IFF ‘DOC ’

IFF DOC FILE FORMAT

OVERVIEW

IFF DOC is a file format “standard” for text, graphics and documents that was created by Soft-Logik Publishing Corporation. Changes can only be made to this format by Soft-Logik Publishing.

This document contains information about the structure and format of Soft-Logik Publishing's IFF DOC file format which is used by PageStream and other Soft-Logik Publishing products. This information is intended to assist developers who need to know about the internal structure of these files to create filters that load and/or save IFF DOC files.

This document is subject to change without notice. In fact, we guarantee that it will change in the future. If you write a filter that loads IFF DOC files, ensure that is flexible enough to skip data types that are added later. If you write a filter that writes IFF DOC files, the files must be compatible with PageStream, which is the ultimate test of IFF DOC compatibility.

Permission is hereby granted to use this file in any and all applications free of royalties. Modification of the structures and/or defines included in this file, and of this file itself, is not permitted under any circumstances without the express written consent of Soft-Logik Publishing Corporation.

Questions about this file format must be submitted electronically to tech@slpc.com.

IFF DOC, PageStream and Soft-Logik are trademarks or registered trademarks of Soft-Logik Publishing Corporation. The IFF DOC file format is copyright 1995 by Soft-Logik Publishing Corporation.

#define DATE ULONG
#define TIME ULONG
#define PGNUM ULONG
#define COORD LONG (1828800 dpi)
#define ANGLE LONG (.001 degrees)
#define SCALE WORD (.1%)
#define EMS WORD (.0001 em)
#define NSTR char[] standard null terminiated C string
VSTR is UWORD length + UBYTE string[length] of Unicode VNUM sequence of data.

Simple structure overview. This sample structure leaves out many chunks and shows but a single combination. Each FORM should later list which FORMS/chunks are valid within them and where they are valid in.

FORM ‘DOC ’
    FORM ‘MPAG’
        FORM ‘LMPG’
            FORM ‘ILUS’
        FORM ‘RMPG’
            FORM ‘ILUS’
    FORM ‘CHAP’
        FORM ‘CHAP’
        Chunk ‘REVS’
        FORM ‘CTXT’
        FORM ‘CTXT’
        Chunk ‘CTAG’
        FORM ‘PAGE’
            FORM ‘ILUS’
                FORM ‘ILBM’
                FORM ‘CTXT’
        FORM ‘SPRD’
            FORM ‘PAGE’
            FORM ‘PAGE’
        FORM ‘PAGE’

Random notes to be kept and consumed into the greater as needed.

Kerning pair table, Track Tables, Hyphenation Tables

what about FORM ‘PROP’ for chunks like PGDM, GRID, VGUI, HGUI,CNUM,PNUM, etc?
what about Chunk ‘PREF’ - string prefs like DR2D. **Baseline Grid**?

Object coordinate relativity / sizeability for changing of pages inside vs outside and landsacpe vs portrait, etc.

doc creation data & doc open timer time? Print area for ILUS or whatever?
Hypertext Links

FORM 'DOC '

Valid inside FORMs:

<none at this time>

Valid FORMs inside:

CHAP	CTXT	MPAG	PAGE

Valid Chunks inside:

DOC	CNUM	PNUM	NAME
ANNO	AUTH	(C)	FILE
REVS	HLID	CTAG	

Chunk 'DOC '

ULONG	Flags
VSTR	Job # String

no flags defined yet

FORM 'CHAP'

Valid inside FORMs:

DOC	CHAP

Valid FORMs inside:

CHAP	CTXT	MPAG	PAGE

Valid Chunks inside:

DOC	CNUM	PNUM	NAME
ANNO	AUTH	(C)	FILE
REVS	HLID	CTAG	

Chunk 'CHAP'

Valid inside FORMs:

CHAP

ULONG	Flags	
PGNUM	Chapter Number	for non-sequencial unique ordering

#define	CHAP_USERNUMBER	`0x00000001`

Chunk 'CNUM'

Valid inside FORMs:

DOC CHAP

UWORD	Flags	
PGNUM	Starting Chapter Number	
PGNUM	Chapter Length	
VSTR	Blank Page MasterPage Name	
UWORD	Chapter Number Default Format	
VSTR	Chapter Number Default Language	
VSTR	Chapter Number Prefix	
#define	CNUM_STARTMASK	0x000f
#define	CNUM_STARTAUTO	0x0000
#define	CNUM_STARTEVEN	0x0001
#define	CNUM_STARTODD	0x0002
#define	CNUM_STARTUSER	0x0003
#define	CNUM_LENGTHMASK	0x00f0
#define	CNUM_LENGTHAUTO	0x0000
#define	CNUM_LENGTHEVEN	0x0010
#define	CNUM_LENGTHODD	0x0020
#define	CNUM_LENGTHUSER	0x0030

Chunk 'PNUM'

Valid inside FORMs:

DOC CHAP

UWORD	Flags	
PGNUM	Starting Page Number	
PGNUM	Page Length	
VSTR	Blank Page MasterPage Name	
UWORD	Page Number Default Format	
VSTR	Page Number Default Language	
VSTR	Page Number Prefix	
#define	PNUM_STARTMASK	0x000f
#define	PNUM_STARTAUTO	0x0000
#define	PNUM_STARTEVEN	0x0001
#define	PNUM_STARTODD	0x0002
#define	PNUM_STARTUSER	0x0003
#define	PNUM_LENGTHMASK	0x00f0
#define	PNUM_LENGTHAUTO	0x0000
#define	PNUM_LENGTHEVEN	0x0010
#define	PNUM_LENGTHODD	0x0020
#define	PNUM_LENGTHUSER	0x0030

FORM 'MPAG'

Valid inside FORMs:

DOC	CHAP		

Valid FORMs inside:

SMPG	LMPG	CMPG	RMPG
CTXT	ILUS		

Valid Chunks inside:

MPAG	NAME	ANNO	AUTH
(C)	FILE	REVS	HLID

Chunk 'MPAG'

ULONG Flags

```
#define   MPAG_SINGLE     0x00000000
#define   MPAG_DOUBLE     0x00000003
#define   MPAG_SDMASK     0x0000000f
```

FORM 'SMPG' single masterpage
FORM 'LMPG' left masterpage
FORM 'CMPG' center masterpage
FORM 'RMPG' right masterpage

Valid inside FORMs:

DOC	CHAP		

Valid FORMs inside:

SMPG	LMPG	CMPG	RMPG
CTXT	ILUS		

Valid Chunks inside:

NAME	ANNO	AUTH	(C)
FILE	REVS	HLID	

these FORMs "wrap" the objects/grids/guides/etc for the "side" of the masterpage they represent.

FORM 'SPRD'

Valid inside FORMs:

DOC	CHAP		

Valid FORMs inside:

PAGE			

Valid Chunks inside:

NAME	ANNO	AUTH	(C)
FILE	REVS	HLID	SPRD

Chunk 'SPRD'

ULONG	Flags	
PGNUM	StartingPage	
PGNUM	EndingPage	
#define	SPRD_VERTICAL	0x00000001

used as a wrapper around several pages which make up a spread.
Never more than one deep, only SPRD chunk and PAGEs inside.

FORM 'PAGE'

Valid inside FORMs:

DOC	CHAP		

Valid FORMs inside:

CTXT	ILUS		

Valid Chunks inside:

NAME	ANNO	AUTH	(C)
FILE	REVS	HLID	PAGE

Chunk 'PAGE'

ULONG	Flags
PGNUM	Page Number
VSTR	MasterPage (or 0 if default or previos defined MP)

#define	PAGE_SHOWMASTERPAGE	0x00000001
#define	PAGE_HIDEMASTERPAGE	0x00000002
#define	PAGE_MASTERPAGEINFRONT	0x00000004
#define	PAGE_MASTERPAGEINBACK	0x00000008
#define	PAGE_NOPRINT	0x00000010
#define	PAGE_GREEKDISPLAY	0x00000020

Chunk 'PGDM'

Valid inside FORMs:

MPAGE xMPG PAGE ILUS

ULONG	Flags
COORD	Left
COORD	Top
COORD	Right
COORD	Bottom
COORD	Left Bleed
COORD	Top Bleed
COORD	Right Bleed
COORD	Bottom Bleed

#define PGDM_LANDSCAPE `0x00000001`

Chunk 'MRGN'

Valid inside FORMs:

MPAGE xMPG PAGE ILUS

ULONG	Flags
COORD	Inside Margin
COORD	Top Margin
COORD	Outside Margin
COORD	Bottom Margin
UWORD	Number of Columns
COORD	Column Gutter

no flags defined yet

Chunk 'GRID'

Valid inside FORMs:

MPAGE xMPG PAGE ILUS

ULONG	Flags
COORD	Grid Snap Width
COORD	Grid Snap Height
COORD	Grid Snap Left Offset
COORD	Grid Snap Top Offset
COORD	Grid Snap Width Range
COORD	Grid Snap Height Range
UWORD	Grid Display Horizontal Interval
UWORD	Grid Display Vertical Interval

#define GRID_SNAPSTRENGTH `0x00000001`

implementation note: grid points have precedence over guides

Chunk 'VGUI'

Chunk 'HGUI'

Valid inside FORMs:

MPAGE xMPG PAGE ILUS

ULONG	Flags	
COORD	Guide Snap Range	
struct	Guide[]	
	UBYTE	Type
	UBYTE	Flags
	COORD	Position

#define xGUI_SNAPSTRENGTH `0x00000001`

Generic Chunks

Chunk 'REVS'

Valid inside FORMs:

DOC	CHAP	MPAGE	xMPG
SPRD	PAGE	ILUS	CTXT
ILBM			

Punk 'RV'

UWORD	Type
DATE	Date
TIME	Time
UWORD	MajorVersionNumber
UWORD	MinorVersionNumber
VSTR	UserName
VSTR	Description

Where type can be:

#define	REVTYPE_CREATE	0x0001
#define	REVTYPE_MODIFY	0x0002
#define	REVTYPE_OPEN	0x0003
#define	REVTYPE_SAVE	0x0004

This chunk is valid in any FORM. It's used to track revisions to the structure/objects. What it tracks is the level it is in. If it's at the FORM DOC level, it's document revisions. If it's in FORM PAGE level, it's revisions at the PAGE level.

Chunk 'CTAG'

Valid inside FORMs:

DOC	CHAP	MPAG	xMPG
PAGE	ILUS	CTXT	

Punk 'TG'

UWORD	Type
UWORD	Flags
VSTR	TagName
VSTR	NextTagName
VSTR	TagData

#define	CTAG_PARAGRAPH	0x0001
#define	CTAG_CHARACTER	0x0002
#define	CTAG_OBJECT	0x0003
#define	CTAG_COLOR	0x0004
#define	CTAG_FPATTERN	0x0005
#define	CTAG_LPATTERN	0x0006
#define	CTAG_NONEDITABLE	0x0001
#define	CTAG_TOC	0x0002

Chunk 'VARI'

Valid inside FORMs:

DOC	CHAP	MPAG	xMPG
PAGE	ILUS	CTXT	

Punk 'VA'

UWORD	Type
UWORD	Flags
VSTR	Name
VSTR	Value

Chunk 'FILE'

Valid inside FORMs:

DOC	CHAP	MPAGE	xMPG
SPRD	PAGE	ILUS	CTXT
ILBM			

```
UWORD   Flags
UWORD   FileType
DATE    ModDate
TIME    ModTime
UWORD   File Name Length
char    File Name[File Name Length]

#define FILEFLAGS_EXTERNAL  0x00000001

#define FILETYPE_AMIGA      0x01
#define FILETYPE_MACINTOSH  0x02
#define FILETYPE_WINDOWSNT  0x03
#define FILETYPE_DOS        0x04
```

File Name uses local character set as defined by FileType.

Chunk 'NAME'

Valid inside FORMs:

DOC	CHAP	MPAGE	xMPG
SPRD	PAGE	ILUS	CTXT
ILBM			

```
STR     Unicode VNum -Name
```

Chunk 'ANNO'

Valid inside FORMs:

DOC	CHAP	MPAGE	xMPG
SPRD	PAGE	ILUS	CTXT
ILBM			

```
STR     Unicode VNum -Comment
```

Chunk 'AUTH'

Valid inside FORMs:

DOC	CHAP	MPAGE	xMPG
SPRD	PAGE	ILUS	CTXT
ILBM			

```
STR     Unicode VNum -Author
```

Chunk '(C) '

Valid inside FORMs:

DOC	CHAP	MPAGE	xMPG
SPRD	PAGE	ILUS	CTXT
ILBM			

```
STR     Unicode VNum -Copyright
```

Font Chunk

- Full Name String
- Family Name String
- Attributes String
- Desc (Serif, SanSerif, Block, etc?)
- Type (Adobe Type 1, CompuGraphic Intellifont, BitStream Speedo, etc..)
- character map (important!)
- kerning pair table
- composition table

This chunk would normally be to indicate which fonts the document uses, and any pertinate info for the document. Most programs would need to interpret the font anyway, and could derive most of this info that way.
Full Name is the complete name of the font. This would be the FontName in Adobe Type 1 fonts. (ex, Helvetica-Bold)
Family Name is the same for all fonts in the same family (ex, Helvetica)
Attributes is the styles specific to this font (ex, Bold)
characterset map would be mapping from internal numbers to Unicode. similar to Character Map chunk. from internal to Unicode

FORM ‘ILUS’

FORM ‘ILUS’

Valid inside FORMs:

xMPG	PAGE

Valid FORMs inside:

CTXT	IOBJ

Valid Chunks inside:

REVS	CTAG	NAME	ANNO
AUTH	(C)	FILE	HLID
IOBJ			

A wrapper of objects for the page or a standalong illustration. Will include chunks and stuff to define the page size if this was a stand alone drawing format.

FORM 'IOBJ' large objects

Valid inside FORMs:
ILUS WRAP
Valid FORMs inside:
WRAP ILBM EPS
Valid chunks inside:
REVS CTAG NAME ANNO
AUTH (C) FILE HLID
IOBJ

Chunk 'IOBJ'

valid inside FORMs:
ILUS IOBJ
valid punks inside:
OB RT TC TW
AT PI PS LY
GR DR TL
BX RB LN CI
AR PO CO PA
PD

Punk 'OB' basic object definitions

ULONG Flags
UWORD Type
COORD BBox.left
COORD BBox.top
COORD BBox.right
COORD BBox.bottom

```
#define OB_LOCKED          0x00000001
#define OB_CONSTRAINED     0x00000002
#define OB_NOPRINT         0x00000004
#define OB_FILLED          0x00000100
#define OB_STROKED         0x00000200
```

Punk 'RT' rotation

ULONG Flags
ANGLE Slant
ANGLE Twist
COORD RotationPoint.x
COORD RotationPoint.y

```
#define RT_ABOUTPOINT      0x00000001
```

Punk 'TC' text container

ULONG Flags
ULONG ContainerNumber
ULONG ArticleNumber
UWORD ColumnCount
COORD ColumnGutter

what about from/to strings
flags undefined (direction?)

Punk 'TW' text wrap

	ULONG	Flags
	UBYTE	Mode
	UBYTE	Shape
	COORD	Offset.left
	COORD	Offset.top
	COORD	Offset.right
	COORD	Offset.bottom

#define	IOBJTW_MODENONE	0x00
#define	IOBJTW_MODELEFT	0x01
#define	IOBJTW_MODERIGHT	0x02
#define	IOBJTW_MODEJUMP	0x03
#define	IOBJTW_MODEAROUND	0x04
#define	IOBJTW_MODEINSIDE	0x05

#define	IOBJTW_SHAPECONTOUR	0x00
#define	IOBJTW_SHAPEBBOX	0x01
#define	IOBJTW_SHAPEFENCE	0x02

Punk 'AT' attribute data (line/fill)

char[punk length] AttributeData

command string like CTAG. contains Fill/Stroke attribute info.

Punk 'TL' text link

ULONG	Flags
ULONG	MarkNumber
COORD	XOffset
COORD	YOffset

Punk 'FR' frame

ULONG	Flags
UWORD	Type
COORD	ContentXOffset
COORD	ContentYOffset
SCALE	ContentXScale
SCALE	ContentYScale
ANGLE	ContentSlant
ANGLE	ContentTwist

'IOBJ' Object Punks

Punk 'LY' layer

	UWORD	Flags

#define	LAYR_VISIBLE	0x0001
#define	LAYR_EDITABLE	0x0002
#define	LAYR_ACTIVE	0x0004
#define	LAYR_PRINTABLE	0x0100

Punk 'DR' drawing

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
COORD	OriginalWidth
COORD	OriginalHeight

Punk 'GR' group

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
COORD	OriginalWidth
COORD	OriginalHeight

Punk 'CX' complex

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
COORD	OriginalWidth
COORD	OriginalHeight

Punk 'PI' picture

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
COORD	OriginalWidth
COORD	OriginalHeight

FORM 'ILBM'

Punk 'PS ' eps

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
COORD	OriginalWidth
COORD	OriginalHeight

Punk 'BX' box

```
        UWORD   Flags
        COORD   X1
        COORD   Y1
        COORD   X2
        COORD   Y2
#define IOBJBX_TYPEMASK      0x000f
#define IOBJBX_TYPENORMAL    0x0000
#define IOBJBX_TYPEROUND     0x0001
#define IOBJBX_TYPESCALLOP   0x0002
#define IOBJBX_TYPEBEVEL     0x0003
#define IOBJBX_TYPEINSET     0x0004
```

Punk 'RB' rounded corner box

```
        UWORD   Flags
        COORD   X1
        COORD   Y1
        COORD   X2
        COORD   Y2
        COORD   XRadius
        COORD   YRadius
```

same flags as BX

Punk 'LN' line

```
        UWORD   Flags
        COORD   X1
        COORD   Y1
        COORD   X2
        COORD   Y2
```

Punk 'EL' ellipse

```
        UWORD   Flags
        COORD   Center.x
        COORD   Center.y
        COORD   XRadius
        COORD   YRadius
```

Punk 'AR' arc

```
        UWORD   Flags
        COORD   Center.x
        COORD   Center.y
        COORD   XRadius
        COORD   YRadius
        ANGLE   BAngle
        ANGLE   EAngle
#define IOBJAR_TYPEMASK      0x000f
#define IOBJAR_TYPEELLIPSE   0x0000
#define IOBJAR_TYPEARC       0x0001
#define IOBJAR_TYPEPIE       0x0002
```

Punk 'PO' regular polygon

UWORD	Flags
COORD	Center.x
COORD	Center.y
COORD	XRadius
COORD	YRadius
UWORD	Sides
ANGLE	OffsetAngle
EMS	SRadiusDeflection
EMS	SAngleDeflection

```
#define   IOBJPO_TYPEMASK      0x000f
#define   IOBJPO_TYPENORMAL    0x0000
#define   IOBJPO_TYPESTAR      0x0001
#define   IOBJPO_TYPEPUFFY     0x0002
#define   IOBJPO_TYPESCALLOP   0x0003
#define   IOBJPO_TYPEWAVY      0x0004
#define   IOBJPO_TYPEGEAR      0x0005
```

Punk 'GD' grid

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
UWORD	HDivisions
UWORD	VDivisions

Punk 'G3' 3D grid

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
COORD	X3
COORD	Y3
COORD	X4
COORD	Y4
UWORD	HDivisions
UWORD	VDivisions

Punk 'CO' column

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2

Punk 'PA' path

UWORD	Flags
COORD	X1
COORD	Y1
COORD	X2
COORD	Y2
COORD	OriginalWidth
COORD	OriginalHeight

Punk 'PD' path data

UWORD Flags

PD_TYPEMASK		0x000f
PD_MOVETO		0x0001
	COORD	X
	COORD	Y
PD_LINETO		0x0002
	COORD	X
	COORD	Y
PD_CURVETO		0x0003
	COORD	Control1.x
	COORD	Control1.y
	COORD	Control2.x
	COORD	Control2.y
	COORD	X
	COORD	Y
PD_ARCTO		0x0004
PD_ARCNTO		0x0005
	COORD	Center.x
	COORD	Center.y
	COORD	Radius.x
	COORD	Radius.y
	ANGLE	BAngle
	ANGLE	EAngle
	ANGLE	Slant
	ANGLE	Twist
PD_CLOSEPATH		0x0006
PD_FLAGISCLOSE		0x0080
PD_FLAGCORNER		0x0100
PD_FLAGCONSTRAIN		0x4000
PD_RELATIVE		0x8000

FORM ‘CTXT’

FORM ‘CTXT’

Valid inside FORMs:

DOC	CHAP	MPAG	xMPG
PAGE	ILUS	DRAW	

Valid FORMs inside:

none at this time

Valid chunks inside:

CTAG	ARTI	CSET	CTXT
AUTH	(C)	HLID	FILE

Chunk ‘ARTI’

ULONG	Flags
ULONG	ArticleNumber
ULONG	ContainerCount

no flags defined yet

Chunk ‘CSET’

Valid inside FORMs:

CTXT

Valid punks inside:

NM	UC	LI

Punk ‘NM’

NSTR	CSETName

Punk ‘UC’

UBYTE	CharSize;
UBYTE	UnicodeSize;
UWORD	NumberEntries;

where (source/destination) size is

0- vnum 1- 1 byte 2- 2 byte 3- 3 byte

4- 4 byte (hey, it sounds crazy, but ANSI / ISO was talking about it)

Punk ‘LI’

UBYTE	CharSize;
UBYTE	UnicodeSize;
UWORD	NumberEntries;

CharSize	Character number	
UBYTE	number of Unicode entries	
UnicodeSize*number of Unicode entries		Unicode entries

where (source/destination) type is same as for UC Punk.

Chunk ‘CTXT’

char	AttributeData[chunklength]

if a character is non-zero, it’s a display glyph. If it’s zero, then it’s the beginning of an attribute data command. See the Attribute Data section for command details.

Attribute Data

Attribute Data commands follow the format:
(BYTE) 0, (VNum) Command, (VNum) Flags, (VNum) Length, (Length) Data

where VNum is a number encoded as,

```
value=0
while (*byte>127)
        value<<7
        value+= *byte++ - 128 ;
value+=*byte++
```

SVNum is like VNum, only that bit 6 of the first byte indicates whether or not the number is negative. If this bit is set, then negate the number after processing it. Bit 7 always indicates whether or not to get another byte just like another VNum.

text flags are

```
#define  TFLAG_KEEPLAST          0x00        keep last occurance of command
#define  TFLAG_KEEPNONE          0x01
#define  TFLAG_KEEPALL           0x02
#define  TFLAG_UNUSED1           0x03
#define  TFLAG_KEEPMASK          0x03

#define  TFLAG_NOTWHITESPACE     0x00
#define  TFLAG_WHITESPACE        0x04
#define  TFLAG_BLACKSPACE        0x08
#define  TFLAG_WHITEMASK         0x0c

#define  TFLAG_ATTRIBUTE         0x10
```

Commands can not partially replace revious commands. A new commands will completly override any previous command of the same number.

TEXT COMMANDS

STYLETAG_PARAGRAPH	1	paragraph tag (Unicode VNum tag name string)
STYLETAG_CHARACTER	2	character tag (Unicode VNum tag name string)
STYLETAG_OBJECT	3	object tag (Unicode VNum tag name string)
STYLETAG_COLOR	4	color tag (Unicode VNum tag name string)
STYLETAG_FPATTERN	5	fill pattern tag (Unicode VNum tag name string)
STYLETAG_LPATTERN	6	line pattern tag (Unicode VNum tag name string)

This is a very important command. This command has a single parameter which is the tag name to find and execute in place. The tag name string is a VNum encoded Unicode string. It's length is defined by the cmd data length. The tag is defined in the chunk CTAG. The tag is recursive, and may call upon others to complete the tag data.

A blank string (command data length = 0) indicates the end of a tag range.

Attributes described inside of a range that is tagged overrides the tag, but initial placement of a tag will remove the underlying commands which are defined inside the tag. End of a tag doesn't mean much except for editing purposes and is meaning less for display.

TEXT_TAB 10 tab

Causes text to be moved over to the next tab position (see TEXT_TABRULER for more details).

TEXT_RINDENTTAB 11 right indent tab

Forces a right justified tab with its position set to the right most edge.

TEXT_NEWPARA 12 new paragraph

Will shift down amount specified for PARAGRAPH SPACING AFTER then amount PARAGRAPH SPACING BEFORE in preperation for the next paragraph. Standard Return should convert to this command.

TEXT_NEWLINE 13 new line

Will force text to continue at the beginning of the next line shifted down by LEADING amount. For standard newparagraph see TEXT_NEWPARA.

TEXT_EOC 14 forced end of column

Force text to continue at the top of the next container whether it be on the same page or some other page. If no other containers exists in this articles list, then the text is overflow.

TEXT_EOP 15 forced end of page

Force text to continue at the top of the next container found at the top of the next page. If no other containers exists in this articles list, then the text is overflow.

TEXT_NUMBER 16 number (vnum type, vnum typedatalen, typedata, vnum format, vnum languagelen, language)

The number is shown in place of the command in the current text attributes. If format is not present, it inserts the number in the default format for the article. Otherwise, possibilites are arabic, roman and spelt out in numerous languages (should use some standard numbering scheme for languages).

Where type is the number to show. Type can be:

#define NUMBER_PAGE	1	page# current container is on
#define NUMBER_PREVPAGE	2	page# of prev container (- if none?)
#define NUMBER_NEXTPAGE	3	page# of next container (- if none?)
#define NUMBER_FIRSTPAGE	4	page# of first container
#define NUMBER_LASTPAGE	5	page# of last container
#define NUMBER_PAGECOUNT	6	# of pages in chapter current container is in
#define NUMBER_PAGESTART	7	first page# of chapter current container is in
#define NUMBER_PAGEEND	8	last page# of chapter current container is in
#define NUMBER_CHAPTER	9	chapter# current container is in (- if no chapter)
#define NUMBER_CHAPTERCOUNT	10	# of siblings of chapter current container is in (- if no chapter)
#define NUMBER_CHAPTERSTART	11	first sibling chapter# of chapter current container is in (- if no chapter)
#define NUMBER_CHAPTEREND	12	last sibling chapter# of chapter current container is in (- if no chapter)
#define NUMBER_SUBCHAPTERCOUNT	13	# of children chapters of chapter current container is in
#define NUMBER_SUBCHAPTERSTART	14	# of first child chapter of chapter current container is in
#define NUMBER_SUBCHAPTEREND	15	# of last child chapter of chapter current container is in
#define NUMBER_COLUMN	16	container # in the link (starting at 1)?
#define NUMBER_COLUMNCOUNT	17	# of containers in the link?
#define NUMBER_PREVCOLUMN	18	prev container # in the link (starting at 1) (- if first container?)
#define NUMBER_NEXTCOLUMN	19	next container # in the link (starting at 1) (- if last container?)
#define NUMBER_SECTION	20	
#define NUMBER_PARAGRAPH	21	
#define NUMBER_LINE	22	
#define NUMBER_MARK	23	data is the TEXT_MARK # to display the page it is on.
#define NUMBER_INDEXMARK	24	data is the TEXT_INDEXMARK # to display the page it is on.

Where format is the format to display in:

#define NUMBER_DEFAULT	0
#define NUMBER_ARABIC	1
#define NUMBER_ROMANUPPER	2
#define NUMBER_ROMANLOWER	3
#define NUMBER_ALPHAUPPER	4
#define NUMBER_ALPHALOWER	5
#define NUMBER_LANGUAGE	6

TEXT_REVISION 17 number (vnum type, vnum formatlen, formatstring, vnum languagelen, language)

type is the revision “level” to display. formatstring is of the form ‘m’ for major, ‘n’ for minor.

Where type is:

#define REVISION_DOCUMENT	1
#define REVISION_CHAPTER	2
#define REVISION_PAGE	3
#define REVISION_ARTICLE	4

TEXT_DATE 18 date (vnum type, vnum typedatalen, typedata, vnum formatlen, formatstring, vnum languagelen, language)

Inserts the date in the text stream at the command's location in the current text attributes. Where update indicates when date should be updated. Date is the date to display in YYYYMMDD format. Format is an ascii format string for formating.
Where update can be:

```
#define  DATE_CUSTOM              0
#define  DATE_CREATED             1
#define  DATE_MODIFIED            2
#define  DATE_SAVED               3
#define  DATE_LOADED              4
#define  DATE_PRINTED             5
#define  DATE_DOCUMENTREVISION    6
#define  DATE_CHAPTERREVISION     7
#define  DATE_PAGEREVISION        8
#define  DATE_ARTICLEREVISION     9
```

Format string examples: (where date is Sunday, December 19th, 1993 or Monday, February 3rd, 2001)
Where format string follows this:
Year: If y or yy, then last 2 digits of the year (93). If yyyy, then 4 digit year (1993).
Month: If m, then integer month, no leading zero. If mm, then 2 digit integer month. If mmm, then 3 letter abbreviated month. If mmmm, then 3-4 letter abbreviated month. If mmmmm, then full month.
Day: If d, then day, no leading zeros. If dd, then 2 digit day. If dddd, then day, no leading zero, with "st,nd,rd,th" appended.
DayOfWeek: if w, then numerical day of the week, sunday = 1. If ww, then Su,M,Tu,W,Th,F,Sa. If www, then Sun,Mon,Tue,Wed, Thu, Fri, Sat. If wwww, then Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.

y-m-d	93-12-19	or	1-2-3
mm/dd/yy	12/19/93	or	02/03/01
d/mm/yyyy	19/12/1993	or	3/02/2001

TEXT_TIME 19 time(vnum type, vnum typedatalen, typedata, vnum formatlen, formatstring, vnum languagelen, language)

Inserts the time in the text stream at the command's location in the current text attributes. Where update indicates when time should be updated. Time is the time to display in HHMMSSII format. Format is an ascii format string for formating.
Where update can be:

```
#define  TIME_CUSTOM              0
#define  TIME_CREATED             1
#define  TIME_MODIFIED            2
#define  TIME_SAVED               3
#define  TIME_LOADED              4
#define  TIME_PRINTED             5
#define  TIME_DOCUMENTREVISION    6
#define  TIME_CHAPTERREVISION     7
#define  TIME_PAGEREVISION        8
#define  TIME_ARTICLEREVISION     9
```

Format string examples: (where time is 11:29 AM or 3:25 PM)
Where format string follows this:
Hour: If h, then hour with no leading zero. If hh, then 2 digit hour.
Minute: If m, then minutes, no leading zero. If mm, then 2 digit minutes.
Second: If s, then seconds, no leading zero. If ss, then 2 digit seconds.
AM/PM: If AMPM, then AM or PM. If PM, then only shows PM when time > 12:00.

hh:mm:ss	93-12-22

TEXT_NAME 20 name (vnum type, vnum typedatalen, typedata)

Inserts the name of the item specified by type in the text stream at the command's location in the current text attributes.
Where type can be:

#define	NAME_DOCUMENT	1	
#define	NAME_CHAPTER	2	
#define	NAME_MASTERPAGE	3	
#define	NAME_PAGE	4	
#define	NAME_JOBID	5	
#define	NAME_MARK	6	data is the TEXT_MARK number
#define	NAME_INDEXMARK	7	data is the TEXT_INDEXMARK number
#define	NAME_VARIABLE	8	data is the variable name

TEXT_TYPEFAMILY 25 family (7-bit ASCII family name string)

Sets the font family. Data is the family name to use. Helvetica is a family. Helvetica-Bold is not, and is considered an invalid use of FONT. Instead, Helvetica-Bold would be set by FONT(Helvetica) and ATTRB(B) or ATTRB("Bold") for styles undefined. This and TEXT_TYPESTYLE are the only commands to use 7 bit ascii. At this time **no** unicode or native character sets are permitted for these two commands.

TEXT_TYPESTYLE 26 style attribute (7-bit ASCII style string)

Sets the font style. Data is the style. This and TEXT_TYPEFAMILY are the only commands to use 7 bit ascii. At this time **no** unicode or native character sets are permitted for these two commands.

TEXT_TYPESIZE 27 type size (vnum COORD size [, vnum EMS width])

Where (COORD)size is type size. If more data, then (EMS)width is the type width change. (this is different than TEXT_POINT)

TEXT_ROTATE 28 rotate text (svnum ANGLE slant [, svnum ANGLE twist])

This causes the glyphs to be rotated about their orgin on the baseline by (ANGLE)slant and (ANGLE)twist amounts. Oblique text and other more bizarre effects are possible.

TEXT_BOLD 29 bold attribute (vnum type)

Where type is the state as follows:

```
#define  BOLD_OFF        0
#define  BOLD_ON         1
#define  BOLD_LIGHT      2
```

TEXT_ITALIC 30 italic attribute (vnum type [, svnum ANGLE angle])

Where type is the state as follows:

```
#define  ITALIC_OFF      0
#define  ITALIC_CUSTOM   1
#define  ITALIC_STANDARD 2
#define  ITALIC_BACKSLANT 3
```

custom italic angle = angle. This really just controls the obliqueness of the text, and is not much more than a twist of the object unless a type italic font is available.

TEXT_OUTLINE 31 outline attribute (vnum type [, attribute-data])

Where type is the state as follows:

```
#define  OUTLINE_OFF     0
#define  OUTLINE_CUSTOM  1
#define  OUTLINE_STANDARD 2
```

Where attribute-data is fill style to use. This is really nothing more than a toggle between two different fill styles.

TEXT_SHADOW 32 shadow attribute (vnum type [, svnum EMS xoffset, svnum EMS yoffset, svnum ANGLE slant, svnum ANGLE twist, attribute-data])

Where type is the state as follows:

```
#define  SHADOW_OFF          0
#define  SHADOW_CUSTOM       1
#define  SHADOW_GRAY         2
```

Where xoffset is the horizontal offset (+ is to the right) in EMS, yoffset is the vertical offset (+ is down) in ems, slant is the amount to slant the text by (+ is counterclockwise like object slant) in AUNITS degrees, twist is the amount to twist the text by (+ is counterclockwise like object twist) in AUNITS degrees, and attribute-data is the fill style to use.

TEXT_REVERSE 33 reverse attribute (vnum type [, svnum EMS left, svnum EMS top, svnum EMS right, svnum EMS bottom, attribute-data])

Where type is the state as follows:

```
#define  REVERSE_OFF         0
#define  REVERSE_CUSTOM      1
#define  REVERSE_STANDARD    2
```

TEXT_UNDERLINE 34 underline attribute (vnum type [, vnum range, svnum EMS yoffset, attribute-data])

Where type is the state as follows:

```
#define  UNDERLINE_OFF       0
#define  UNDERLINE_CUSTOM    1
#define  UNDERLINE_SINGLE    2
#define  UNDERLINE_DOUBLE    3
#define  UNDERLINE_STRIKE    4
```

Should define applications idea of standard. Where yoffset is offset from the baseline for the center of the line running parallel to the baseline in EMS EUNITS units. attribute-data is the line style to use.

Where range can be:

```
#define  UNDERLINE_WORD      1
#define  UNDERLINE_CHARACTER 2
#define  UNDERLINE_CONTINUOUS 3
```

TEXT_SCRIPT 35 super/subscript attribute (vnum type, vnum EMS yoffset, vnum EMS xsize, vnum EMS ysize)

Where type is the state as follows:

```
#define SCRIPT_OFF          0
#define SCRIPT_SUPERCUSTOM  1
#define SCRIPT_SUPERSTANDARD 2
#define SCRIPT_SUBCUSTOM    3
#define SCRIPT_SUBSTANDARD  4
```

Yoffset offsets by the current size before ysize is applied.

TEXT_SMALLCAPS 36 small caps attribute (vnum type [, vnum EMS xsize, vnum EMS ysize])

Where type is the state as follows:

```
#define SMALLCAPS_OFF       0
#define SMCAPSTYLE_CUSTOM   1
#define SMCAPSTYLE_SMALL    2     don't ask me what difference is between this
#define SMCAPSTYLE_TITLING  3     and this????
```

TEXT_CASE 37 case attribute (vnum type)

Where type is the state as follows:

```
#define CASE_OFF            0
#define CASE_UPPER          1
#define CASE_LOWER          2
#define CASE_CAPITALIZED    3
```

TEXT_TRACKING 50 intercharacter tracking (svnum EMS value)

Inter-character spacing is modified by (EMS)value.

TEXT_TRACKTABLE 51 tracking table (vnum type)

Where type is:

```
#define TRACKTABLE_NONE         0
#define TRACKTABLE_MONOSPACED   1
#define TRACKTABLE_VERYLOOSE   10
#define TRACKTABLE_LOOSE       15
#define TRACKTABLE_NORMAL      20
#define TRACKTABLE_TIGHT       25
#define TRACKTABLE_VERYTIGHT   30
```

TEXT_TRACKRANGE 52 tracking range (svnum EMS MinChar, svnum EMS OptChar, svnum EMS MaxChar, vnum EMS MinWord, vnum EMS OptWord, vnum EMS MaxWord)

TEXT_ALIGNMENT 53 justification (vnum type)

Where type is the justification type as follows:

```
#define ALIGNMENT_LEFT      0
#define ALIGNMENT_CENTER    1
#define ALIGNMENT_RIGHT     2
#define ALIGNMENT_JUSTIFY   3
```

TEXT_LASTLINEFLUSH 54 last line flush zone (vnum COORD zone)

If the last line of text in a paragraph lies less than (COORD) zone from the right edge, then justify that line like any other line. Otherwise, leave if unjustified (ragged right). If zone == 0, then no last line justification will occur.

TEXT_MINLINELENGTH 55 minimum line length (vnum COORD minlen)

Minimum line length that will be considered to placing text on.

TEXT_LEADING 56 leading (vnum type, type data) *** this was done in a non-extensible way. To late?

Where type can be:

#define	LEADING_AUTOMATIC	1	svnum COORD value. where leading = value + line's max type size
#define	LEADING_FIXED	2	vnum COORD value. where leading = value
#define	LEADING_RELATIVE	3	vnum EMS value. where leading = line's max type size * value
#define	LEADING_LOCALGRID	4	vnum COORD value. where line is placed on an even multiple of value
#define	LEADING_GLOBALGRID	5	no data. line is place on the global grid offset/spacing.

TEXT_MAINTAINLEAD 57 maintain leading (vnum flag)

Where flag can be:

#define	MAINTAINLEAD_OFF	0
#define	MAINTAINLEAD_ON	1

TEXT_BASELINE 58 baseline offset (svnum COORD offset)

This is the old super-subscript command. Just a basic baseline shift.

TEXT_BASELINELEAD 59 baseline position (vnum type, type data) *** this was done in a non-extensible way. To late?

Where type can be:

#define	BASELINELEAD_FIXED	1	vnum COORD value.
#define	BASELINELEAD_RELATIVE	2	vnum EMS value.
#define	BASELINELEAD_TFIXED	3	vnum COORD value.
#define	BASELINELEAD_TRELATIVE	4	vnum EMS value.

TEXT_PARALEAD 60 paragraph leading (vnum beforetype, type data (, vnum aftertype, type data)

Where beforetype & aftertype can be:

#define	PARALEAD_NONE	0	no data.
#define	PARALEAD_FIXED	1	vnum COORD value. where para spacing = value
#define	PARALEAD_RELATIVE	2	vnum EMS value. where para spacing = line leading * value
#define	PARALEAD_LOCALGRID	3	vnum COORD value. where para is placed on an multiple of value
#define	PARALEAD_GLOBALGRID	4	no data

TEXT_CCB 61 conditional column break (vnum flag)

Where flag is:

#define	CCB_OFF	0
#define	CCB_ON	1

If a range sufficiently large is marked that can not be held inside any subsequent containers, then the text is overflow.

TEXT_CPB 62 conditional page break (vnum flag)

Where flag is:

```
#define  CPB_OFF                    0
#define  CPB_ON                     1
```

The text may exist across several columns provided it stays on the same page. If a range sufficiently large is marked that can not be held on any subsequent pages, then the text is overflow.

TEXT_WIDOWORPHAN 63 widow orphan control (vnum type [, vnum start [, vnum end]])

Where type is:

```
#define  WIDOWORPHAN_NONE               0
#define  WIDOWORPHAN_KEEPPARAGRAPH      1       keep each paragraph together
#define  WIDOWORPHAN_KEEPLINES          2
```

start is the number of lines at the start of the paragraph to keep together, and end is the number of lines at the end of the paragraph to keep together. If end is ommited then end == start.

TEXT_INDENT 64 left/right margin indent (vnum COORD left indent, vnum COORD right indent)

Left is the left offset measure from the left side of the container in COORD units. Right is the right offset measure from the right side of the container (local or global edge?) in COORD units.

TEXT_FIRSTLINE 65 first paragraph line indent/outdent (svnum COORD value)

If value > 0, then indent (indent first line of paragraph by value amount).
If value < 0, then outdent (indent all first line of paragraph by value amount).
If value == 0 means no indent or outdent. Value is in COORD units.

TEXT_INDENTMARK 66 indent paragraph here mark

The left offset where in the text this command falls is where the subsequent lines will left indent in this paragraph.

TEXT_DROPCAP 67 drop cap (vnum characters [, vnum lines, vnum datalen, attribute-data])

Drop Cap at start of each paragraph. Where characters is the number of characters to use from the start of each paragraph. If characters == 0, then no drop cap (and lines & attribute-data may be omitted). Else, lines is the number of lines tall to make the text (as measured in the current lines leading amount). If attribute-data is omitted, than the current font & width ratio is used, otherwise, the typeface & size is defined within using standard TEXT_FONT, TEXT_ATTR, TEXT_TYPESIZE, and possibly even TEXT_TAG. If the size is defined, only the width scale is used.

TEXT_BULLET 68 bulleted paragraph (vnum stringlen, UNICODE string [,vnum datalen, attribute-data])

Bulleted paragraph define. Where character is the glyph value to place before each paragraph. If stringlen ==0, then no bulleted paragraphs. If attribute-data is omitted, than the current font & size is used, otherwise, the typeface & size is defined within using standard TEXT_FONT, TEXT_ATTR, TEXT_TYPESIZE, and possibly even TEXT_TAG.

TEXT_KERN 70 manual kern (svnum EMS kernamt)

Where kernamt is the amount in 1/10000ems to kern the two character together on either side of this command. This command was manually entered by the user at this location.

TEXT_BKERN 71 batch inserted kern (svnum EMS kernamt)

Where kernamt is the amount in 1/10000ems to kern the two character together on either side of this command. This command was inserted by a program at the users request in a batch mode over a range of text.

TEXT_AKERN 72 auto kern (vnum flag, [vnum type, vnum COORD kernabove])

Where flag is:

#define	AKERN_OFF	0
#define	AKERN_ON	1

Where type is:

#define	AKERN_TEXT	1	use text kerning (a looser set of kern values)
#define	AKERN_DESIGN	100	use design kerning (a much tighter set of kern values)

Kernabove (COORD) is the size of the type at which kerning begins. Any type larger than this will be kerned if AKERN_ON.

TEXT_OPTICALALIGN 73 optical alignment (vnum flag)

Where flag is:

#define	OPTICAL_OFF	0
#define	OPTICAL_ON	1

Which defines the start/stop where characters with soft edges will be optically aligned with the edges, ie overhung.

TEXT_HANGINGPUNC 74 hanging punctuation (vnum flag)

Where flag is:

#define	HANGING_OFF	0
#define	HANGING_ON	1

Which defines the start/stop of a marked range of text which will have hanging punctuation.

TEXT_ALIGATURE 75 automatic ligature control (vnum flag [, vnum type, vnum EMS mintrack, vnum EMS maxtrack])

Where flag is:

#define	ALIGATURE_OFF	0
#define	ALIGATURE_ON	1

Where type is:

#define	ALIGATURE_ANY	1
#define	ALIGATURE_NOFFIFFL	2

TEXT_HYPHEN 76 manual soft hyphen ([vnum priority])

Where optional priority is the priority value used to distinguise between good and bad points to break. The lowest numbered hyphen or softbreak within the range that can be reached via min/max tracking will be used. Hyphens without priority are assumed to be first to pick (priority of 0). This command was manually entered by the user at this location

TEXT_BHYPHEN 77 batch inserted soft kern ([vnum priority])

Where optional priority is the priority value used to distinguise between good and bad points to break. The lowest numbered hyphen or softbreak within the range that can be reached via min/max tracking will be used. Hyphens without priority are assumed to be first to pick (priority of 0). Batch hyphens are ignored when auto-hyphenation is turned on. This command was inserted by a program at the users request in a batch mode over a range of text.

TEXT_AHYPHEN 78 auto hyphen (vnum flag, [vnum COORD zone, vnum inarow])

Where flag is:

```
#define  AHYPHEN_OFF          0
#define  AHYPHEN_ON           1
```

Where zone (COORD) is the range from the right edge to begin considering hyphenation break points. No points outside of this zone are possible. Inarow is the most number of hyphens to be seen in a row.

TEXT_AHYPHENLANGUAGE 79 auto hyphenation language values (vnum languagelen, language, vnum datalen, data)

American driver data is : vnum smallest, vnum minbefore, vnum minafter, vnum lowercase

Where smallest is the the smallest word to be hyphenated. Minbefore is the fewest number of characters to chop off the front, minafter is the fewest to chop off the back.

Where lowercase is:

```
#define  AHYPHEN_LOWERONLY    1
#define  AHYPHEN_UPPERLOWER   2
```

TEXT_SOFTBREAK 80 soft break (vnum priority , vnum stringlen, string)

Where optional priority is the priority value used to distinguise between good and bad points to break. The lowest numbered hyphen or softbreak within the range that can be reached via min/max tracking will be used. If a line is broken at a softbreak, then (UNICODE)character is displayed at the end (room must be left for it on the line to be broken there). len == 0 is a softline break. Softbreaks without priority are assumed to be last to pick.

TEXT_NONBREAKING 81 non-breaking (vnum stringlen , string)

Text data is data to be processed as normal text (ie, formated and displayed), but is not to be considered for breakpoints. This is usually used for non-breaking spaces, hyphens, etc.

TEXT_INDEXMARK 82 index mark point (vnum number, vnum namelen, name)

Number is index mark number. This is used for index generation. Data to be defined...

TEXT_MARK 83 mark point (vnum number, vnum namelen, name)

Number is mark number. This is used for linking objects to text. Actual placement flags are defined on the object end (and each mark point could be used by several items). Other uses are possible

TEXT_RANGE 84 range point (vnum flag, vnum number)

Where flag is:

```
#define  RANGE_OFF          0
#define  RANGE_ON           1
```

Number is range number. Unknown uses at this time. Each range could be used by several items.

TEXT_FOOTNOTE 85 footnote (????)

Footnote should specify the location (end of paragraph, end of column, bottom of page, end of chapter, end of document). Text of the footnote. The format of the number and the number. Need somekind of global data specifing the numbering method?

TEXT_TABRULER 86 tab ruler (tab data)

List of the type & locations of the tab stops. If a ruler also include margins, that would be defined with the FIRSTLINE and INDENT commands.

```
vnum count,
        vnum type, vnum COORD offset,  vnum datalen, data

#define TAB_TYPELEFT                0x01    <tabcmd data>
#define TAB_TYPECENTER              0x02    <tabcmd data>
#define TAB_TYPERIGHT               0x03    <tabcmd data>
#define TAB_TYPEUSDECIMAL           0x04    <tabcmd data>
#define TAB_TYPEEURODECIMAL         0x05    <tabcmd data>
#define TAB_TYPECUSTOMDECIMAL       0x06    vnum UNICODE align glyph <tabcmd data>
```

Where count is the number of tab stops defined.Type is the type of alignment to use. Leader glyph (VNUM) is the unicode? character to use as a leader (zero means no leader). Align Glyph (VNUM) is the unicode? character to align with the point. What about left/center/right alignment of the decimal? Offset (COORD) is the offset from the left edge of the container. What about irregular containers? Should it be from the left most point or the local left most point?

Tab command data is a series of (vnum command, vnum commandlen, commanddata) where commands are:

```
#define  TAB_LEADER             0x01        series of characters to fill with
#define  TAB_LEADERSPACING      0x02        vnum EMS character width
```

TEXT_DIRECTION 88 direction (vnum direction)

Where direction is:

```
#define  DIRECTION_LEFT         1       left to right
#define  DIRECTION_RIGHT        2       right to left
#define  DIRECTION_TOP          3       top to bottom
#define  DIRECTION_BOTTOM       4       bottom to top
```

TEXT_LANGUAGE 89 language (language-driver)

TEXT_BREAKING 90 breaking text (vnum priority, vnum strlen, string)

The text can be broken after each character in string. Same as placing a soft new line between each character, except they stay together!

Some things left to sort out and define:

Continued From/On
Hypertext Link
Min/Max/Opt Char Width(?)- at one time I knew what this meant...
Book Marks

FILL COMMANDS

FILL 100 fill (attribute-data)

Used to encapsulate attribute data that pertains to the fill. For example, one may wish to use the same color tag for a fill and stroke, and this way the tag command would be wrapped with the FILL command to indicate the color is for the fill.

STROKE 101 stroke (vnum #lines, {vnum data-length, attribute-data ...})

Used to encapsulate attribute data that pertains to the stroke. For example, one may wish to use the same color tag for a fill and stroke, and this way the tag command would be wrapped with the STROKE command to indicate the color is for the stroke fill. FILL_STYLE and other commands are still found inside STROKE, since a line has a fill style plus stroke data. Data between the { } is to be repeated once for every #lines.

COLOR 102 color (vnum #colors,{vnum data-length, attribute-data ...})

Used to encapsulate attribute data that pertains to fill color. If defining the color for a basic (solid) fill, the #colors would be 1, and would be followed by 1 set of attribute-data. For a 2 color gradient, #colors would be = 2, followed by 2 sets of attribute-data.

FILLED 126 color (vnum flag)

#define	FILL_OFF	0x00	turn fill on
#define	FILL_ON	0x01	turn fill off

STROKED 127 color (vnum flag)

#define	STROKE_OFF	0x00	turn stroke on
#define	STROKE_ON	0x01	turn stroke off

FILL_STYLE 103 fill style (vnum type, type-data)

#define FS_TYPE_SOLID	0x01	no data	
#define FS_TYPE_GRAD	0x02	vnum ANGLE angle, grad-data *see below for details *	
#define FS_TYPE_RADIAL	0x03	radial-data * see below for details *	
#define FS_TYPE_SHAPE	0x04	shape-data * see below for details *	
#define FS_TYPE_COLORPTRN	0x05	vnum w, vnum h, vnum depth, data	
#define FS_TYPE_FRACTAL?	0x06	?????	
#define FS_TYPE_COMPLEX?	0x07	?????	

FS_TYPE_GRAD: Gradient fill. Default start and end is full range with no over/underlap of item filled. angle (ANGLE) is angle of gradient. grad-data is a series of grad-command, datalength, data until type-data is gone.
grad-commands are:

#define GRAD_RAMP	0x01	vnum type, type data	see RAMP type
#define GRAD_POINTS	0x02	vnum type, type data	see GRAPOINTS type

GRADPOINTS type:

#define GRADPOINTS_PATH	0x00	no data	just edge to edge of path
#define GRADPOINTS_ABS	0x01	vnum COORD's sx, sy,ex,ey	absolute page coordinates
#define GRADPOINTS_REL	0x02	vnum COORD's sx, sy,ex,ey	relative to rotation point

FS_TYPE_RADIAL: Radial fill. Default center is bbox center of item filled. radial-data is a series of radial-command, datalength, data until type-data is gone.
radial-commands are:

#define RADIAL_RAMP	0x01	vnum type, type data	see RAMP type
#define RADIAL_CENTER	0x02	vnum type, type data	see RADIALCENTER type
#define RADIAL_RADIUS	0x03	vnum type, type data	see RADIALRADIUS type

RADIALCENTER type:

#define RADIALCENTER_PATH	0x00		just in the center of path
#define RADIALCENTER_ABS	0x01	vnum COORD x, vnum COORD y	absolute page coordinates
#define RADIALCENTER_REL	0x02	vnum COORD x, vnum COORD y	relative to rotation point
#define RADIALCENTER_RPATH	0x03	vnum EMS xscl, vnum EMS yscl	relative center of path. 0,0 is center

RADIALRADIUS type

#define RADIALRADIUS_PATH	0x00	no data	radius from center to path edge
#define RADIALRADIUS_ABS	0x01	vnum COORD r	absolute radius

FS_TYPE_SHAPE:

#define SHAPE_RAMP	0x01	vnum type, type data	see RAMP type
#define SHAPE_INSET	0x02	vnum type, type data	see SHAPEINSET type

SHAPEINSET type

#define SHAPEINSET_ABS	0x01	vnum COORD inset	absolute inset

FS_TYPE_COLORPTRN:

RAMP type:

#define RAMP_SPECIAL	0x01	name
#define RAMP_LINEAR	0x02	
#define RAMP_INVLINEAR	0x03	
#define RAMP_LOG	0x04	
#define RAMP_INVLOG	0x05	
#define RAMP_SIN	0x06	
#define RAMP_INVSIN	0x07	
#define RAMP_SAW	0x08	
#define RAMP_INVSAW	0x09	

FILL_PATTERN 104 (vnum type, type-data)

#define	FP_TYPE_SOLID	0x01	
#define	FP_TYPE_PATTERN	0x02	(vnum w, vnum h, bitmap)

W is the pixels wide in the pattern. H is the pixels high in the pattern. Row width is w+7 / 8 bytes. Bitmap is the data and is H * ((W + 7) / 8) in len. Data is right justified such that the first byte for each row encountered is the high order bits of the row and contains the unused bits (if w not 8,16,24...).

STROKE_STYLE 105 (vnum type, type-data)

#define	SS_TYPE_SOLID	0x01	no data
#define	SS_TYPE_PATTERN	0x02	vnum count, data
#define	SS_TYPE_DASH	0x03	vnum offset, vnum count, vnum vnumarray[count]
#define	SS_TYPE_BORDER	0x04	?????

LS_TYPE_PATTERN: count is the pixels used from the data. data length is (count+7) / 8. Each pixel is n dash units long. Pixels are right justified in the data such that the first byte encountered is the high order bits and contains the unused bits (if count not 8, 16, 24,...).

LS_TYPE_DASH: offset is the offset into the dash pattern to begin from. count is the number of vnums that follow that comprise the dash pattern. This is the same as the postscript setdash. count of 0 is a solid line (but it's prefered that SS_TYPE_SOLID be used).

LS_TYPE_BORDER:

STROKE_WEIGHT 106 (vnum type, type-data)

#define	SW_TYPE_ABS	0x01	vnum COORD width
#define	SW_TYPE_REL	0x02	vnum EMS width (relative to what?)

STROKE_OFFSET 113 (vnum type, type-data)

#define	SO_TYPE_ABS	0x01	vnum COORD offset
#define	SO_TYPE_REL	0x02	vnum EMS roffset (relative to what?)

STROKE_JOIN 107 (vnum type, type-data)

#define	SJ_TYPE_MITER	0x01	vnum ANGLE miter limit
#define	SJ_TYPE_ROUND	0x02	no data
#define	SJ_TYPE_BEVEL	0x03	no data

SJ_TYPE_MITER: miter limit (ANGLE) is the smallest angle between two lines that will show a miter join. Otherwise, shows as a join of type SJ_TYPE_BEVEL.

STROKE_LINESTART 108 (vnum type, type-data)
STROKE_LINEEND 109 (vnum type, type-data)

#define	SL_TYPE_SPECIAL	0x01	arrow function name
#define	SL_TYPE_FLAT	0x02	no data
#define	SL_TYPE_ROUND	0x03	no data
#define	SL_TYPE_ARROW	0x04	no data

STROKE_CAPEND 110 (vnum type, type data)

#define	SC_TYPE_BUTT	0x01	no data
#define	SC_TYPE_ROUND	0x02	no data
#define	SC_TYPE_SQUARE	0x03	no data

COLOR_STYLE 111 (vnum flags, vnum EMS c, vnum EMS m, vnum EMS y, vnum EMS k, vnum EMS tint, vnum type, type-data)

#define	CS_FLAG_OVERPRINT	0x01	
#define	CS_FLAG_SPOT	0x02	
#define	CS_TYPE_BLACK	0x01	no data
#define	CS_TYPE_WHITE	0x02	no data
#define	CS_TYPE_GREY	0x03	vnum %K
#define	CS_TYPE_RGB	0x04	vnum %R, vnum %G, vnum %B
#define	CS_TYPE_HLS	0x05	vnum °H, vnum %L, vnum %S
#define	CS_TYPE_HSV	0x06	vnum °H, vnum %S, vnum %V
#define	CS_TYPE_CMYK	0x07	vnum %C, vnum %M, vnum %Y, vnum %K
#define	CS_TYPE_LIBRARY	0x08	vnum libnamelen, libname, vnum namelen, name
#define	CS_TYPE_TRUMATCH	0x09	
#define	CS_TYPE_FOCALTONE	0x0a	
#define	CS_TYPE_CIE	0x0b	
#define	CS_TYPE_YCC	0x0c	
#define	CS_TYPE_YIQ	0x0d	

COLOR_TINT 112 color tint (vnum EMS tint)

Percentage tint to be applied to the “active” color.

SCREEN 114 screen (vnum count, { vnum data-length, attribute-data})

SCREEN_FREQUENCY 115 screen frequency (vnum freq)
if (cmdlen == 0) then default.

SCREEN_ANGLE 116 screen angle (svnum angle)
if (cmdlen == 0) then default.

SCREEN_SPOT 117 screen spot (vnum type, type-data)

#define	SS_TYPE_DEFAULT	0x00	no data
#define	SS_TYPE_SPECIAL	0x01	spot function name
#define	SS_TYPE_DOT	0x02	no data
#define	SS_TYPE_LINE	0x03	no data

if (cmdlen == 0) then default.

TRAPPING_ATTR? ?

\/choke, auto/manual, amount (.0001 inc, .003-.006 good def)
trap type (no trap,no knockout, trap w/ common, w/uncommon, with K)

GAMMA ?
UCRGCR ? max ink coverage (300% 250-260% for web presses)
dotgain function??