 and 24 (snap spacing, X and Y), 15 and 25 (grid spacing, X and Y), 16, 26, 36 (view direction from target point), 17, 27, 37 (view target point), 40 (view height), 41 (viewport aspect ratio), 42 (lens length), 43 and 44 (front and back clipping planes; offsets from target point),50 (snap rotation angle), 51 (view twist angle), 68 (status field), 69 (ID), 71 (view mode; see VIEWMODE system variable in appendix A), 72 (circle zoom percent), 73 (fast zoom setting), 74 (UCSICON setting), 75 (snap on/off), 76 (grid on/off), 77 (snap style), 78 (snap isopair).

The VPORT table is unique in that it may contain several entries with the same name (indicating a multiple-viewport configuration). The entries corresponding to the active viewport configuration all have the name *ACTIVE. The first such entry describes the current viewport.

BLOCKS Section

The Blocks section of the DXF file contains all the Block Definitions. This section contains the entities that make up the Blocks used in the drawing, including anonymous Blocks generated by the HATCH command and by associative dimensioning. The format of the entities in this section is identical to those in the Entities section described later, so see that section for details. All entities in the Blocks section appear between Block and Endblk entities. Block and Endblk entities appear only in the Blocks section. Block definitions are never nested (that is, no Block or Endblk entity ever appears within another Block-Endblk pair), although a Block definition can contain an INSERT entity.
External References are written in the DXF file as any Block Definition, except they also include a text string (group code 1) of the path and filename of the External Reference. This is the text

string format: ***Xref filename***

ENTITIES Section

Entity items appear in both the BLOCK and ENTITIES sections of the DXF file. The appearance of entities in the two sections is identical.

The following gives the format of each entity as it appears in the file. Some groups that define an entity always appear, and some are optional and appear only if they differ from their default values. In the following discussion, groups that always occur are given by their group number and function, while optional groups are indicated by -optional N following the group description. N is the default value if the group is omitted.

Programs that read DXF files should not assume that the groups describing an entity occur in the order given here. The end of the groups that make up an entity is indicated by the next 0 group, beginning the next entity or indicating the end of the section.

Remember that a DXF file is a complete representation of the drawing database, and that as AutoCAD is further enhanced, new groups will be added to entities to accommodate additional features. Accommodating DXF files from future releases of AutoCAD will be easier if you write your DXF processing program in a table-driven way, ignoring any groups not presently defined, and making no assumptions about the order of groups in an entity.

Each entity begins with a 0 group identifying the entity type. The names used for the entities are given on the following pages. Every entity contains an 8 group that gives the name of the layer on which the entity resides. Each entity may have elevation, thickness, linetype, or color information associated with it.

If handles are enabled, every entity has a 5 group containing its handle (as a string representing a hexadecimal number).

The following groups are included only if the entity has nondefault values for these properties. When a group is omitted, its default value upon input (when using DXFIN) is indicated in the third column. If the value of a group is equal to the default, it is omitted upon output (when using DXFOUT).

__Table 11-8. Group codes common to all entities__

Group code	Meaning	If omitted, defaults to...
6	Linetype name (if not BYLAYER). The special name BYBLOCK indicates a floating linetype	BYLAYER
38	Elevation (if nonzero). Exists only in output from versions prior to R11. Otherwise, Z coordinates are supplied as 3x-groups as part of each of the entity's defining points	0
39	Thickness (if nonzero)	0
62	Color number (if not BYLAYER). Zero indicates the BYBLOCK (floating) color. 256 indicates the BYLAYER color	BYLAYER
67	Absent or zero indicates entity is in model space. One indicates entity is in paper space, other values are reserved	0
210,	These groups are included for each Line, 0,0,1	
220,	Point, Circle, Shape, Text, Arc, Trace,	
230	Solid, Block Reference, Polyline, Dimension, Attribute, and Attribute Definition entity if its extrusion direction is not parallel to the World Z axis. They indicate the X, Y, and Z components of the entity's extrusion direction	

The rest of the groups that make up an entity item are described next. Many of the entities include "flag" groups. These are integer codes (6x or 7x groups) that encode various pieces of information regarding the entity, and are specific to the particular entity type. In the following descriptions, the term bit-coded means that the flag contains various true/false values coded as the sum of the bit values given. Any bits not defined in the following section should be ignored in these fields and set to zero when constructing a DXF file.

LINE 10, 20, 30 (start point), 11, 21, 31 (endpoint).

POINT 10, 20, 30 (point).

Point entities have an optional 50 group that determines the orientation of PDMODE images. The group value is the negative of the Entity Coordinate Systems (ECS) angle of the UCS X axis in effect when the point was drawn. The X axis of the UCS in effect when the point was drawn is always parallel to the XY plane for the point's ECS, and the angle between the UCS X axis and the ECS X axis is a single 2D angle. The value in group 50 is the angle from horizontal (the effective X axis) to the ECS X axis. Entity Coordinate Systems (ECS) are described later in this section.

CIRCLE 10, 20, 30 (center), 40 (radius).

ARC 10, 20, 30 (center), 40 (radius), 50 (start angle), 51 (end angle).

TRACE Four points defining the corners of the trace: (10, 20, 30), (11, 21, 31), (12, 22, 32), and (13, 23, 33).

SOLID Four points defining the corners of the solid: (10, 20, 30), (11, 21, 31), (12, 22, 32), and (13, 23, 33). If only three points were entered (forming a triangular solid), the third and fourth points will be the same.

TEXT 10, 20, 30 (insertion point), 40 (height), 1 (text value), 50 (rotation angle -optional 0), 41 (relative X-scale factor -optional 1), 51 (oblique angle -optional 0), 7 (text style name -optional STANDARD), 71 (text generation flags -optional 0), 72 (horizontal justification type -optional 0), 73 (vertical justification type -optional 0) 11, 21, 31 (alignment point -optional, appears only if 72 or 73 group is present and nonzero).

Table 11-9. Group 71 bit codes for Text entity

Flag bit value	Meaning
2	Text is backward (mirrored in X)
4	Text is upside down (mirrored in Y)

The justification-type value (group codes 72 and 73, not bit-coded) indicates the text-justification style used on the text, as shown in the following table:

Table 11-10. Group 72 & 73 bit codes for Text entity

Group 73 (vertical alignment)	Group 72 (horizontal alignment)			
	0	1	2	3
3 (Top)	TLeft	TCenter	TRight	
2 (Middle)	MLeft	MCenter	MRight	
1 (Bottom)	BLeft	BCenter	BRight	
0 (Baseline)	Left	Center	Right	Aligned

If the justification is anything other than baseline/left (groups 72 and 73 both 0), group codes 11, 21, and 31 specify the alignment point (or the second alignment point for Align or Fit).

DXFOUT handles ASCII control characters in text strings by expanding the character into a ^ (caret) followed by the appropriate letter. For example, an ASCII Control-G (BEL, decimal code 7) is output as ^G. If the text itself contains a caret character, it is expanded to ^ (caret, space).

DXFIN performs the complementary conversion.

SHAPE 10, 20, 30 (insertion point), 40 (size), 2 (shape name), 50 (rotation angle -optional 0), 41 (relative X-scale factor -optional 1), 51 (oblique angle -optional 0).

BLOCK 2 (Block name), 3 (this is also the Block name), 70 (Block type flag), 10, 20, 30 (Block base point), and if the Block is an Xref Block it will also contain group code 1 (Xref pathname). Block entities appear only in the BLOCKS section, not in the ENTITIES section. The "Block type flag" (group 70) is bit-coded, with the following bit meanings:

Table 11-11. Group 70 bit codes for Block table

Flag bit value	Meaning
1	This is an anonymous Block generated by hatching, associative dimensioning, other internal operations, or an application
2	This Block has Attributes
4	This Block is an external reference (Xref)
8	not used
16	This Block is externally dependent
32	This is a resolved external reference, or
	dependent of an external reference
64	This definition is referenced

ENDBLK No groups. Appears only in BLOCKS section.

INSERT 66 (Attributes follow flag -optional 0), 2 (Block name), 10, 20, 30 (insertion point), 41 (X- scale factor -optional 1), 42 (Y scale factor -optional 1), 43 (Z- scale factor -optional 1), 50 (rotation angle -optional 0), 70 and 71 (column and row counts -optional 1), 44 and 45 (column and row spacing -optional 0).

If the value of the "Attributes follow" flag is 1, a series of Attribute (Attrib) entities is expected to follow the Insert, terminated by a sequence end (Seqend) entity.

ATTDEF 10, 20, 30 (text start), 40 (text height), 1 (default value, see "Text" on page 260 for handling of ASCII control characters), 3 (prompt string), 2 (tag string), 70 (Attribute flags), 73 (field length -optional 0), 50 (text rotation - optional 0), 41 (relative X scale factor -optional 1), 51 (oblique angle -optional 0), 7 (text style name -optional STANDARD), 71 (text generation flags -optional 0, see "Text" on page 260), 72 (horizontal text justification type - optional 0, see "Text" on page 260), 74 (vertical text justification type -optional 0 see group 73 in "Text" on page 260), 11, 21, 31 (alignment point -optional, appears only if 72 or 74 group is present and nonzero).

The "Attribute flags" (group code 70) are a bit-coded field in which the bits have the following meanings:

Table 11-12. Group 70 bit codes for Attdef entity

Flag bit value	Meaning
1.00 F	Attribute is invisible (does not display)
2.00 F	This is a constant Attribute
4.00 F	Verification is required on input of this Attribute
8.00 F	Attribute is preset (no prompt during insertion)

ATTRIB 10, 20, 30 (text start), 40 (text height), 1 (value, see "Text" on page 260 for handling ASCII control characters), 2 (Attribute tag), 70 (Attribute flags; see Attdef), 73 (field length -optional 0), 50 (text rotation -optional 0), 41 (relative X scale factor -optional 1), 51 (oblique angle -optional 0), 7 (text style name -optional STANDARD), 71 (text generation flags -optional 0, see "Text" on page 260), 72 (horizontal text justification ype -optional 0, see "Text" on page 260), 74 (vertical text justification type -optional 0, see group 73 in "Text" on page 260), 11, 21, 31 (alignment point optional, appears only if 72 or 74 group is present and nonzero).

POLYLINE 66 (vertices-follow flag), 10, 20, 30 (polyline elevation - 30 supplies elevation, 10 and 20 are always set to zero), 70 (Polyline flag -optional 0), 40 (default starting width - optional 0), 41 (default ending width optional 0), 71 and 72 (polygon mesh M and N vertex counts -optional 0), 73 and 74 (smooth surface M and N densities -optional 0), 75 (curves and smooth surface type - optional 0). The default widths apply to any vertex that doesn't supply widths (see later).

The "vertices follow" flag is always 1, indicating that a series of Vertex entities is expected to follow the Polyline, terminated by a sequence end (Seqend) entity. The polyline flag (group code 70) is a bit-coded field with bits defined as follows:

Table 11-13. Group 70 bit codes for Polyline entity

Flag bit value	Meaning
1	This is a closed Polyline (or a polygon mesh closed in the M direction)
2	Curve-fit vertices have been added
4	Spline-fit vertices have been added
8	This is a 3D Polyline
16	This is a 3D polygon mesh. Group 75 indicates the smooth surface type as follows: 0 = no smooth surface fitted 5 = quadratic B-spline surface 6 = cubic B-spline surface 8 = Bezier surface
32	The polygon mesh is closed in the N direction
64	This Polyline is a polyface mesh
128	The linetype pattern is generated continuously around the vertices of this Polyline

A polyface mesh is represented in DXF as a variant of a Polyline entity. The Polyline header is identified as introducing a polyface mesh by the presence of the 64 bit in the Polyline flags (70) group. The 71 group specifies the number of vertices in the mesh, and the 72 group, the number of faces. While these counts are correct for all meshes created with the PFACE command, applications are not required to place correct values in these fields, and AutoCAD actually never relies upon their accuracy.

Following the Polyline header is a sequence of Vertex entities that specify the vertex coordinates and faces that compose the mesh. Vertices such as these are described in the following subsection on Vertex.

Applications might want to represent polygons with an arbitrarily large number of sides in polyface meshes. However, the AutoCAD entity structure imposes a limit on the number of vertices that a given face entity can specify. You can represent more complex polygons by decomposing them into triangular wedges. Their edges should be made invisible to prevent visible artifacts of this subdivision from being drawn. The PFACE command performs this subdivision automatically, but when applications generate polyface meshes directly, the applications must do this themselves.

The number of vertices per face is the key parameter in this subdivision process. The PFACEVMAX system variable provides an application with the number of vertices per face entity. This value is read-only, and is set to 4.

Polyface meshes created with the PFACE command are always generated with all the vertex coordinate entities first, followed by the face definition entities. The code within AutoCAD that processes polyface meshes does not, at present, require this ordering; it works even with interleaved vertex coordinates and face definitions as long as no face specifies a vertex with an index that appears after it in the database. Programs that read polyface meshes from DXF would be wise to be as tolerant of odd vertex and face ordering as AutoCAD is.

VERTEX 10, 20, 30 (location), 40 (starting width -optional, see earlier), 41 (ending width -optional, see above), 42 (bulge -optional 0), 70 (vertex flags -optional 0), 50 (curve fit tangent direction -optional). The bulge is the tangent of 1/4 the included angle for an arc segment, made negative if the arc goes clockwise from the start point to the endpoint; a bulge of 0 indicates a straight segment, and a bulge of 1 is a semicircle. The meaning of the bit-coded Vertex flag (group code 70) is shown in the following table

Table 11-14. Group 70 bit codes for Vertex entity

Flag bit value	Meaning
1	Extra vertex created by curve-fitting
2	Curve-fit tangent defined for this vertex. A curve-fit tangent direction of 0 may be omitted from the DXF output, but is significant if this bit is set
4	Unused (never set in DXF files)
8	Spline vertex created by spline-fitting
16	Spline frame control point
32	3D Polyline vertex
64	3D polygon mesh vertex
128	Polyface mesh vertex

Every Vertex that is part of a polyface mesh has the 128 bit set in its Vertex flags (70) group. If the entity specifies the coordinates of a vertex of the mesh, the 64 bit is set as well and the 10, 20, and 30 groups give the vertex coordinates. The vertex indexes are determined by the order in which the Vertex entities appear within the Polyline, with the first numbered 1.

If the Vertex defines a face of the mesh, its Vertex flags (70) group has the 128 bit set but not the 64 bit. The 10, 20, and 30 (location) groups of the face entity are irrelevant and are always written as zero in a DXF file. The vertex indexes that define the mesh are given by 71, 72, 73, and 74 groups, the values of which are integers specifying one of the previously defined vertices by index. If the index is negative, the edge that begins with that vertex is invisible. The first zero vertex marks the end of the vertices of the face. Since the 71 through 74 groups are optional fields with default values of zero, they are present in DXF only if nonzero.

SEQEND No fields. This entity marks the end of vertices (Vertex type name) for a Polyline, or the end of Attribute entities (Attrib type name) for an Insert entity that has Attributes (indicated by 66 group present and nonzero in Insert entity).

3DFACE Four points defining the corners of the face: (10, 20, 30), (11, 21, 31), (12, 22, 32), and (13, 23, 33). 70 (invisible edge flags -optional 0). If only three points are entered (forming a triangular face), the third and fourth points will be the same. The meanings of the bit-coded "invisible edge flags" are shown in the following table:

Table 11-15. Group 70 bit codes for 3D Face entity

Flag bit value	Meaning
1	First edge is invisible
2	Second edge is invisible
4	Third edge is invisible
8	Fourth edge is invisible

VIEWPORT 10,20,30 (center point of entity in paper space coordinates), 40 (width in paper space units), 41 (height in paper space units), 68 (viewport status field), 69 (viewport ID, permanent during editing sessions, but mutable between sessions; the paper space viewport entity always has an ID of 1).

The value of the viewport status field (68) is interpreted as follows:

-1	On,	but is fully off-screen or is one of the viewports not active because the $MAXACTVP count is currently being exceeded.
0	Off.	
<positive value>	On,	active and the value indicates the order of "stacking" for the viewports, with 1 applying to the active viewport, which is also the highest, 2 applying to the next viewport in the stack, and so on.

In addition, the extended entity data groups in the following table apply to viewports.

NOTE: In contrast to normal entity data, the same extended entity group code can appear multiple times, and order is important.

Table 11-16. Extended entity group codes for Viewports

Group	**Description**
1001	Application name. This field will always be the string "ACAD"
1000	Begin viewport data. This field will always be the string "MVIEW". Other data groups may appear in the future
1002	Begin window descriptor data. This field will always be the string "{"
1070	Extended entity data version number. For Releases 11 and 12, this field will always be the integer 16
1010	View target point X value
1020	View target point Y value
1030	View target point Z value
1010	View direction vector X value
1020	View direction vector Y value

Group	Description
1030	View direction vector Z value
1040	View twist angle
1040	View height
1040	View center point X value
1040	View center point Y value
1040	Perspective lens length
1040	Front clip plane Z value
1040	Back clip plane Z value
1070	View mode
1070	Circle zoom
1070	Fast zoom setting
1070	UCSICON setting
1070	Snap ON/OFF
1070	Grid ON/OFF
1070	Snap style
1070	Snap ISOPAIR
1040	Snap angle
1040	Snap base point UCS X coordinate
1040	Snap base point UCS Y coordinate
1040	Snap X spacing
1040	Snap Y spacing
1040	Grid X spacing
1040	Grid Y spacing
1070	Hidden in plot flag
1002	Begin frozen layer list (possibly empty). This field will always be the string "{"
1003...	The names of layers frozen in this viewport. This list may include Xref-dependent layers. Any number of 1003 groups may appear here
1002	End frozen layer list. This field will always be the string "}"
1002	End viewport data. This field will always be the string "}"

DIMENSION 2 (name of pseudo-Block containing the current dimension entity geometry), 3 (dimension style name), 10, 20, 30 (definition point for all dimension types), 11, 21, 31 (middle point of dimension text), 12, 22, 32 (dimension block translation vector), 70 (Dimension type), 1 (dimension text explicitly entered by the user. If null or "<>", the dimension measurement is drawn as the text, if " " [one blank space], the text is suppressed. Anything else is drawn as the text). 13, 23, 33 (definition point for linear and angular dimensions), 14, 24, 34 (definition point for linear and angular dimensions), 15, 25, 35 (definition point for diameter, radius, and angular dimensions), 16, 26, 36 (point defining dimension arc for angular dimensions), 40 (leader length for radius and diameter dimensions), 50 (angle of rotated, horizontal, or vertical linear dimensions).

The dimension type (group code 70) is an integer-coded field with the following values:

Table 11-17. Group 70 integer codes for Dimension entity

Group	**Description**
0	Rotated, horizontal, or vertical
1	Aligned
2	Angular
3	Diameter
4	Radius
5	Angular 3-point
6	Ordinate
64	Ordinate type. This is a bit value (bit 7) used only with integer value 6. If set, ordinate is X-type; if not set, ordinate is Y-type
128	This is a bit value (bit 8) added to the other group 70 values if the dimension text has been positioned at a user-defined location rather than at the default location

In addition, all dimension types have an optional group (code 51) that indicates the horizontal direction for the Dimension entity. This determines the orientation of dimension text and dimension lines for horizontal, vertical, and rotated linear dimensions. The group value is the negative of the Entity Coordinate Systems (ECS) angle of the UCS X axis in effect when the Dimension was drawn. The X axis of the UCS in effect when the Dimension was drawn is always parallel to the XY plane for the Dimension's ECS, and the angle between the UCS X axis and the ECS X axis is a single 2D angle. The value in group 51 is the angle from horizontal (the effective X axis) to the ECS X axis. Entity Coordinate Systems (ECS) are described later in this section.

Linear dimension types with an oblique angle have an optional group (code 52). When added to the rotation angle of the linear dimension (group code 50) this gives the angle of the extension lines. The optional group code 53 is the rotation angle of the dimension text away from its default orientation (the direction of the dimension line).

For all dimension types, the following groups represent 3D WCS points:

10, 20, 30
13, 23, 33
14, 24, 34
15, 25, 35

For all dimension types, the following groups represent 3D ECS points:

11, 21, 31
12, 22, 32
16, 26, 36

<u>Linear</u>

(13,23,33) The point used to specify the first extension line.

(14,24,34) The point used to specify the second extensionline.

(10,20,30) The point used to specify the dimension line.

Refer to figure 11-1 on page 267 of the "AutoCAD Customization Manual"

Angular

(13,23,33) and (14,24,34) The endpoints of the first extension line.

(10,20,30) and (15,25,35) The endpoints of the second extension line.

(16,26,36) The point used to specify the dimension line arc.

Refer to figure 11-2 on page 267 of the "AutoCAD Customization Manual"

Angular

(15,25,35) (3-point) (13,23,33) The endpoints of the first extension line.
(13,23,33) The endpoints of the first extension line.
(14,24,34) The endpoints of the second extension line.
(10,20,30) The point used to specify the dimension line arc.
Refer to figure 11-3 on page 267 of the "AutoCAD Customization Manual"

Diameter
(15,25,35) The point used to pick the circle/arc to dimension.
(10,20,30) The point on that circle directly across from the pick point.
Refer to figure 11-4 on page 268 of the "AutoCAD Customization Manual"

Radius
(15,25,35) The point used to pick the circle/arc to dimension.
(10,20,30) The center of that circle.
Refer to figure 11-5 on page 268 of the "AutoCAD Customization Manual"

Ordinate
(13,23,33) The point used to select the feature.
(14,24,34) The point used to locate the leader end point.
Refer to figure 11-6 on page 268 of the "AutoCAD Customization Manual"

Entity Coordinate Systems (ECS)

To save space in the drawing database (and in the DXF file), the points associated with each entity are expressed in terms of the entity's own Entity Coordinate System (ECS). The Entity Coordinate System allows AutoCAD to use a much more compact means of representation for entities. With ECS, the only additional information needed to describe the entity's position in 3D space is the 3D vector describing the Z axis of the ECS, and the elevation value.

For a given Z axis (or extrusion) direction, there are an infinite number of coordinate systems, defined by translating the origin in 3D space and by rotating the X and Y axes around the Z axis. However, for the same Z axis direction, there is only one Entity Coordinate System. It has the following properties:

o Its origin coincides with the WCS origin.
o The orientation of the X and Y axes within the XY plane are calculated in an arbitrary,
   but consistent manner. AutoCAD performs this calculation using the arbitrary axis algorithm
   (described later).

For some entities, the ECS is equivalent to the World Coordinate System and all points (DXF groups 10 - 37) are expressed in World coordinates. See the following table.

Table 11-18. Coordinate systems associated with an entity type

Entities	**Notes**
Line, Point, 3DFace,	These entities do not lie in a
3D Polyline, 3D Vertex,	particular plane. All points are
3D Mesh, 3D Mesh vertex	expressed in World coordinates. Of these entities, only Lines and Points can be extruded; their extrusion direction can differ from the World Z axis
Circle, Arc, Solid, Trace,	These entities are planar in
Text, Attrib, Attdef,	nature. All points are expressed
Shape, Insert, 2D Polyline,	in Entity coordinates. All of these
2D Vertex	entities can be extruded; their extrusion direction can differ from the World Z axis
Dimension	Some of a Dimension's points are expressed in WCS, and some in ECS
Viewport	Expressed in World coordinates
Others	The remaining entities have no point data and their coordinate systems are therefore irrelevant

Once AutoCAD has established the ECS for a given entity, here's how it works:

o The elevation value stored with an entity indicates how far along the Z axis to shift the XY plane from the WCS origin to make it coincide with the plane that the entity is in. How much of this is the user-defined elevation is unimportant.

o Any 2D points describing the entity that were entered through the UCS are transformed into the corresponding 2D points in the ECS, which (more often than not) is shifted and rotated with respect to the UCS.

These are a few ramifications of this process:

o You cannot reliably find out what UCS was in effect when an entity was acquired.

o When you enter the XY coordinates of an entity in a given UCS and then do a DXFOUT, you probably won't recognize those XY coordinates in the DXF file. You'll have to know the method by which AutoCAD calculates the X and Y axes in order to work with these values.

o The elevation value stored with an entity and output in DXF files will be a sum of the Z-coordinate difference between the UCS XY plane and the ECS XY plane, and the elevation value that the user specified at the time the entity was drawn.

Arbitrary Axis Algorithm

The arbitrary axis algorithm is used by AutoCAD internally to implement the arbitrary but consistent generation of Entity Coordinate Systems for all entities except Lines, Points, 3D Faces, and 3D Polylines, which contain points in World coordinates.

Given a unit-length vector to be used as the Z axis of a coordinate system, the arbitrary axis algorithm generates a corresponding X axis for the coordinate system. The Y axis follows by application of the right-hand rule.

The method is to examine the given Z axis (also called the normal vector) and see if it is close to the positive or negative World Z axis. If it is, cross the World Y axis with the given Z axis to arrive at the arbitrary X axis. If not, cross the World Z axis with the given Z axis to arrive at the arbitrary X axis. The boundary at which the decision is made was chosen to be both inexpensive to calculate and completely portable across machines. This is achieved by having a sort of "square" polar cap, the bounds of which is 1/64, which is precisely specifiable in 6 decimal fraction digits and in 6 binary fraction bits.

In mathematical terms, the algorithm does the following (all vectors are assumed to be in 3D space, specified in the World Coordinate System):

Let the given normal vector be called N.
Let the World Y axis be called Wy, which is always (0,1,0).
Let the World Z axis be called Wz, which is always (0,0,1).

We are looking for the arbitrary X and Y axes to go with the normal N. They'll be called Ax and Ay. N could also be called Az (the arbitrary Z axis):

```
If (abs (Nx) < 1/64) and (abs (Ny) < 1/64) then
        Ax = Wy  N (where "" is the cross-product operator).
Otherwise,
        Ax = Wz  N.

Scale Ax to unit length.

The method of getting the Ay vector would be:
Ay = N  Ax. Scale Ay to unit length.
```

Extended Entity Data
==============

Extended entity data is created by applications such as the Advanced Modeling Extension (AME), or by routines written with AutoLISP or ADS. Extended entity data is also produced by creating PostScript output with PSOUT. If an entity contains extended data, it follows the entity's normal definition data.

The group codes 1000 through 1071 describe extended entity data. The following is an example of an entity containing extended entity data in DXF format.

```
0
INSERT
 8
0
 5
5
F11
15             -- Normal entity definition data.
 2
BLOCK_A
 10
0.0
 20
```

```
0.0
 30
0.0
1001
AME_SOL
1002
{
1070
 0
1071
 1.95059E+06
1070
 519
1010
2.54717
1020
2.122642           -- Extended entity data.
1030
2.049201
1005
ECD
1005
EE9
1005
0
1040
0.0
1040
1.0
1000
MILD_STEEL
```

Figure 11-7. Example of extended entity data

Organization of Extended Entity Data
========================

As you can see in the above example, group code 1001 indicates the beginning of extended entity data. This is followed by one or more 1000 group codes. Application names are string values (in the example, the application name is AME_SOL). In contrast to normal entity data, the same group code can appear multiple times, and order is important.

Extended entity data are grouped by registered application name, and each registered application's group begins with a 1001 group code with the registered application name as the string value. Registered application names correspond to APPID symbol table entries, which are essentially placeholders for registered application names.

An application can use as many APPID names as needed, although one will often suffice. APPID names are permanent, although they can be purged if they aren't currently used in the drawing.

Each APPID name can have no more than one data group attached to each entity. Within an application's group, the sequence of extended entity data groups and their meaning is defined by the application.

NOTE: PostScript images and PostScript fill requests for Polylines are stored in the AutoCAD database as extended entity data belonging to the AUTOCAD_POSTSCRIPT_FIGURE application.

As the example in the previous figure shows, the group codes for extended entity data begin at 1000 and currently extend to 1071. The following list of extended entity data group codes are supported by AutoCAD, which maintains and manipulates their values as described:

Table 11-19. extended entity data group codes and descriptions

Entity Name	Group code	Description
String	1000	Strings in extended entity data can be up to 255 bytes long (with the 256th byte reserved for the null character)
Application name	1001 also a string value	Application names can be up to 31 bytes long (the 32d byte is reserved for the null character). Use of application names is described in more detail later in this section CAUTION: Do not add a 1001 group into your extended entity data, as AutoCAD will assume it is the beginning of a new application extended entity data group
Control string	1002	An extended data control string can be either "{"or "}": these braces enable applications to organize their data by subdividing the data into lists. The left brace begins a list, and a right brace terminates the most recent list; lists can be nested When AutoCAD reads the extended entity data for a particular application, it checks to ensure that braces are balanced correctly
Layer name	1003	Name of the layer associated with the extended entity data
Binary data	1004	Binary data is organized into variable-length chunks.The maximum length of each chunk is 127 bytes. Binary data is represented as a string of hexadecimal digits, two per binary byte, in ASCII DXF files
Database	1005	Handles of entities in the
handle		drawing database NOTE: When a drawing with handles and extended entity data handles is imported into another drawing using INSERT, INSERT *, XREF Bind, XBIND, or partial DXFIN, the extended entity data handles are translated in the same manner as their corresponding entity handles, thus maintaining their binding.

		This is also done in the EXPLODE Block operation, or for any other AutoCAD operation. When AUDIT detects an extended entity data handle that doesn't match the handle of an entity in the drawing file, it is considered an error. If AUDIT is fixing entities, it sets the handle to 0.
3 reals	1010, 1020, 1030	Three real values, in the order X, Y, Z. They can be used as a point or vector record. AutoCAD never alters their value
World space position	1011, 1021, 1031	Unlike a simple 3D point,the World space coordinates are moved, scaled, rotated, and mirrored along with the parent entity to which the extended data belongs. The world space position is also stretched when the STRETCH command is applied to the parent entity and this point lies within the select window
World space displacement	1012, 1022, 1032	Also a 3D point that is scaled, rotated, and mirrored along with the parent (but not moved or stretched)
World direction	1013, 1023, 1033	Also a 3D point that is rotated and mirrored along with the parent (but not moved, scaled, or stretched).
Real	1040	A real value
Distance	1041	A real value that is scaled along with the parent entity
Scale factor	1042	Also a real value that is scaled along with the parent. The difference between a distance and a scale factor is application-defined
Integer	1070	A 16-bit integer (signed or unsigned)
Long	1071	A 32-bit signed (long) integer

For more information on extended entity data and the APPID table, refer to the "AutoCAD Development System Programmer's Reference" and the "AutoLISP Programmer's Reference."

Writing DXF Interface Programs

Writing a program that communicates with AutoCAD via the DXF mechanism often appears far more difficult than it really is. The DXF file contains a seemingly overwhelming amount of information, and examining a DXF file manually may lead to the conclusion that the task is hopeless.

However, the DXF file has been designed to be easy to process by program, not manually. The format was intentionally constructed to make it easy to ignore information you don't need while easily reading the information you do need. Just remember to handle the groups in any order and ignore any group you don't care about.

As an example, the following is a Microsoft BASIC program that reads a DXF file and extracts all the Line entities from the drawing (ignoring lines that appear inside Blocks). It prints the endpoints of these lines on the screen. As an exercise you might try entering this program into your computer, running it on a DXF file from one of your drawings, then enhancing it to print the center point and radius of any circles it encounters. This program is not put forward as an example of clean programming technique nor the way a general DXF processor should be written; it is presented as an example of just how simple a DXF-reading program can be.

```
1000  REM
1010  REM Extract lines from DXF file
1020  REM
1030  G1% = 0
1040  LINE INPUT "DXF file name: "; A$
1050  OPEN "i", 1, A$ + ".dxf"
1060  REM
1070  REM Ignore until section start encountered
1080  REM
1090  GOSUB 2000
1100  IF G% <> 0 THEN 1090
1110  IF S$ <> "SECTION" THEN 1090
1120  GOSUB 2000
1130  REM
1140  REM Skip unless ENTITIES section
1150  REM
1160  IF S$ <> "ENTITIES" THEN 1090
1170  REM
1180  REM Scan until end of section, processing LINEs
1190  REM
1200  GOSUB 2000
1210  IF G% = 0 AND S$ = "ENDSEC" THEN 2200
1220  IF G% = 0 AND S$ = "LINE" THEN GOSUB 1400 : GOTO 1210
1230  GOTO 1200
1400  REM
1410  REM Accumulate LINE entity groups
1420  REM
1430  GOSUB 2000
1440  IF G% = 10 THEN X1 = X : Y1 = Y : Z1 = Z
1450  IF G% = 11 THEN X2 = X : Y2 = Y : Z2 = Z
1460  IF G% = 0 THEN PRINT "Line from (";X1;",";Y1;",";Z1;") to
    (";X2;",";Y2;",";Z2;")":RETURN
1470  GOTO 1430
2000  REM
2010  REM Read group code and following value
2020  REM For X coordinates, read Y and possibly Z also
2030  REM
2040  IF G1% < 0 THEN G% = -G1% : G1% = 0 ELSE INPUT #1, G%
2050  IF G% < 10 OR G% = 999 THEN LINE INPUT #1, S$ : RETURN
2060  IF G% >= 38 AND G% <= 49 THEN INPUT #1, V : RETURN
```

```
2080  IF G% >= 50 AND G% <= 59 THEN INPUT #1, A : RETURN
2090  IF G% >= 60 AND G% <= 69 THEN INPUT #1, P% : RETURN
2100  IF G% >= 70 AND G% <= 79 THEN INPUT #1, F% : RETURN
2110  IF G% >= 210 AND G% <= 219 THEN 2130
2115  IF G% >= 1000 THEN LINE INPUT #1, T$ : RETURN
2120  IF G% >= 20 THEN PRINT "Invalid group code";G% : STOP
2130  INPUT #1, X
2140  INPUT #1, G1%
2150  IF G1% <> (G%+10) THEN PRINT "Invalid Y coord code"; G1% :
    STOP
2160  INPUT #1, Y
2170  INPUT #1, G1%
2180  IF G1% <> (G%+20) THEN G1% = -G1% ELSE INPUT #1, Z
2190  RETURN
2200  CLOSE 1
```

Writing a program that constructs a DXF file is more difficult, because you must maintain consistency within the drawing in order for AutoCAD to find the file acceptable. AutoCAD lets you omit many items in a DXF file and still obtain a usable drawing. The entire HEADER section can be omitted if you don't need to set any header variables. Any of the tables in the TABLES section can be omitted if you don't need to make any entries, and the entire TABLES section can be dropped if nothing in it is required. If you define any linetypes in the LTYPE table, this table must appear before the LAYER table. If no Block Definitions are used in the drawing, the BLOCKS section can be omitted. If present, however, the BLOCKS section must appear before the ENTITIES section. Within the ENTITIES section, you can reference layer names even though you haven't defined them in the LAYER table. Such layers are automatically created with color 7 and the CONTINUOUS linetype. The EOF item must be present at the end-of-file.

The following Microsoft BASIC program constructs a DXF file representing a polygon with a specified number of sides, leftmost origin point, and side length. This program supplies only the ENTITIES section of the DXF file, and places all entities generated on the default layer 0. This may be taken as an example of a minimum DXF generation program. Since this program doesn't create the drawing header, the drawing limits, extents, and current view will be invalid after performing a DXFIN on the drawing generated by this program. You can do a ZOOM E to fill the screen with the drawing generated. Then adjust the limits manually.

```
1000  REM
1010  REM Polygon generator
1020  REM
1030  LINE INPUT "Drawing (DXF) file name: "; A$
1040  OPEN "o", 1, A$ + ".dxf"
1050  PRINT #1, 0
1060  PRINT #1, "SECTION"
1070  PRINT #1, 2
1080  PRINT #1, "ENTITIES"
1090  PI = ATN(1) * 4
1100  INPUT "Number of sides for polygon: "; S%
1110  INPUT "Starting point (X,Y): "; X, Y
1120  INPUT "Polygon side: "; D
1130  A1 = (2 * PI) / S%
1140  A = PI / 2
1150  FOR I% = 1 TO S%
1160  PRINT #1, 0
1170  PRINT #1, "LINE"
1180  PRINT #1, 8
1190  PRINT #1, "0"
1200  PRINT #1, 10
1210  PRINT #1, X
1220  PRINT #1, 20
1230  PRINT #1, Y
```

```
1240   PRINT #1, 30
1250   PRINT #1, 0.0
1260   NX = D * COS(A) + X
1270   NY = D * SIN(A) + Y
1280   PRINT #1, 11
1290   PRINT #1, NX
1300   PRINT #1, 21
1310   PRINT #1, NY
1320   PRINT #1, 31
1330   PRINT #1, 0.0
1340   X = NX
1350   Y = NY
1360   A = A + A1
1370   NEXT I%
1380   PRINT #1, 0
1390   PRINT #1, "ENDSEC"
1400   PRINT #1, 0
1410   PRINT #1, "EOF"
1420   CLOSE 1
```

The DXFIN command is relatively forgiving with respect to the format of data items. As long as a properly formatted item appears on the line on which the data is expected, DXFIN will accept it (of course, string items should not have leading spaces unless these are intended to be part of the string). This program takes advantage of this flexibility in input format, and does not try to generate a file appearing exactly like one generated by AutoCAD.

In the case of error loading a DXF file using DXFIN, AutoCAD reports the error with a message indicating the nature of the error and the last line processed in the DXF file before the error was detected. This may not be the line on which the error occurred, especially in the case of errors such as omission of required groups.

Binary Drawing Interchange Files
****

The ASCII DXF file format described in the preceding sections of this chapter is a complete representation of an AutoCAD drawing in an ASCII text form easily processed by other programs. In addition, AutoCAD can produce or read a binary form of the full DXF file, and accepts limited input in another binary file format. These binary files are described in the following sections.

Binary DXF Files
================

The DXFOUT command provides a Binary option that writes binary DXF files. Such a file contains all of the information present in an ASCII DXF file, but in a more compact form that takes, typically, 25% less file space and can be read and written more quickly (typically 5 times faster) by AutoCAD. Unlike ASCII DXF files, which entail a trade-off between size and floating-point accuracy, binary DXF files preserve all of the accuracy in the drawing database. AutoCAD Release 10 was the first version to support this form of DXF file; it cannot be read by older versions.

A binary DXF file begins with a 22-byte sentinel consisting of:

```
AutoCAD Binary DXF<CR><LF><SUB><NUL>
```

Following the sentinel are (group, value) pairs as in an ASCII DXF file, but represented in binary form. The group code is a single-byte binary value, and the value that follows is one of the following:

o A two-byte integer with the least-significant byte first and the most-significant byte last.

- An eight-byte IEEE double precision floating-point number stored with the least-significant byte first and the most-significant byte last.

- An ASCII string terminated by a zero (NUL) byte.

The type of the datum following a group is determined from the group code according to the same rules used in decoding ASCII DXF files. Translation of angles to degrees, and dates to fractional Julian date representation, is performed for binary files as well as for ASCII DXF files. The comment group, 999, is not used in binary DXF files.

Extended entity data group codes are represented in Binary DXF as a single byte with the value 255, followed by a 2-byte integer value containing the actual group code, followed by the actual value.

Extended entity data long (group code 1071) values occupy 4 bytes of data. Extended entity data binary chunks (group code 1004) are represented as a single-byte, unsigned integer length, followed by the specified number of bytes of chunk data. For example, to transfer an extended entity data long group, the following values would appear, occupying 1, 2, and 4 bytes respectively:

255	Escape group code.
1071	True group code.
999999	Value for the 1071 group code.

DXFOUT writes binary DXF files with the same file type (.dxf) as for ASCII DXF files. The DXFIN command automatically recognizes a binary file (by means of its sentinel string) and loads the file. There is no need for you to identify it as a binary file.

If DXFIN encounters an error in a binary DXF file, it reports the byte address within the file where the error was detected.

Binary Drawing Interchange (DXB) Files
****

The DXF file formats described earlier in this chapter are complete representations of an AutoCAD drawing that can be written and read by AutoCAD and other programs. However, AutoShade and programs executed via the external commands facility (chapter 3) often need to supply simple geometric input to AutoCAD. For these purposes, another file format even more compact than the binary DXF format is supported. This format, called DXB (for drawing interchange binary) is limited in the entities it can represent.

DXBIN Command
=============

To load a DXB file produced by a program such as AutoShade, enter the DXBIN command:

```
Command: dxbin
```

When AutoCAD prompts you, respond with the name of the file you want to load. You don't need to include a file type; .dxb is assumed.

DXB File Format
============

IMPORTANT: This information is for experienced programmers and is subject to change without notice.

The format of a DXB file is as follows:

```
Header: "AutoCAD DXB 1.0" CR LF ^Z NUL   (19 bytes)
```

Data: Zero or more data records
Terminator: NUL (1 byte)

Each data record begins with a single byte identifying the record type, followed by data items. The data items have various forms of representation and encoding. In the descriptions following, each data item is prefixed with a letter and a hyphen. The meaning of the letter codes is as follows:

w- 16-bit integer, byte reversed in the standard 80x86 style (least- significant byte first, most-significant byte second).

f- IEEE 64-bit floating-point value stored with lsb first, msb last (as stored by an 80x87).

l- 32-bit integer with the bytes reversed 80x86 style.

n- Number which may be either a 16-bit integer or a floating-point number depending on the most recent setting of the number mode data item. The number mode defaults to 0, signifying integers. If set to 1, all n- items will be read as floating-point.

u- Item which is either a 32-bit integer or a floating-point number depending on the most recent number mode setting. If a 32-bit integer, the value is scaled by multiplying it by 65536 (2^16). If a floating-point value, no scaling is applied.

a- Item representing an angle. If number mode is integer, this is a 32-bit integer representing an angle in units of millionths of a degree (range 0 to 360,000,000). If a floating-point number, represents degrees.

In the following table, the lengths include the item-type byte and assume the number mode is set to zero (integer mode). If number mode is floating-point, add 6 bytes to the length for each n- item present and 4 bytes for each a-, or u- item present.

Table 11-20. Byte length for item types

Item type	**Code** (decimal)	**Data items**	**Length** (bytes)
Line	1	n-fromx n-fromy n-tox n-toy n-fromx n-fromy n-fromz n-tox n-toy n-toz	13
Point	2	n-x n-y	5
Circle	3	n-ctrx n-ctry n-rad	7
Arc	8	n-ctrx n-ctry n-rad a-starta a-enda	19
Trace	9	n-x1 n-y1 n-x2 n-y2 n-x3 n-y3 n-x4 n-y4	17
Solid	11	n-x1 n-y1 n-x2 n-y2 n-x3 n-y3 n-x4 n-y4	17
Seqend	17	(none)	1
Polyline	19	w-closureflag	3
Vertex	20	n-x n-y	5

Item type	**Code** (decimal)	**Data items**	**Length** (bytes)
3Dface	22	n-x1 n-y1 n-z1 n-x2 n-y2 n-z2 n-x3 n-y3 n-z3 n-x4 n-y4 n-z4	25
Scale Factor	128	f-scalefac	9
New Layer	129	"layername" NUL	layername length + 2
Line Extension	130	n-tox n-toy	5
Trace Extension	131	n-x3 n-y3 n-x4 n-y4	9
Block Base	132	n-bx n-by	5
Bulge	133	u-2h/d	5
Width	134	n-startw n-endw	5
Number Mode	135	w-mode	3
New Color	136	w-colornum	3
3Dline	137	n-tox n-toy n-toz	7

The **Line Extension item** extends the last line or line extension from its To point to a new To point:.

The **Trace Extension item** similarly extends the last trace solid, or Trace Extension from its x3,y3-x4,y4 ending line to a new x3,y3--x4,y4 line.

The **Scale Factor** is a floating-point value by which all integer coordinates are multiplied to obtain the floating-point coordinates used by the actual entities. The initial scale factor when a file is read is 1.0.

The **New Layer item** creates a layer if none exists, giving the new layer the same defaults as the LAYER New command, and sets that layer as the current layer for subsequent entities. At the end of the DXB file load, the layer in effect before the command is restored.

The **Block Base item** specifies the base (origin) point of a created Block. The Block base must be defined before the first entity record is encountered. If DXB is not defining a Block, this specification will be ignored.

A **Polyline** consists of straight segments of fixed width connecting the vertices, except as overridden by the Bulge and Width items described below. The closure flag should be 0 or 1; if it is 1, then there is an implicit segment from the last vertex (immediately before the Seqend) to the first vertex.

A **Bulge item**, encountered between two Vertex items (or after the last Vertex of a closed Polyline), indicates that the two vertices are connected by an arc rather than a straight segment. If the line segment connecting the vertices would have length d, and the perpendicular distance from the midpoint of that segment to the arc is h, then the magnitude of the Bulge is (2 * h / d). The sign is negative if the arc from the first vertex to the second is clockwise. A semicircle thus has a bulge of 1 (or -1). If the number mode is 0 (integer), Bulge items are scaled by 2 16. If the number mode has been set to floating-point, then the floating-point value supplied is just 2*h/d (not scaled).

The **Width item** indicates the starting and ending widths of the segment (straight or curved) connecting two vertices. This width stays in effect until the next width item or the Seqend. If there is a Width item between the Polyline item and the first Vertex, it is stored as a default width for the Polyline; this saves considerable database space if the Polyline has several segments of this width.

The **Number Mode** item controls the mode of items with types given in the table above as n-, a-, or u-. If the value supplied is zero, these values will be integers, otherwise floating-point. The storage and implicit scaling conventions for these values in both modes are described earlier.

Lines share the same cells to remember the last to-point, so you shouldn't mix extension groups for the two entities without an initial group before the extension. There is no extension group for 3Dfaces, as there's no obvious edge to extend from.

The New Color group specifies the color for subsequent entities in the DXB file. The w-colornum word argument is in the range from 0 to 256. 0 means color by block, 1-255 are the standard AutoCAD colors, and 256 means color by layer. A color outside the range from 0 to 256 sets the color back to the current entity color (you can do this deliberately, and it can be quite handy). The initial entity color of material added by DXBIN is the current entity color.

All points specified in the DXB file are interpreted in terms of the current UCS at the time the DXBIN command is executed.

Writing DXB Files
==============

There is no direct AutoCAD command to write a DXB file, but the special ADI plotter driver can write such a file. If you want to create a DXB file from an AutoCAD drawing, configure the ADI plotter and select its DXB file output option.

Initial Graphics Exchange Specification (IGES) Files
****

Using the commands described in this section, you can instruct AutoCAD to read and write IGES-format interchange files.

NOTE: The format of IGES files and the mapping performed to translate between AutoCAD drawing information and IGES are described in the separate AutoCAD/IGES Interface Specifications document.

IGESOUT Command
===============

You can generate an Initial Graphics Exchange Specification (IGES) interchange file from an existing AutoCAD drawing by means of the IGESOUT command:

```
Command: igesout
```

When AutoCAD prompts you, respond with a filename or press 5 to accept the default.

The default name for the output file is the same as that of the current drawing, but with a file type of .igs. If you specify an explicit filename without including a file type, .igs is assumed. If a file with the same name already exists, it is deleted. If FILEDIA is on, and a file with the same name already exists, AutoCAD tells you; allowing you to OK or cancel the deletion.

IGESIN Command
==============

An IGES interchange file can be converted into an AutoCAD drawing by means of the IGESIN command:

```
Command: igesin
```

When AutoCAD prompts you, respond with the name of the IGES file to be loaded.

To load a complete IGES file, you must use IGESIN in an empty drawing, before any entities have been drawn and before any additional Block definitions, layers, linetypes, text styles, named views, named coordinate systems, or named viewport configurations have been created.

NOTE: If the drawing you are using as a prototype is not empty, you
might find it helpful to open a new drawing using the No Prototype... button of the Create New Drawing dialogue box, as described in chapter 4 of the AutoCAD Reference Manual. You should also be aware that some third-party applications include an acad.lsp or .mnl file that modifies your drawing upon startup.

If a serious error is encountered, the input process stops and an error message is displayed reporting where the error was found. The partial drawing is not discarded.