Version 16.0.3.0
December 2020
702P08485

Xerox® VIPP® Language Reference Manual

xerox™

Table of Contents

1

VIPP® Language Overview

This chapter contains:

This manual documents the Variable Information Production PrintWare (VIPP®) programming language. VIPP® is a programming language dedicated to the design of variable information (VI) applications. The VIPP® language is used to describe the structure and logic of the variable data to process, and to produce the appearance of the pages. The VIPP® language is a variable document composition language that provides nearly unlimited capability and flexibility to users.

The *VIPP® Language Reference Manual* contains information about VIPP® commands, markers, transform functions, variables, parameters, error messages, programming tips, and answers to many frequently asked questions about the VIPP® language. For background information and descriptions of VIPP® resources, files, and utilities, and for information about FreeFlow VI Compose, refer to the *FreeFlow® VI Compose User Guide*.

The description of each VIPP® command includes this information:

- Command syntax
- Applicable modes for use
- Related commands

Information about the VIPP® language and components is grouped into these topics:

- VIPP® Commands
- Markers
- Transform functions
- Variables
- Parameters
- Composite constructs

VI Suite Customer Forum

Xerox hosts a Community Support Forum. The VI Suite Customer forum is now part of this larger support forum, allowing you to post and review information about Xerox products and services all from one location. Take a minute to log into the customer forum community: https://VIPPsupport.xerox.com.

Font Download

To download specialty imaging and barcode fonts, go to www.xerox.com/support, then search for **VIPP**. Select **Software**.

Some variable information programs, specialty imaging fonts, and barcode fonts are available for purchase by customers in the United States, using a credit card. The downloadable products are at the Xerox eStore https://buy.xerox.com.

Note: When you download fonts, you are directed to review an End User License Agreement. To download the fonts, review and accept the End User License Agreement. If you do not accept the End User License Agreement, the program exits the font download page.

Using PDF Resources with VIPP®

VI Compose can support PDF resources in the legacy PostScript RIP, the Adobe PDF Print Engine (APPE), and the various PDF output capabilities of the FreeFlow VI Suite. For more information, refer to *Using PDF resources with VIPP®, APPE, PDF/VT, and Embed EPS options* in the *FreeFlow VI Compose User Guide*.

FreeFlow® VI Compose Open Edition

The FreeFlow VI Compose Open Edition or VIC(OE) software is a modified version of the VI Compose software that is supported on Xerox print engines. VIC(OE) has been modified to install on non-Xerox devices and to support license activation through normal Xerox channels.

Due to internal operational differences between Xerox and non-Xerox print devices, some limitations apply. These limitations exist mainly around the areas of feeding and finishing, but can exist in other areas. For example, in jobs where VIPP® attempts to write file position information to the device. Xerox has no control over the third-party devices. Operations that are normal for Xerox devices may not be allowed on third-party devices. Because of these possible limitations, it is recommended that all jobs run to a third-party device are validated fully before running jobs in production.

Xerox has no engineering support for non-Xerox production devices. Any issue reported are validated against a similar Xerox production device. If the Xerox production device exhibits the same issue, Xerox can fix the issue in a software patch and provide the fix to the customer to install and verify. If this does not fix the issue on the non-Xerox production device, Xerox may be unable to assist further.

For more information, refer to *FreeFlow® VI Compose (Open Edition) Installation and Overview*.

VIPP® Command List

This table contains an alphabetical list of all VIPP® commands, markers, transform functions, and variables. In addition to each command type, the information includes the command function. When using this document in Help or PDF format, you can click any of the command names in the list to access the command description.

Command functions are grouped into the categories described here. The descriptions of the categories provide only general examples, and they are not inclusive.

- Cyclecopy commands control the number of copies and how those copies are handled by the printer.
- Date and time functions allow you to set date and time information.
- Design and debug commands are used to design VIPP® jobs and to aid in debugging the applications that you create.
- Font and color commands control the type and appearance of fonts and text backgrounds in VIPP® jobs.
- Output device control commands control options at the output device. These commands allow you to set stapling, jog, offset, and duplex options, media requirements, output resolution, and so on.
- Page control commands control page breaks, page imposition, skipping and printing pages, inserting slip sheets, and so on.
- Page layout commands determine the appearance of the job. Page layout commands include commands to set column and margin width, define forms used, set page orientation, and so on.
- Page marking commands allow you to insert images, segments, text, glyphs, and so on. Page marking commands allow you to set DDG and format parameters, and draw charts.
- PCC processing commands allow you to set printer control character options.
- PDF control commands are a set of VIPP® commands that allow you to create interactive elements when the VIPP® job is rendered into a PDF document.
- Print file processing commands determine how the files are processed, and set Native, Line, Database, and XML modes, and so on.
- Resource control commands allow you to use cache and pre-cache options, set paths and patterns for jobs and VI Projects, and store data in memory as a virtual file.
- RPE processing allows you to use Record Processing Entry data processing options.
- Transform control commands allow you to change text within the document, strip spaces, create barcodes, and so on.

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
%	M	
% %	M	
%!	M	
% %BoundingBox	M	

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
% %EOD XGF	M	
% %EOF	M	
% %PagesPerBooklet	M	
% %XGF	M	
$$name.	M	
++/–	C	Miscellaneous
[=name=]	M	
2OF5	T	Transform Control
64TO256	T	Transform Control
ABSPOS	C	Page Marking
ACCLOG	C	Miscellaneous
ADD	C	Miscellaneous
ADVPAGE	C	Page Control
AUTOGRID	V	Design and Debug
AZTEC	C	Page Marking
ACK	V	Output Device Control
ACKSP off	C	Print File Processing
ACKSPF on	C	Print File Processing
ACKSPP on	C	Print File Processing
BCALL	C	Page Marking
BBOX	M	
BCOUNT	V	Miscellaneous
BEGINARBM	C	Print File Processing
BEGINARBT	C	Print File Processing
BEGINFRAME	C	Miscellaneous
BEGINIMP	C	Page Control
BEGINPAGE	C	Miscellaneous
BEGINPCC	C	PCC Processing
BEGINRPE	C	RPE Processing
BEGINTABLE	C	Page Marking

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
BEGINXPD	C	Print File Processing
BIDI	T	Print File Processing
BLGRID	V	Design and Debug
BOOKLETRANGE	C	Print File Processing
BOOKMARK	C	PDF Control
BPCOUNT	V	Miscellaneous
BSTRIP	T	Transform Control
BSTRIP off	C	Print File Processing
BTA	C	Print File Processing
BTRIM	T	Transform Control
BTS	C	Print File Processing
CACHE	C	Resource Control
CASE	C	Miscellaneous
CASELOW	T	Transform Control
CASETI	T	Transform Control
CASEUP	T	Transform Control
CHKPOINT	C	Cyclecopy
CLIP	V	Page Marking
CODE39	T	Transform Control
CODE128 and EAN128	T	Transform Control
COLLATE dbm	C	Cyclecopy
COLLATE off	C	Cyclecopy
COLLATE on	C	Cyclecopy
COLW	V	Page Layout
COPYRANGE	C	Cyclecopy
CPCOUNT	V	Miscellaneous
CS	T	Transform Control
CURLINE	V	RPE Processing
CUTMARK	C	Page Marking
D DD	V	Date and Time
D DOY	V	Date and Time

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
D DWL	V	Date and Time
D DWS	V	Date and Time
D MO	V	Date and Time
D MOL	V	Date and Time
D MOS	V	Date and Time
D YY	V	Date and Time
D YYYY	V	Date and Time
DATAMATRIX	C	Page Marking
DAYS	T	Date and Time
DEVRES	V	Output Device Control
DIV	C	Miscellaneous
DJDEBEGIN	C	Print File Processing
DJDECMD	V	Print File Processing
DJDEPAR	V	Print File Processing
DRAWB and DRAWBR	C	Page Marking
DRAWBAR	C	Page Marking
DRAWBC	C	Page Marking
DRAWBM and DRAWBRM	C	Page Marking
DRAWC	C	Page Marking
DRAWCRV	C	Page Marking
DRAWPAR	C	Page Marking
DRAWPATH and DRAWPATHR	C	Page Marking
DRAWPFF	C	Page Marking
DRAWPIE	C	Page Marking
DRAWPOL	C	Page Marking
DRAWRDR	C	Page Marking
DUPLEX_off	C	Output Device Control
DUPLEX_on	C	Output Device Control
EAN13/EAN8	T	Transform Control
ENDARBM	C	Print File Processing

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
ENDARBT	C	Print File Processing
ENDBOOKLET	C	Print File Processing
ENDCASE	C	Miscellaneous
ENDCLIP	C	Page Marking
ENDIFALL	C	RPE Processing
ENDIMP	C	Page Control
ENDJOB	C	Print File Processing
ENDOFRUN	C	Output Device Control
ENDOFSET	C	Output Device Control
ENDPAGE	C	Miscellaneous
ENDPCC	C	PCC Processing
ENDRPE	C	RPE Processing
ENDTABLE	C	Page Marking
ENDXPD	C	Print File Processing
ETA	C	Print File Processing
ETCLIP	C	Page Marking
ETS	C	Print File Processing
EXIST	C	Resource Control
EXIT	C	Print File Processing
EXPAND	M	
EXTVAR	M	
F2S	T	Transform Control
FBIND	C	Resource Control
FCALL	C	Page Marking
FILLOMR	C	Page Marking
FOREACH	C	Miscellaneous
FORMAT	T	Transform Control
FORMSHIFT	C	Page Layout
FRCOUNT	V	Print File Processing
FRLEFT	V	Print File Processing

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
FROMLINE	C	RPE Processing
FSHOW	C	Design and Debug
GETDATE	C	Date and Time
GETFIELD	C	Miscellaneous
GETINTV	T	Transform Control
GETITEM	C	Miscellaneous
GLT	V	Font and Color
GOTOFRAME	C	Page Control
GRIDSKIP	V	Page Control
HCOLOR	V	Font and Color
HDISP	V	Page Marking
HMS	T	Transform Control
HPOS	V	Page Layout
HPOS2	V	Page Layout
ICALL	C	Page Marking
IF/ELSE/ELIF/ENDIF	C	Miscellaneous
IGNOREBT off	C	Miscellaneous
IGNOREBT on	C	Miscellaneous
IHEIGHT	V	Page Layout
ILAND	C	Page Layout
INDEXALIGN	C	Font and Color
INDEXBAT	C	Font and Color
INDEXCOLOR	C	Font and Color
INDEXFONT	C	Font and Color
INDEXKERN	C	Font and Color
INDEXLSP	C	Font and Color
INDEXOTL	C	Font and Color
INDEXPIF	C	Print File Processing
INDEXRPE	C	RPE Processing
INDEXSST	C	Font and Color
IPORT	C	Page Layout

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
IREVERSE off	C	Page Marking
IREVERSE on	C	Page Marking
IWIDTH	V	Page Layout
JOG_on and JOG_off	C	Output Device Control
LAND	C	Page Layout
LMSKIP	C	Print File Processing
LNCOUNT	V	Miscellaneous
LPCOUNT	V	Miscellaneous
LPINDEX	V	Miscellaneous
LSP	V	Page Layout
MAKEVMFILE	C	Resource Control
MAKEVMFORM	C	Resource Control
MAXICODE	C	Page Marking
MOVEH	C	Page Marking
MOVEHR	C	Page Marking
MOVETO	C	Page Marking
MPR	V	Font and Color
MSPP on	C	Miscellaneous
MUL	C	Miscellaneous
NEWBACK	C	Page Control
NEWFRAME	C	Page Control
NEWFRONT	C	Page Control
NEWGROUP	C	RPE Processing
NEWPOS	C	RPE Processing
NEWSIDE	C	Page Control
NEWSTACK	C	Page Control
NL	C	Page Marking
NMP off	C	Design and Debug
NOHYPHEN	T	Transform Control
OMRINIT	C	Page Marking

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
OMRSHOW	C	Page Marking
ONEUP	C	Page Layout
ORIBL	C	Page Layout
ORITL	C	Page Layout
OTCLIP and ITCLIP	V	Page Marking
OVERPRINT on	C	Print File Processing
PAGEBRK	C	Page Control
PAGEH	V	Page Layout
PAGERANGE	C	Print File Processing
PAGEW	V	Page Layout
PDF417	C	Page Marking
PDFBOUND	C	PDF Control
PDFDEST	C	PDF Control
PDFDEVICE	V	PDF Control
PDFFORMOCG	C	PDF Control
PDFINFO	C	PDF Control
PDFOCG	C	PDF Control
PDFOPEN	C	PDF Control
PDFPAGES	V	PDF Control
PLINES	V	Page Layout
PORT	C	Page Layout
POSTJPN	T	Transform Control
POSTNET	T	Transform Control
PPCOUNT	V	Print File Processing
PRECACHE	C	Resource Control
PREV and NEXT	V	Miscellaneous
PROCESSDJDE	C	Print File Processing
PRODUCT	V	Miscellaneous
PSIZE	V	Page Layout
QRCODE	C	Page Marking

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
QSTRIP	T	Transform Control
QSTRIP on	C	Print File Processing
RELVAR	C	Miscellaneous
REPEAT	C	Cyclecopy
REPLACE	T	Transform Control
RESET	C	Miscellaneous
ROUND	T	Miscellaneous
RPCOUNT	V	Cyclecopy
RPEDEF	C	RPE Processing
RPEKEY	C	RPE Processing
RPEPOS	V	RPE Processing
RPLEFT	V	Cyclecopy
RSAVE	C	Miscellaneous
RUN	C	Miscellaneous
RUNDD	C	Miscellaneous
RUNPDF	C	Miscellaneous
RUNTIF	C	Miscellaneous
VIPP® Commands	C	Page Marking
SCALL	C	Page Marking
SETBAT	C	Font and Color
SETBFORM	C	Page Layout
SETBIDI	C	Print File Processing
SETBUFSIZE	C	Print File Processing
SETCJKENCMAP	C	Font and Color
SETCJKRULES	C	Font and Color
SETCOL	C	Font and Color
SETCOLWIDTH	C	Page Layout
SETCYCLECOPY	C	Cyclecopy
SETDATE	C	Date and Time
SETDBSEP	C	Print File Processing
SETDLFILE	C	Print File Processing

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
SETENCODING	C	Font and Color
SETEPATH	C	Resource Control
SETFINISHING	C	Output Device Control
SETFONT	C	Font and Color
SETFORM	C	Page Layout
SETFPATH	C	Resource Control
SETFRAME	C	Page Layout
SETFTSP	C	Font and Color
SETFTSW	C	Font and Color
SETGEP	C	Font and Color
SETGRID	C	Page Layout
SETGUNIT	C	Page Layout
SETINDENT	C	Page Marking
SETIPATH	C	Resource Control
SETJDT	C	Print File Processing
SETJPATH	C	Resource Control
SETKERN	C	Font and Color
SETLAYOUT	C	Page Marking
SETLFI	C	Page Layout
SETLKF	C	Page Layout
SETLMFILE	C	Print File Processing
SETLSP	C	Page Layout
SETMARGIN	C	Page Layout
SETMAXBFORM	C	Page Layout
SETMAXCOPY	C	Cyclecopy
SETMAXFORM	C	Page Layout
SETMEDIA	C	Output Device Control
SETMEDIAT	C	Output Device Control
SETMPATH	C	Resource Control
SETMULTIUP	C	Page Layout

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
SETNMP	C	Print File Processing
SETOBIN	C	Output Device Control
SETOBINT	C	Output Device Control
SETOTL	C	Font and Color
SETPAGEDEF	C	Resource Control
SETPAGENUMBER	C	Page Layout
SETPAGESIZE	C	Page Layout
SETPARAMS	C	Print File Processing
SETPAT	C	Font and Color
SETPBRK	C	Page Control
SETPCC	C	PCC Processing
SETPCD	C	Miscellaneous
SETPIF	C	PDF Control
SETPPAT	C	Font and Color
SETPPATH	C	Resource Control
SETPROJECT	C	Resource Control
SETRCD	C	RPE Processing
SETRES	C	Output Device Control
SETRPE	C	RPE Processing
SETRPEPREFIX	C	RPE Processing
SETSKIP	C	PCC Processing
SETTAB	C	Page Layout
SETTABS	C	Page Layout
SETTPAT	C	Font and Color
SETTRAN	C	Font and Color
SETTXB	C	Font and Color
SETTXC	C	Font and Color
SETTXS	C	Font and Color
SETUNIT	C	Page Layout
SETUTAB	C	Page Layout

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
SETV2HCONV	C	Font and Color
SETV2HTABLE	C	Font and Color
SETVAR	C	Miscellaneous
SETVFU	C	PCC Processing
SETZEBRA	C	Page Layout
SHC and SHc	C	Page Marking
SHEETH	V	Page Layout
SHEETW	V	Page Layout
SHIFT	C	Page Layout
SHIFTDATE	C	Date and Time
SHJ and SHj	C	Page Marking
SHL and SH	C	Page Marking
SHMF, SHMf, and SHmf	C	Page Marking
SHP and SHp	C	Page Marking
SHPATH	C	Page Marking
SHPIT	C	Page Marking
SHPOS	V	Page Marking
SHR and SHr	C	Page Marking
SHROW	C	Page Marking
SHT and SHt	C	Page Marking
SHX	C	Page Marking
SKIPPAGE	C	Page Control
SLENGTH	V	Page Layout
SLIPSHEET	C	Page Control
SOF off	C	Print File Processing
SORT	C	Miscellaneous
SPOOLNAME	C	Cyclecopy
SSIZE	V	Page Layout
STARTBOOKLET	C	Print File Processing
STARTDBM	C	Print File Processing

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
STARTLM	C	Print File Processing
STARTOFSET	C	Output Device Control
STARTXML	C	Print File Processing
STOREVAR	C	Resource Control
SUB	C	Miscellaneous
SUBSTFONT	T	Resource Control
SVPOS	V	Page Marking
T AMPM	V	Date and Time
T HH	V	Date and Time
T HH2	V	Date and Time
T MM	V	Date and Time
T SS	V	Date and Time
T TZN	V	Date and Time
TIFORI off	C	Page Layout
TIFORI on	C	Page Layout
TLENGTH	V	Design and Debug
TLGRID	V	Design and Debug
TPATH	V	Page Marking
TRIO	T	Print File Processing
TUMBLEDUPLEX_off	C	Output Device Control
TUMBLEDUPLEX_on	C	Output Device Control
TWOUP	C	Page Layout
UPCA	T	Transform Control
UPDATE	C	Miscellaneous
USPS4CB	C	Page Marking
UTF8TOLOC	T	Transform Control
UV2L for Two-Layer UV Effect	V	Print File Processing
VARDataFileName	V	Page Marking
VDISP	V	Page Marking
VPDISTPAT	C	Font and Color

VIPP® Commands, Markers, Transform Functions, and Variables	Type: C=Command, M=Marker,T= Transform Function,V=Variable	Function
VPOS	V	Page Layout
VSUB	T	Transform Control
VSUB2	T	Transform Control
VSUB3	T	Transform Control
VSUB4	T	Transform Control
XGF	C	Miscellaneous
XGFDEBUG	C	Design and Debug
XGFEND	C	Print File Processing
XGFENTRY	C	Miscellaneous
XGFRESDEF	C	Resource Control
XGFVER	V	Miscellaneous
XMLATL	C	Print File Processing
XMLATN	V	Print File Processing
XMLATV	V	Print File Processing
XMLDTH	V	Print File Processing
XMLPAR	V	Print File Processing
XMLPATH	V	Print File Processing
XMLSYN	C	Print File Processing
XMLTAG	V	Print File Processing
XMLVAL	V	Print File Processing
YINIT	V	Page Layout
ZSORT	C	Print File Processing
ZSPAGE	V	Print File Processing
ZSRECNUM	V	Print File Processing
ZSREPCNT	V	Print File Processing
ZSREPIDX	V	Print File Processing

2

VIPP® Commands

This chapter contains:

A VIPP® command is a stand-alone sequence made up of any number of operands and a VIPP® command keyword as described in the syntax. Operands, if any, are always placed before the command. Commands are all uppercase to prevent conflict with PostScript operators.

++ and –

These commands can be applied to numeric variables defined with **SETVAR** to increment or decrement them by one. They can be used to implement a counter and associated actions in conjunction with IF/ELSE/ENDIF.

Syntax

- `/VARname ++`
- `/VARname –`

Where

VARname must have been previously initialized with **SETVAR**.

Examples

Use this example to staple each ten-page document set on an NPS or FreeFlow Print Server printer.

```
/Staple /ON SETFINISHING
/VAR.CNT1 0 SETVAR
{ /VAR.CNT1 ++
IF VAR.CNT1 10 eq
{ ENDOFSET /VAR.CNT1 0 SETVAR }
ENDIF
} BEGINPAGE
```

Note: The variable type can be an integer or a string. A string variable can be printed or merged with the **VSUB** command.

Counters using a string value may hold numbers up to 25 digits.

Leading zeros present at initialization are preserved when the string is printed.

Example

```
/VAR1 (0000) SETVAR
/VAR1 ++
VAR1 SH
```

will print `0001`

Modes

This command is applicable in all modes.

Related Commands

- ADD
- SETVAR

- SUB

ABSPOS

Use **ABSPOS** in Multi-Up mode to place subsequent commands, such as **MOVETO** or **MOVEH**, in a position relative to the physical page rather than to the current logical page.

In general, this command is used to convert LCDS data streams using the DJDE GRAPHIC= statements in Multi-Up mode.

The effect of **ABSPOS** is automatically cancelled at the end of the page, or by any orientation command such as **PORT**, **LAND**, **IPORT**, and **ILAND**.

Syntax

```
ABSPOS
```

Examples

This example prints page 1 and page 2 side by side, reduced to a single physical page. image1.tif prints at x0, y1000 (current units) from the origin of the physical page, and overlaps the two logical pages.

```
TWOUP
Page 1 data/commands
PAGEBRK
Page 2 data/commands
ABSPOS
0 1000 MOVETO
(image1.tif) 1 0 ICALL
PAGEBRK
```

Modes

This command is applicable in all modes.

Related Commands

- MOVETO
- MOVEH

ACCLOG

Use **ACCLOG** to capture data when a VIPP® job processes, then log the data into the demographics output file, for example, .vpr and/or .vpd.

This command is only effective when the Demographics feature has been activated for the job, refer to the *FreeFlow VI Compose User Guide*.

The data captured by **ACCLOG** may include, but is not limited to:

- Detailed statistical information that is accumulated during the job using dedicated variables. Place the **ACCLOG** commands in **ENDJOB** to dump them at the end of the job.
- Detailed information on each record or page processed, which is selected for the purpose of job integrity checks. Place the **ACCLOG** commands in the DBM or in **BEGINPAGE**.
- Accounting information accumulated during the job for re-charge purposes.

Syntax

```
TagName Contents ACCLOG
TagName () ACCLOG
() () ACCLOG
```

Where:

(TagName)	is a string or variable representing a label to identify the information in contents.
(Contents)	is a string or variable containing the information to be captured.

When this string is empty (2nd syntax) TagName represents a label for a set of information provided in the subsequent **ACCLOG** commands. Groups can be nested on several levels.

When both strings are empty (3rd syntax) the current group of information, previously opened by a command using the 2nd syntax, is closed.

Caution: Do not execute **ACCLOG** after page initialization. Place the **ACCLOG** command before the first marking command, for example, after a **PAGEBRK** command, or in a **BEGINPAGE** or **/P ENDPAGE** procedure.

Examples

The following sequence may be placed at the beginning of a DBM to capture selected information for the current record:

```
(Customer) () ACCLOG
(FullName) ($$FIRST.$$NAME.) VSUB ACCLOG
(Address) ADDRESS VSUB ACCLOG
(City) CITY ACCLOG
() () ACCLOG
```

Data captured by **ACCLOG** is stored in the XML and/or database file, depending on the Demographics options.

In the XML file, `.vpd`, the data are deployed as a tree, each group being a node, under the root tag name `<ACCLOG>`.

In the database file data is dumped sequentially with the prefix number 4.

Modes

This command is applicable in all modes.

Related commands

None

ADD

ADD adds a value to a variable defined by **SETVAR** or an XML variable. When the variable is a number either a numeric string, a real, or an integer the operand must be a number and ADD performs the mathematical addition. When the variable is an array the operand must also be an array and **ADD** includes the operand items as new items in the variable array.

Syntax

```
/VARname additem ADD
/^XMLname additem ADD
```

Where

/VARname

Refers to a numeric variable previously initialized by **SETVAR**.

/^XMLname

Refers to an XML variable. An XML variable needs not be explicitly initialized. VIPP® initializes all XML variables to an empty string equivalent to a numeric string equal to zero.

additem

Is one of the following:

- A positive or negative number added to the variable when the variable itself is a number. It can be either an integer or a real or a numeric string. When large numbers are involved a numeric string is mandatory.
- An array to be appended to the variable array.

Numeric strings accommodate large numbers up to 40 digits, 25 digits for the integer part and 15 digits for the decimal part. In a numeric string the negative sign and the decimal delimiter are defined by the parameters /DecimalPoint and /NSign and can occur anywhere in the string.

It is mandatory to set these parameters with appropriate values to ensure accurate results. Defaults are defined in the file `/usr/xgf/src/xgf.def`. Characters in the numeric string other than these two plus the digits 0–9 are ignored.

The initial length of the string defined by SETVAR is automatically extended up to 40 digits when needed.

Reals and integers must be used only for small values <= 99999, for instance the implementation of a counter. The decimal delimiter, when present, is always the point (.). The negative sign, when present, is always the minus (-) and must be the first character.

Examples

```
/VAR.CNT1 0 SETVAR
/VAR.CNT1 12 ADD
/VAR.CNT1 -3 ADD
/VAR_SUM (0) SETVAR
/VAR_SUM (1'234'890'566,00-) ADD
```

This example shows how to use ADD to capture data in a line mode job and produce a chart on the page.

```
{ /VAR_CHARTDATA [ ] SETVAR
{ IF RPCOUNT 10 ge RPCOUNT 40 le and
```

```
{ /VAR_LINE RPCOUNT 0 100 GETFIELD %lines 10 to 40
/VAR_CHARTDATA
[ VAR_LINE 15 10 GETINTV %extract label
VAR_LINE 30 10 GETINTV %extract value
] ADD %accumulate in CHARTDATA
} ENDIF
} LNCOUNT REPEAT
x y MOVETO VAR_CHARTDATA 800 DRAWPIE %draw a pie
} ENDPAGE
```

This example shows how to use **ADD** to accumulate data in an array and produce a chart with them. Each line goes in a different part of the formatting resource (JDT/DBM/XJT), typically initialization, **BEGINPAGE**, **ENDPAGE**.

```
/VAR_CHART [] SETVAR %initialize array
...
/VAR_CHART [ VAR_LAB VAR_VAL ] ADD %accumulate label/value pairs
...
x y MOVETO VAR_CHART 800 DRAWPIE %draw chart
```

Modes

This command is applicable in all modes.

Related commands

- ++/-
- SETVAR
- SUB
- MUL
- DIV

ADVPAGE

ADVPAGE enables various options related to blank page and empty line processing.

Syntax

`option ADVPAGE`

Where:

option	is a number that may combine these values: **+1** print pages that consist of only one empty line **+2** print pages that consist of only one NMP line **+4** process empty lines for SETRCD tests

Pages that consist of only one empty line or a single NMP will be skipped by default. SETRCD tests are not applied on empty lines by default.

Modes

This command is applicable in line mode.

Related commands

SETPBRK

AZTEC

AZTEC creates and images an AZTEC barcode based on the specified string and parameter data. No special fonts are required.

Syntax

```
(data) AZTEC
(data) scale rotate align AZTEC
(data) [width] rotate align AZTEC
(data) [ /ELevel xx ... ] AZTEC
(data) [ /ELevel xx ... ] scale rotate align AZTEC
(data [ /ELevel xx ... ] [width] rotate align AZTEC
```

Where:

(data)	is a string containing the information to store in the symbol.
scale	is the scale value (default is 1).
rotate	is the rotation value (default is 0).
align	is one of these alignment codes which indicate which point of the barcode will be aligned on the secondary print position: **0** top left (default) **1** top right **2** top center **10** bottom left **11** bottom right **12** bottom center **20** center left **21** center right **22** center center

The optional array contains a list of key/value pairs that may be used to specify the following additional options:

/ELevel integer	is the error correction level between 0 and 99. The default is 23.
/MinDim integer	is the minimum dimension of the symbol.

Other options may be added in future releases.

Advanced features such as ECI and structure append may be supported in future releases.

Examples

```
(data) [ /ELevel 40 /MinDim 5 ] AZTEC
```

Modes

This command is applicable in all modes.

Related commands

- PDF417
- DATAMATRIX
- MAXICODE
- QRCODE
- MOVEH
- MOVEHR
- MOVETO

BACKSP_off

BACKSP_off disables backspace processing.

Syntax

```
BACKSP_off
```

Modes

This command is applicable in line mode only with no RPE.

Related commands

BACKSPP_on

BACKSPF_on

BACKSPF_on enables backspace processing with fixed fonts only.

Syntax

```
BACKSPF_on
```

Modes

This command is applicable in line mode only with no RPE.

Related commands

BACKSP_off

BACKSPP_on

BACKSPP_on enables backspace processing with fixed and proportional fonts.

Syntax

```
BACKSPP_on
```

Modes

This command is applicable in line mode only with no RPE.

Related commands

BACKSP_off, BACKSPF_on

BCALL

BCALL executes a fragment of VIPP® code encapsulated in a procedure. **BCALL** protects any settings inside the procedure from any effect outside. Settings in effect before the **BCALL** statement will remain active after the statement.

Protected settings include:

- Font as defined by **SETFONT** or instantiated by a font index
- Color as defined by **SETTXC** or instantiated by a color index
- Background as defined by **SETTXB** or instantiated by a **BAT** index
- Line spacing as defined by **SETLSP**
- Indentation as defined by **SETINDENT**
- Outline as defined by **SETOTL**, or instantiated by an **OTL** index

Syntax

```
{ VIPP code } BCALL
```

Where:

{ VIPP code }	is a fragment of VIPP® code that paints an element on the page.

Examples

```
{ /NHE 12 SETFONT
RED SETTXC
/UNDL SETTXB
(text text text ... ) 3 SHP
] BCALL
```

Modes

This command is applicable in all modes.

Related commands

FCALL, MOVETO, SCALL

BEGINARBM

BEGINARBM starts an bi-directional merge definition that will be selected by the **SETBIDI** command and used by the BIDI transform function.

The definition must end with an **ENDARBM** command.

Syntax

```
/mrg_name BEGINARBM
(char31) (char21) (char11) (charR1)
.....
(char3N) (char2N) (char1N) (charRN)
ENDARBM
```

Where:

/mrg_name	is the name of the table to be used by **SETBIDI**
(char3X)	is a string representing the third character of a triplet to be merged or 0 for a pair
(char2X)	is a string representing the second character of a pair or triplet to be merged
(char1X)	is a string representing the first character of a pair or triplet to be merged
(charRX)	is a string representing the character to be substituted to the pair or triplet

Note: Pre-defined tables for Windows1256 for Arabic Windows1255 for Hebrew, and UTF8 are provided in the bi-directional configuration file at `xgf/src/arb.def`.

Modes

This command is applicable in all modes.

Related commands

- BIDI
- SETBIDI
- BEGINARBT
- ENDARBT

BEGINARBT

BEGINARBT starts an bi-directional context definition that will be selected by the **SETBIDI** command and used by the BIDI transform function.

The definition must end with an **ENDARBT** command.

Syntax

```
/ctx_name BEGINARBT
(input1) (begin1) (middle1) (end1) (isolated1) join_group1
....
(inputN) (beginN) (middleN) (endN) (isolatedN) join_groupN
ENDARBT
```

Where:

/ctx_name	is the name of the table to be used by **SETBIDI**.
(inputX)	is a string representing a character in the input data string processed by the BIDI transform function.
(beginX)	is a string to be substituted when the character is at the beginning of a word. For an input string representing a digit (0–9) this string defines the alternate Hindi digit.
(middleX)	is a string to be substituted when the character is in the middle of a word.
(endX)	is a string to be substituted when the character is at the end of a word.
(isolatedX)	is a string to be substituted when the character is isolated.
join_groupX	is the join group to which this character belongs. One of: **1** Isolated right-to-left character **2** Arabic right connected character **3** Arabic right and left connected character **4** Arabic left connected character **5** Left-to-right character **6** special characters, isolated right-to-left or left-to-right depending on the adjacent characters.

When a character is not defined in the table it is not substituted and is assigned a join group of 5.

Each substituted string is related to a specific font mapping. Thus, the table selected by **SETBIDI** must match the font selected by **SETFONT** or **INDEXFONT**.

Note: Pre-defined tables for Windows1256 Arabic, Windows1255 Hebrew, and UTF8 are provided in the bi-directional configuration file in `xgf/src/arb.def`.

Modes

This command is applicable in all modes.

Related commands

- BIDI
- SETBIDI
- FCALL
- MOVETO
- SCALL
- ENDARBM

BEGINFRAME

BEGINFRAME defines actions to be performed at the beginning of each frame on a page. By default these actions are null. **BEGINFRAME** allows marking commands as actions, though it is usually used to set up some conditional logic and variables in order to execute specific actions prior to rendering the next frame.

Syntax

```
{ "start of frame" actions } BEGINFRAME
```

Modes

This command is applicable in all modes.

Related commands

FRCOUNT, NEWFRAME, SETLKF

BEGINIMP

BEGINIMP turns on the imposition feature. Use this feature to print a document that consists of a collection of images, segments, and/or EPS files so that the printed pages create a booklet. Use this command with a Multi-Up setting such as TWOUP, as well as a duplex mode such as **DUPLEX_on** or **TUMBLEDUPLEX_on**. This ensures that the printed document can be folded and stitched correctly. **BEGINIMP** is ended using **ENDIMP**.

Note: Do not associate this command with any imposition feature on the printer controller.

Syntax

```
BEGINIMP
calls to page elements using ICALL and/or SCALL
ENDIMP
```

Use this information to enable printing the imposed output in the reverse order:

[/IMPmode 0] SETPARAMS	the current and default mode produces pages in 1 to N order. The page order is [last first] [second last-1] ... [middle+1 middle].
[/IMPmode 1] SETPARAMS	produces pages in N to 1 order. The page order is [middle+1 middle] ... [second last-1] [last first].

Examples

```
TWOUP
TUMBLEDUPLEX_on
BEGINIMP
(report1.ps) RUNDD
ENDIMP
```

Modes

This command is applicable in all modes except line mode.

Related commands

- ENDIMP
- DUPLEX_off
- DUPLEX_on
- ICALL
- PAGEBRK
- RUNDD
- RUNTIF
- RUNPDF
- TUMBLEDUPLEX_off
- TUMBLEDUPLEX_on
- TWOUP
- TUMBLEDUPLEX_on
- SCALL

BEGINPAGE

BEGINPAGE defines actions to be performed at the beginning of each page. By default these actions are null. Do not use **BEGINPAGE** to perform marking commands, use it to perform tests on a page prior to the page being imaged. Refer to **ENDPAGE** for more information about page marking. This allows you to test for a value and perform an action, such as setting a new JDT file.

Note: Place the **BEGINPAGE** procedure before any marking commands, including the **MOVETO** command on the current page. If you place the **BEGINPAGE** command after a marking command, the **BEGINPAGE** command is not executed for the current page and is discarded by **PAGEBRK** at the end of the page.

Syntax

- `{ start of page actions } BEGINPAGE`
- `{ start of page actions } /M BEGINPAGE`
- `{ start of page actions } /B BEGINPAGE`
- `{ start of page actions } /Z BEGINPAGE`

{ start of page actions } BEGINPAGE

Defines an action on the base or slave JDT level.

{ start of page actions } /M BEGINPAGE

Defines an action on the master JDT level.

{ start of page actions } /B BEGINPAGE

Defines an action on the banner JDT level.

{ start of page actions } /Z BEGINPAGE

Defines a procedure that is executed at imposition time rather than composition time.

Use the /B switch when defining **BEGINPAGE** actions in the JDT for banner pages. Use the /M switch when defining **BEGINPAGE** actions in the master JDT. No switch is required when defining **BEGINPAGE** actions in the slave **JDT**.

The /Z switch is only relevant when Generic **ZSORT** is active, and is used to prevent simultaneous composition and imposition. It is primarily intended for conditional logic using the **PINDEX** variable because, when Generic **ZSORT** is active, **LPINDEX** is always equal to 1 at composition time.

Examples

This example shows how to staple sets of multi-copy pages when using noncollate mode. This avoids the need to place an ENDOFRUN command at the beginning of each page in the data.

- `/Staple /ON SETFINISHING`
- `3 SETCYCLECOPY`
- `COLLATE_off`
- `{ ENDOFRUN } BEGINPAGE`

This example shows how to capture the data field following **FORM=** on the banner page and use it to activate a JDT on the subsequent pages.

```
/BANNER 10 9 0 5 /eq (FORM=) SETPCD
{IF BANNER
{/VARjdt 0 5 8 /BANNER GETFIELD
($$VARjdt..jdt) VSUB 1 SETJDT
}
ENDIF
}/B BEGINPAGE
```

To test for a **START OF JOB** or **END OF JOB** banner page, use a **BEGINPAGE** procedure. This procedure will look for the text **START OF JOB** or **END OF JOB** and when found, set a null form (no form).

```
/StartBanner 3 3 5 13 /eq (START OF JOB:) SETPCD
/EndBanner   3 3 5 11 /eq (END OF JOB:) SETPCD
{
IF StartBanner EndBanner or
{ null SETFORM } ENDIF
}   BEGINPAGE
```

Use the or operator to set the condition to true when either **StartBanner** or **EndBanner** is true.

Modes

This command is applicable in all modes.

Related commands

- ENDOFRUN
- ENDPAGE
- GETFIELD
- SLIPSHEET

BEGINPCC

BEGINPCC starts a PCC table definition used later with the **SETPCC** command.

Syntax

```
/Staple /ON SETFINISHING
/pccname BEGINPCC
value [ pre_skip print_action post_skip ]
......
ENDPCC
```

Where:

value	is the PCC code (0–255 or 16#0–16#FF).
pre_skip	is one of the following: • a number of lines • a key to be assigned later by **SETVFU**.
print_action	is one of the following: • true (print the record) • false (do not print the record).
post_skip	is one of the following: • a number of lines • a key to be assigned later by **SETVFU**.

Examples

```
/ANSI BEGINPCC                          % (first entry is the default)
  16#20  [   1    true    0  ]   % asa blank :  space 1 line and print
  16#30  [   2    true    0  ]   % asa 0 : space 2 line and print
  16#2D  [   3    true    0  ]   % asa - : space 3 line and print
  16#2B  [   0    true    0  ]   % asa + : print without spacing
  16#31  [  /SK1  true    0  ]   % asa 1 : skip to channel 1 and print
  ....
ENDPCC
```

Modes

This command is applicable in line mode.

Related commands

ENDPCC, SETPCC, SETVFU

BEGINRPE

BEGINRPE starts an RPE library definition in a JDT file. Refer to RPE command information and to other related **RPE** commands for more information.

Syntax

```
Maxrpe BEGINRPE
```

Where:

Maxrpe	must be equal to or greater than the number of **FROMLINE** or **RPEKEY** commands following **BEGINRPE** and up to **ENDRPE**.

Note: Theoretically, the maximum number of entries in an RPE library or in an RPE entry is 65,535. However, a large number can affect performance or cause a stack overflow error.

Define RPE libraries in a master JDT

For more information on defining several RPE libraries in a master JDT, refer to INDEXRPE.

Cancel an active RPE

To cancel an active RPE when switching between JDTs use the following syntax without **ENDRPE**:

```
0 BEGINRPE
```

Modes

This command is applicable in line mode.

Related commands

- ENDRPE
- FROMLINE
- INDEXRPE
- RPEKEY
- STARTLM

BEGINTABLE

Use **BEGINTABLE** to initiate a table.xx

Syntax

```
[ /param1 value1 ... /paramX ValueX ] BEGINTABLE
```

Where:

/param	can be one of the following:
	/Margins [top bottom left right] default cell margins in current units
	/TableStroke GEPkey table border stroke
	/Width default width of the cells in current units
	/Height default minimum height of the cells in current units
	/MaxHeight default maximum height of the cells in current units
	/CellFill default color to fill the cells
	/CellStroke GEPkey default Gepkeys to stroke the borders of the cells
	/TextAtt default VIPP® code to set text attributes
	/Align default align attribute same as SHP

Modes

This command is applicable in all modes.

Related commands

ENDTABLE,SHROW

BEGINXPD

BEGINXPD starts an XML Processing Definition (XPD) table in an XML Job Ticket (XJT) file. It must be coupled with an **ENDXPD** command.

Between **BEGINXPD** and **ENDXPD** only definitions for tag actions and/or tag substitutions using **BTA/ ETA** and **BTS/ETS** commands are allowed.

Syntax

```
BEGINXPD
```

Inside BTA and BTS definitions these built-in variables are available:

XMLATL	list of all the attributes of an XML tag
XMLATN	attribute name
XMLATV	attribute value
XMLTAG	name of the current node
XMLVAL	contents of the current node
XMLPAR	name of the parent node of the current node
XMLPATH	VXVpath of the current node
XMLDTH	depth of the current node

Modes

This command is applicable in XML mode.

Related commands

- ENDXPD
- BTA
- BTS
- ETA
- ETS
- STARTXML

BOOKLETRANGE

BOOKLETRANGE is similar to **PAGERANGE** but is booklet oriented. **BOOKLETRANGE** works in conjunction with STARTBOOKLET/ ENDBOOKLET to allow selection of a range of booklets to print.

The **BOOKLETRANGE** command must be placed in the VIPP® job. It is recommended that you add the command at the beginning of the submission file.

Syntax

```
startbooklet endbooklet BOOKLETRANGE
```

Where

startbooklet	is the number of the first booklet to print.
endbooklet	is the number of the last booklet to print.

Depending on the /PageRange parameter set by **SETPARAMS** the job will complete normally or terminate with the message Selected booklets: startbooklet endbooklet.

Modes

This command is applicable in all modes.

Related commands

STARTBOOKLET,ENDBOOKLET

BOOKMARK

BOOKMARK creates an interactive bookmark within a PDF file.

Syntax

input_string is the string used to create the bookmark. Usually it is a field in database mode, or data extracted using **GETFIELD** in line mode.

Single byte data must be encoded using ISO-8859-1. Multi-byte data must be encoded using UTF8. It will be automatically converted into UTF16 by VI Compose for insertion in the PDF because this is the only multi-byte encoding supported by the PDF format. To trigger the conversion of UTF8 data to UTF16 the current font (selected by SETFONT or INDEXFONT) must have an UTF8 encoding.

count is the number of sub-bookmarks following this bookmark. (Default is 0.) It may be a variable. When it is positive the bookmark is open when the file is initially accessed. When negative it is closed.

color is a Colorkey defining the color used to display the bookmark text. (Default is BLACK.) Only plain RGB and grey scale color are supported

style is the style of the bookmark text:

0 plain (default)

1 italic

2 bold

3 bold and italic

/EX tells **BOOKMARK** to create an extended bookmark rather than a regular bookmark. An extended bookmark is a transparent PDF note, which is usually visible from the Comment section of the completed PDF only. /EX is valid only when using VI eCompose (VIeC).

Extended bookmarks support large blocks of text when using the Dispatch function of VIeC. (The contents of a regular bookmark is limited to 256 characters, extended bookmarks can contain as much as 64K.) VIeC processes the contents of the extended bookmark (depending on opt2) by appending it to the contents of the regular bookmark. An extended bookmark is always linked to a regular bookmark and must be created after the regular bookmark. Several extended bookmarks may be created in a sub-document (between two consecutive regular bookmarks). Refer to the *FreeFlow VI eCompose User Guide* for detailed information on VIeC and the VIeC splitter.

opt1 is an integer that tells the VIeC splitter how to handle the child PDF file:

0 Do not produce the child PDF file

1 Produce the child PDF file but do not include bookmark in it.

2 Produce the child PDF file and include bookmark in it. This is the default.

Note: When there is no extended bookmark the splitter will behave as described for opt1=2.

Note: All extended bookmarks within a sub-document require the same opt1 value.

opt2 is an integer that tells the VIeC splitter how to handle the extended bookmark:

0 Do nothing with this extended bookmark

1 Merge the extended bookmark with the associated regular bookmark (for index file processing), but do not include it in the child PDF

2 Merge the extended bookmark with the associated regular bookmark (for index file processing) and include it in the child PDF

Note: To insert comments or instructions in the main PDF file, use `opt2=0`.

Note: The combination of `opt1=0` and `opt2=2` is irrelevant. The software acts as `opt2=1`.

Examples

This example creates an extended bookmark that instructs the VIeC splitter to produce a child PDF that does not contain the extended bookmark. However, this extended bookmark will be included in the VIeC index file and, with appropriate VIeC Dispatch and mail server settings, will trigger an email to John Smith with the child PDF attached.

```
(John Smith@isp.com:April invoice:Dear John,\nAttached you will find your
invoice for April.\nBest regards,\nPaul Martin) [ /EX 2 1 ] BOOKMARK
```

Modes

This command is applicable in all modes.

Related commands

- GETFIELD
- PDFDEST
- SETPIF
- INDEXPIF
- PDFINFO
- PDFOPEN

BSTRIP_off

BSTRIP_off disables the stripping of leading and trailing blanks in delimited fields in database mode.

Syntax

```
BSTRIP_off
```

BSTRIP_off and other global commands such as **DUPLEX_on**, **SETDBSEP**, and **SETBUFSIZE**, must not be coded in the Data Base Master. Code it in the beginning of the database file before the **STARTDBM** command, or in an external Job Descriptor Ticket referenced by a **SETJDT** command placed before the **STARTDBM** command in the database file.

Examples

```
%!
DUPLEX_on
(;) SETDBSEP
BSTRIP_off
(cas.dbm) STARTDBM
....

%! database file
(cas.jdt) SETJDT
(cas.dbm) STARTDBM
...

%!PS-Adobe-2.0
%%Title: cas.jdt
%%Creator: CAS/RXCH
....
DUPLEX_on
(;) SETDBSEP
BSTRIP_off
....
```

Modes

This command is applicable in database mode.

Related commands

BSTRIP,GETINTV,STARTDBM

BTA

BTA starts a tag action definition. It must be coupled with an **ETA** command.

Syntax

```
BTA /VXVkey
{ start tag action }
{ end tag action }
{ partial contents tag action }
ETA
```

Where:

/VXVkey	is a `VXVname` referencing a node of the XML file. Because `VXVkey` references an XML node name (as opposed to a `VXVname` that references XML node contents) the "^" in the first position must be omitted. Contrary to stand-alone `VXVname` references, a `VXVname BTA` operand may be ambiguous. When this happens the definition will apply to all nodes whose `VXVpath` matches the `BTA VXVname`.
{ start tag action }	is a sequence of VIPP® commands that is executed each time the start-tag of the node is encountered.
{ end tag action }	is a sequence of VIPP® commands that is executed each time the end-tag of the node is encountered.
{ partial contents tag action }	is a sequence of VIPP® commands that is executed each time a start-tag is encountered while some node contents of its parent is pending.

Examples

In this example the partial node contents `is an` and `writer born in` trigger the execution of this process. This operand is optional. When not specified, the `{ end tag action }` operand is used instead.

```
<author>
<name>John Smith</name> is an <nat>English</nat>writer born in
<birth>1961</birth>.
</author>
```

This is an example of BTA used with the XML data shown in the previous example.

```
BTA /author
{}
{ x y MOVETO ^author 0 SHP }
ETA
```

Modes

This command is applicable in XML mode.

Related commands

- BEGINXPD
- ENDXPD
- BTS
- ETA
- ETS

- STARTXML

BTS

BTS starts a tag substitution definition. It must be coupled with an **ETS** command.

Syntax

```
BTS /VXVname
(start tag substitution string) | { start tag substitution action }
(end tag substitution string) |
{ end tag substitution action }
ETS:
```

Where

/VXVname	is a VXVname referencing a node of the XML file. Because VXVkey references an XML node name (as opposed to a VXVname that references XML node contents) the "^" in the first position must be omitted. Contrary to stand-alone VXVname references, a VXVname BTS operand may be ambiguous. When this happens the definition will apply to all nodes whose VXVpath matches the BTS VXVname.
(start tag substitution string)	defines a string that will be substituted to the start tag name making it part of the node contents of its parent.
{ start tag substitution action }	is a sequence of VIPP® commands that is executed each time the start-tag of the node is encountered. This sequence must deliver a string that will be substituted to the start tag name making it part of the node contents of its parent.
(end tag substitution string)	defines a string that will be substituted to the end tag name making it part of the node contents of its parent.
{ end tag substitution action }	is a sequence of VIPP® commands that is executed each time the end-tag of the node is encountered. This sequence must deliver a string that will be substituted to the end tag name making it part of the node contents of its parent.

Note: **BTS** substitutions take precedence over **BTA** actions. When a node matches a **BTS** entry, any matching **BTA** action is ignored.

Tip: To substitute tags with font, color, background, and **SST** or **PIF** indexes, use **BTS** and **ETS** definitions.

Examples

Assuming the following XML fragment:

```
<author>
<name>John Smith</name> is an English writer born in 1961.
</author>
```

This coding will print `John Smith is an English writer born in 1961.`

```
/U /UNDL INDEXBAT
/u null INDEXBAT
BEGINXPD
BTS /name (//U) (//u) ETS
BTA /author {} { ^author 0 SHP } ETA
ENDXPD
```

Modes

This command is applicable in XML mode.

Related commands

- BEGINXPD
- ENDXPD
- BTA
- ETA
- ETS
- STARTXML

CACHE

CACHE enables resource caching. **CACHE** converts a resource into a PostScript Form dictionary, which can then be used by the **SCALL**, **SETFORM**, or **SETBFORM** commands. **CACHE** is always combined with one of these commands by replacing (segmentname) or (formname) with (rname) **CACHE** as described below.

For PDF resources, **CACHE** uses **CropBox** or **MediaBox** to determine the image bounding box. **Trimbox** (used instead of CropBox up to Vi Compose 12.0) is no longer used.

Syntax

```
(rname) CACHE ... SCALL
(rname) CACHE ... SETFORM
(rname) CACHE ... SETBFORM
[(rname) CACHE (formname) ... ]SETFORM
[(rname) CACHE (formname) ... ]SETBFORM
```

For additional syntax descriptions refer to SETFORM, SETBFORM, and SCALL.

Where:

rname	can represent a VIPP® segment, an EPS file, a PostScript file, a JPEG file, or a TIFF.

When a VIPP® segment is used with **SETFORM/SETBFORM** its origin is placed at the bottom left corner of the page. To specify a different origin use CACHE/SCALL in an in-line form definition as in this example.

```
{ 50 80 MOVETO (logo1.seg) CACHE SCALL } SETFORM
```

VIPP® forms those encapsulated between braces { }, are not allowed as CACHE operands. When **FSHOW** is used in such forms, **CACHE** is implicitly applied.

CACHE can use files located in these resource directories:

- formlib (as defined by SETFPATH)
- imglib (as defined by SETIPATH)
- mislib (as defined by SETMPATH)

Examples

```
(car1.eps) CACHE .6 SCALL
(car1.eps) CACHE [600 300] 0 22 SCALL %for fit-in-box option
[(form1.ps) CACHE (form2.ps) CACHE ] SETFORM
```

Note: The **CACHE** command has PostScript implementation that is compatible with devices that use the Adobe Red Book PostScript Level 2 forms caching. With this cache method, resources are loaded and retrieved from the cache, using a device-rasterization format.

Modes

This command is applicable in all modes.

Related commands

- SCALL
- SETFORM

- SETBFORM
- SETFPATH
- SETIPATH
- SETMPATH

CASE

CASE is an alternative to IF/ELSE/ENDIF for multiple similar concurrent tests. Always use **ENDCASE** to close the CASE list.

Syntax

```
CASE reference_variable { default action }
  choice1 { action 1 }
  choice2 { action 2 }
  ...
  choicen { action n }
ENDCASE

CASE reference_variable { default action }
  [ choice1a choice1b ... choice1x ] { action 1 }
        choice2 { action 2 }
        ...
        choiceN { action N }
ENDCASE
```

Where:

reference_variable	is a variable that represents a string or number.
default action	is a sequence of native mode commands executed when reference _variable is not equal to any choicen items.
choicen	is a string or number compared to the reference_variable.
action n	is a sequence of native mode commands executed when choicen is equal to reference_variable. This can include additional IF/ELSE statements.
choice1a choice1b ... choice1x	is a list of possible choices associated with a single action.

Examples

In this example, the salutation used differs according to the value assigned to VAR.SCORE. When VAR. SCORE is not 1, 2, or 3, the default value Dear Sirs, is used.

```
CASE VAR.SCORE {(Dear Sirs,) SHL}
(1) {(Dear Mr. $$NAME.,) VSUB SHL}
(2) {(Dear Mrs. $$NAME.,) VSUB SHL}
(3) {(Dear Miss $$NAME.,) VSUB SHL}
ENDCASE
```

The syntax can be expanded to include IF/ELSE statements inside a CASE statement.

```
CASE VARbranch { null SETFORM }         % branch unknown
(7481)  { IF STATE (IL) eq
             {(Illinoisregion1) SETFORM }
              ELSE
             { (region1 ) SETFORM } ENDIF
(7483)   { (region1) SETFORM }
(7496)   { (region2)  SETFORM ]
ENDCASE
```

Related commands

ENDCASE,IF/ELSE/ELIF/ENDIF

CHKPOINT

CHKPOINT defines an end of set point for cycle copy in collate mode. By default, the end of set point is the end of file in line mode. There is no default end of set point in native mode. Use **%%EOF** at the end of every data set or database file to terminate the file.

Syntax

```
CHKPOINT
```

Examples

This example is of a data set using native mode **CHKPOINT** in a data stream.

```
() STARTLM
Page 1
Page 2
Page 3
%%XGF CHKPOINT
PAGE 4
PAGE 5
PAGE 6
%%EOF
```

Use **CHKPOINT** in a **SETPCD** command when there is something in the data that can be used as a trigger between the sets.

```
/SetStart 1 1 0 6 /eq (Page 1) SETPCD
{ IF SetStart
{ CHKPOINT } ENDIF
} BEGINPAGE
() STARTLM
Page 1
Page 2
Page 3
Page 1
Page 2
Page 2
%%EOF
```

Use **CHKPOINT** with a counter there is a known and consistent number of pages in each set (three in this case).

```
/VARPageCount 1 SETVAR
{ IF VARPageCount 3 eq
{ CHKPOINT
/VARPageCount 1 SETVAR % Resets counter to 1
} ENDIF
} BEGINPAGE
{/VARPageCount ++ } /P ENDPAGE % Increment page count
```

Modes

This command is applicable in all modes.

Related commands

COLLATE_on, SETCYCLECOPY

COLLATE_dbm

COLLATE_dbm enables a new collation mode in database mode. When this collation mode is enabled, the Data Base Master is called for each record the number of times specified by SETCYCLECOPY.

Syntax

```
COLLATE_dbm
```

Examples

This example prints two copies of a database mode job using **COLLATE_dbm** and **SETCYCLECOPY**.

```
% DBM code

COLLATE_dbm
2 SETCYCLECOPY

/NTMR 14 SETFONT
300 2700 MOVETO
($$Fname. $$LName.) VSUB 0 SHP
% other code
PAGEBRK
```

Note: Code any collation command before the **SETCYCLECOPY** command.

Modes

This command is applicable in database mode.

Related commands

SETCYCLECOPY

COLLATE_off

COLLATE_off sets the **noncollate** mode for cycle copy mode. **COLLATE_on** is the default. **Noncollate** mode indicates that the requested number of copies is immediately produced after each logical page.

Syntax

```
COLLATE_off
```

Note: Code any collation command before the **SETCYCLECOPY** command.

Tip: Use this command with **TWOUP** to produce two reduced copies of a report on a normal size sheet of paper in one pass.

Modes

This command is applicable in all modes.

Related commands

COLLATE_on, COLLATE_dbm, SETCYCLECOPY

COLLATE_on

COLLATE_on sets collate mode for cycle copy mode. This is the default. In collate mode, multiple copies are created on a job or set basis. For more information, refer to CHKPOINT.

Syntax

```
COLLATE_on
```

Note: Code any collation command before the **SETCYCLECOPY** command.

Modes

This command is applicable in all modes.

Related commands

CHKPOINT, COLLATE_off, SETCYCLECOPY

COPYRANGE

COPYRANGE causes the associated **FROMLINE** or **RPEKEY** Record Processing Entry definition, and all following definitions, to apply only to the specified copies [c1, c2, ... cn]. The number of copies is defined by **SETCYCLECOPY**. Refer to FROM LINE or RPEKEY for the description of the RPE entry parameters.

Syntax

```
Linenumber FROMLINE [ c1 c2 ... cn ]COPYRANGE
/rpekeyname RPEKEY [ c1 c2 ... cn ]COPYRANGE
[ align rotate Xinit Xdispl Yinit Ydispl recpos length /font Colorkey ]
...
[ align rotate Xinit Xdispl Yinit Ydispl recpos length /font Colorkey ]
```

Note: Use this command only with a **FROMLINE** command or a **RPEKEY** command.

Note: Use this command to set up a different layout for each copy.

Modes

This command is applicable in line mode.

Related commands

- FROMLINE
- RPEKEY
- SETCYCLECOPY
- SETTXC
- SETPAT

CUTMARK

The **CUTMARK** command prints marks intended to guide trimming of Multi-Up layouts or areas falling off the finished page.

Syntax

```
Xpox Ypos length width option CUTMARK
```

Where:

Xpos and Ypos	are the origin of the mark in current units
length	is the length of the mark in current units
width	is the width of the lines in current units
option	is an integer representing the type of the mark: **0** top left corner **1** bottom left corner **2** bottom right corner **3** top right corner **4** vertical down **5** vertical up **6** vertical center **7** horizontal right **8** horizontal left **9** horizontal center

Examples

```
ORITL
200 200 150 2 0 CUTMARK
200 PAGEH'-'200 150 2 1 CUTMARK
PAGEW'-'200 PAGEH'-'200 150 2 2 CUTMARK
PAGEW'-'200 200 150 2 3 CUTMARK
```

Modes

This command is applicable in all modes.

Related commands

SETMULTIUP, PAGEH, PAGEW

DATAMATRIX

DATAMATRIX creates and images a Datamatrix ECC200 barcode based on the specified string and parameter data. No special fonts are required.

Syntax

```
(data) DATAMATRIX
(data) scale rotate align DATAMATRIX
(data) [ /key1 value1 ... /keyN valueN ] DATAMATRIX
(data) [ /key1 value1 ... /keyN valueN ] scale rotate align DATAMATRIX
(data) [ /key1 value1 ... /keyN valueN ] [width] rotate align DATAMATRIX
```

Where:

(data)	is a string containing the information to store in the symbol.
scale	is the scale value (default is 1)
width	is the requested width of the symbol in current units. With this syntax the width of the symbol is kept constant by scaling appropriately up or down, regardless of the amount of data.

Note: The barcode is stretched or shrunk proportionally to fit the specified width. Specify a width that is capable of displaying a readable barcode.

rotate	is the rotation value (default is 0)
align	is one of these alignment codes (which indicate which point of the barcode can be aligned on the secondary print position): **0** top left (default) **1** top right **2** top center **10** bottom left **11** bottom right **12** bottom center **20** center left **21** center right **22** center center

The optional array contains a list of key or value pairs that can be used to specify the following additional options:

/MinDim	Minimum dimension of the symbol. Integer between 8 and 144 (default is 8).
/Rect integer	One of: **0** always produce a square symbol (default) **1** produce a rectangular symbol if data can fit in one of them
/Encod /enc_option	One of: **/ASCII** ASCII encoding (default) **/Base256** Base256 encoding **/C40** optimized for digits, uppercase letters and space. **/C40+** optimized for digits, uppercase letters and space with compression of digit sequences > 13 digits **/Text** optimized for digits, lowercase letters and space **/Text+** optimized for digits, lowercase letters and space with compression of digit sequences > 13 digits

Examples

```
(data...) [ /Rect 1 /Encod /Base256 ] DATAMATRIX
(ABC 12345678998765432123) [ /Encod /C40+ ] [400] 0 10 DATAMATRIX
(data...) [/MinDim 36 ] DATAMATRIX
```

Modes

This command is applicable in all modes.

Related commands

- AZTEC
- PDF417
- MAXICODE
- QRCODE
- MOVEH
- MOVEHR
- MOVETO

DEFINELAYOUT

DEFINELAYOUT creates and register a Multi-Up layout to be used later by **SETLAYOUT**.

Syntax

```
/layoutname
[ /PageWidth    pagewidth
  /PageHeight   pageheight
  /TopBleed     topbleed
  /LeftBleed    leftbleed
  /RightBleed   rightbleed
  /BottomBleed  bottombleed
  /HGutter      hgutter
  /VGutter      vgutter
  /Across       across
  /Down         down
  /Rotate       rotate
  /FillOrder    fillorder
  /LayoutMarks  markoption
  /MarkLength   marklength
  /MarkWidth    markwidth
  /MarkOffset   markoffset
] DEFINELAYOUT
```

Refer to SETLAYOUT for a description of the parameters.

Examples

```
/2x3Letter90
[ /PageWidth  216
  /PageHeight  270
  /HGutter  18
  /VGutter  18
  /Across  2
  /Down  3
  /Rotate  90
] DEFINELAYOUT
```

Modes

This command is applicable in all modes.

Related commands

SETLAYOUT

DIV

DIV divides a numeric variable defined by **SETVAR**, or an XML variable by a number.

Syntax

`/VARname number DIV`

`/^XMLname number DIV`

Where:

VARname	refers to a numeric variable previously initialized by **SETVAR**.
/^XMLname	refers to an XML variable. In general XML variables do not need to be explicitly initialized. VI Compose initializes all XML variables to an empty string, which is equivalent to a numeric string equal to zero.
number	is the number by which the variable is divided. It can be an integer, a real, or a numeric string. When large numbers are involved a numeric string is mandatory.

`/Numeric` strings accommodate large numbers up to 40 digits, 25 digits for the integer part and 15 digits for the decimal part. In a numeric string the negative sign and the decimal delimiter are defined by the parameters /DecimalPoint and /NSign and can occur anywhere in the string.

It is mandatory to set these parameters with appropriate values to ensure accurate results. Defaults are defined in the file `/usr/xgf/src/xgf.def`. Characters in the numeric string other than these two plus the digits 0–9 are ignored.

The initial length of the string defined by **SETVAR** is automatically extended up to 40 digits when needed.

Reals and integers can be used only for small values <= 99999. For instance the implementation of a counter. The decimal delimiter, when present, is always the point `(.)`. The negative sign, when present, is always the minus `(-)` and can be the first character.

Modes

This command is applicable in all modes.

Related commands

- ADD
- SUB
- MUL
- SETVAR
- ++/-

DJDEBEGIN

Use the **DJDEBEGIN** command to process the LCDS BEGIN parameter in a PROCESSDJDE procedure.

Syntax

```
( DJDE BEGIN parameter ) DJDEBEGIN
```

IN and CM LCDS units are supported inside the operand string. Multiple **BEGIN** commands are supported through multiple **DJDEBEGIN** commands.

Examples

This is an example of DJDEBEGIN.

```
(1.2 CM, 5 CM) DJDEBEGIN
```

This is an example in PROCESSDJDE.

```
{ CASE DJDECMD
  ...
  (BEGIN) { DJDEPAR DJDEBEGIN }
  ...
 ENDCASE
} 0 (DJDE) 3 PROCESSDJDE
```

Note: This command makes an implicit call to **SETMULTIUP**, therefore **SETMULTIUP** cannot be used explicitly when **DJDEBEGIN** is used.

Modes

This command is applicable in line mode only.

Related commands

PROCESSDJDE,SETMULTIUP

DRAWB and DRAWBR

DRAWB draws a box with square corners. **DRAWBR** draws a box with rounded corners. The box is outlined and filled according to the GEPkey. These commands support strings as operands allowing you to use DBM variables as operands.

Note: Use **DRAWC** to draw circles or ellipses.

Syntax

```
Xpos Ypos width height GEPkey DRAWB
```

```
Xpos Ypos width height GEPkey radius DRAWBR
```

Where:

Xpos and Ypos	are the position of the upper left corner.
width	is the width.
height	is the height.
radius	is the corner radius, width=height=2xradius generates a circle.

The predefined GEPkeys are as follows:

FBLACK	S1	S2	S3	D1	D2	D3
XLT	XLT_S1	XLT_S2	XLT_S3	XLT_D1	XLT_D2	XLT_D3
LT	LT_S1	LT_S2	LT_S3	LT_D1	LT_D2	LT_D3
LMED	LMED_S1	LMED_S2	LMED_S3	LMED_D1	LMED_D2	LMED_D3
MED	MED_S1	MED_S2	MED_S3	MED_D1	MED_D2	MED_D3
DMED	DMED_S1	DMED_S2	DMED_S3	DMED_D1	DMED_D2	DMED_D3
DRK	DRK_S1	DRK_S2	DRK_S3	DRK_D1	DRK_D2	DRK_D3
XDRK	XDRK_S1	XDRK_S2	XDRK_S3	XDRK_D1	XDRK_D2	XDRK_D3

Examples

```
710     1250    1770    950     XLT         DRAWB   % shaded box
1250    900     1230    0       S1          DRAWB   % horizontal line
1250    900     0       1230    S1          DRAWB   % vertical line
710     1250    1770    950     LT_S1 30    DRAWBR  % shaded box + border + round corners
110     00      220     220     D1 110      DRAWBR  % circle
```

Modes

These commands are applicable in all modes.

Related commands

- DRAWC
- TPATH
- SHPATH

DRAWBAR

DRAWBAR draws a bar chart. This command places the bottom left origin of the bar chart at the current secondary print position.

Syntax

```
[ lab1 value1 lab2 value2 … labN valueN ]width height DRAWBAR
[ lab1 value1 lab2 value2 … labN valueN ]width height option DRAWBAR
[ lab1 [val11 val12 ..val 1M] … labN [valN1 valN2 ..val NM] ]  width height DRAWBAR
[ lab1 [val11 val12 ..val 1M] … labN [valN1 valN2 ..val NM] ]  width height option DRAWBAR
ddg_index width height DRAWBAR
ddg_index width height option DRAWBAR
```

Where:

labX	is the string label for a bar item. Variables can be used. Attributes switches such as font, and color are allowed in the string or variable contents. Label placement is determined by the /PrintLabel and /ChartDir parameters. For vertical bar charts word wrapping is performed on the labels based on the bar width. For horizontal bar charts word wrapping is performed based on a percentage of the chart width set by the /LabelColw parameter.
valueX	is the value (string or real) for a bar item. Variables can be used. For formatting options, refer to /Format.
[valX1 valX2 .. valXM]	is the set of string values (string or real) for a bar item. Variables can be used. When a set of values is provided, matching bars are stacked on top of each other. The number of values must be the same in all sets. The number of colors in the /ColorTable parameter must match the number of values in a set. Spot labelling for these colors can be provided using the /SpotLabels parameter. For formatting options, refer to /Format.
width	is the width of the bar chart area.
height	is the height of the bar chart area.
option	is detailed in Parameter Descriptions. All of the parameters have default values and can be omitted. All parameters set here can temporarily override the default value set by **SETPARAMS** and only apply to that command. The default values can be restored for subsequent commands.
ddg_index	refers to a list of label and values captured by an RPE entry, only zero is currently supported. Refer to FROMLINE align parameter.

Examples

```
[ (FF) 100 (US$) 250 (DM) 150 ] 300 200 DRAWBAR
[ (FF) 100 (US$) 250 (DM) 150 ] 300 200 /117 DRAWBAR
```

This is an example of **DRAWBAR** with multiple values stacked.

```
[ (FF) [ 100 20 87 ] (US$) [ 250 120 350 ] (DM) [ 150 75 123 ] ]
300 200 [ /SpotLabels [ (Cash) (Checks) (Credit Cards) ] /ColorTable
```

```
[ RED BLUE GREEN] ] DRAWBAR
```

This is an example of a stacked bar chart using an /OffsetValue to display stack values and totals:

```
/I /NHEB 11 INDEXFONT
/V /NHEB 14 INDEXFONT
/L 60 INDEXLSP
/W WHITE INDEXCOLOR

[ labx valx ....]
width height
[ /OffsetValue          [-2 1.3]
  /FORMATV              (//V//LTotal\n$@@@@#.00)
  /FORMATVI             (//I//W$@@@@#)
]DRAWBAR
```

Modes

This command is applicable in all modes.

Related commands

- SETPARAMS
- RPEKEY
- FROMLINE
- DRAWPIE
- DRAWCRV

DRAWBC

The **DRAWBC** command draws supported linear barcodes without relying on any specific font. A font name and font size for the human-readable text, can be supplied prior to the **DRAWBC** command using SETFONT or a font index value.

Syntax

```
(data) /BCkey option DRAWBC
(data) /BCkey [scaleH scaleV] option DRAWBC
```

Where:

(data)	is a numeric string containing the data to encode in the barcode.
/BCkey	is one of: **/EAN8** **/EAN13** **/EAN5** **/UPCA** **/UPCE** **/DATABAR**. For more information, refer to DRAWBC (Extension to Support GS1 Databar)
scaleH	is an optional horizontal scaling factor (default is 1).
scaleV	is an optional vertical scaling factor (default is 1).
option	is one of: **0** do not print the human-readable digits **10** print the human-readable digits

Examples

```
(78858101497) /UPCA 10 DRAWBC
```

```
(0123456) /UPCE 10 DRAWBC
```

```
(1123456) /UPCE [.5 1] DRAWBC
```

```
(1123456) /UPCE [1 .5] DRAWBC
```

Modes

This command is applicable in all modes.

Related commands

UPCA

DRAWBC (Extension to Support GS1 Databar)

The **DRAWBC** command has been extended to include support for GS1 Databar barcodes.

Description

The GS1 DataBar barcode is based on a family of symbols often used in the GS1 DataBar Coupon. Coupon codes commonly used in retail. GS1 DataBar was formerly known as Reduced Space Symbology (RSS) and has been renamed to align with the name of the GS1 organization.

These barcodes can encode up to 14 digits, which makes them suitable for GTIN 8, 12, 13 and 14.

GS1 DataBar Expanded and GS1 DataBar Expanded Stacked can encode up to 74 numeric or 41 alphanumeric characters, and provide the capability to utilize all GS1 Application Identifiers (for example, expiration date, batch and serial number). These bar codes are often used in manufacturer coupons.

The VI Suite supports seven GS1 DataBar barcodes. For more information about GS1 DataBar barcodes, please review the GS1 DataBar specification, available on the internet.

- The data in the barcodes listed here (option Y = 0 - 3) can include 13 or 14 digits. When 14 digits are entered, the 14th digit is considered a placeholder and can be replaced automatically with a calculated check digit.
  - GS1 DataBar Omnidirectional
  - GS1 DataBar Truncated
  - GS1 DataBar Stacked
  - GS1 DataBar Stacked Omnidirectional
- The data in the barcode listed here (option Y = 4) can include 13 or 14 digits. When 14 digits are entered, the 14th digit is considered a placeholder and can be replaced automatically with a calculated check digit. The indicator digit (first digit) must be a 0 or 1.
  - GS1 DataBar Limited
- The barcodes listed here (option Y = 5 or 6) are capable of encoding a 14 digit GTIN (must be 14 digits in length), and additional data (up to 74 numeric or 41 alphabetic characters). Refer to the GS1 Databar ISO specification for more information.
  - GS1 DataBar Expanded
  - GS1 DataBar Expanded Stacked

GS1 Databar command syntax in DRAWBC

The **DRAWBC** command has the following generic syntax:

```
(data) /BCkey option DRAWBC
```

or

```
(data) /BCkey [scaleH scaleV] option DRAWBC
```

Where:

/BCkey	is /DATABAR
Option	is a 4 digit number (ZZXY) where:
	ZZ Segments per row. Supported values are: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20. Used only for GS1 DataBar Expanded Stacked barcode.
	X: 0 do not print the human readable digits
	1 Print the human readable digits
	Y: 0 GS1 DataBar Omnidirectional
	1 GS1 DataBar Truncated
	2 GS1 DataBar Stacked
	3 GS1 DataBar Stacked Omnidirectional
	4 GS1 DataBar Limited
	5 GS1 DataBar Expanded
	6 GS1 DataBar Expanded Stacked

Example

```
((01) 04412345678909) /DATABAR 10 DRAWBC % GS1 Databar Omnidirectional with
human readable digits.
```

Where option is currently defined as:

1	print the human readable digits
0	omnidirectional

Valid Inputs

For all barcodes except GS1 Databar Expanded and GS1 Databar Expanded Stacked

GS1 Databar accepts 13 or 14 digits (option 5 and 6 only 14 digits plus optional data). When 14 digits are entered, the 14th digit is replaced with a calculated check digit. For example:

```
(0123456789101112)
```

is truncated to:

```
(0123456789101).
```

If the input is less than 13 digits leading 0's can be added. For example, valid input for:

```
(123456789)
```

is:

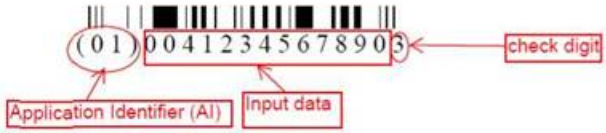
```
(0000123456789).
```

The barcode generator accepts inputs with or without an Application Identifier (AI). AI is not a mandatory input, however, the output always has leading AI in the HRI display.

Input with AI

The example below includes an Application Identifier of ().

```
((01)00412345678909) /DATABAR 11 DRAWBC
```



Note: In the above example the 14th character of the input string (9) can be discarded and replaced with a check digit calculated by the **DRAWBC** command.

Input without AI

The example below does not include an Application Identifier, nor does it include Human Readable Interpretation (HRI).

```
(0041234567890) /DATABAR 01 DRAWBC
```

The example below does not include an Application Identifier, however, it does include Human Readable Interpretation (HRI).

```
(0041234567890) /DATABAR 11 DRAWBC
```

GS1 Databar Expanded and GS1 Databar Expanded Stacked

GS1 DataBar Expanded is a variable length linear symbology capable of encoding up to 74 numeric or 41 alphabetic characters of AI element string data internally represented as a binary number. A 14 digit Application Identifier is mandatory in GS1 Databar Expanded and Expanded Stacked barcodes.

Valid input examples:

```
(01)90012345678908(3103)012233(15)991231
(01)00012345678905(10)ABC123
```

Input without AI is invalid:

```
00012345678905(10)ABC123
```

Usage of GS1 Databar in DRAWBC

Option values:

00	GS1 DataBar Omnidirectional
10	GS1 DataBar Omnidirectional with HRI
01	GS1 DataBar Truncated
11	GS1 DataBar Truncated with HRI
02	GS1 DataBar Stacked

12	GS1 DataBar Stacked with HRI
03	GS1 DataBar Stacked Omnidirectional
13	GS1 DataBar Stacked Omnidirectional with HRI
04	GS1 DataBar Limited
14	GS1 DataBar Limited with HRI
05	GS1 DataBar Expanded
15	GS1 DataBar Expanded with HRI
06	GS1 DataBar Expanded Stacked
16	GS1 DataBar Expanded Stacked with HRI

```
((01)24012345678905) /DATABAR 10 DRAWBC    % Generate GS1 DataBar Omnidirectional with HRI

(24012345678905) /DATABAR 10 DRAWBC    % Generate same GS1 DataBar Omnidirectional with HRI as
above

((01)15012345678907) /DATABAR [.5 1] 04 DRAWBC    %Generate GS1 DataBar Limited without HRI
with scale x:0.5 , y:1

((01)00012345678905) /DATABAR [1.5 1.5] 02 DRAWBC    %Generate GS1 DataBar Stacked without HRI
with scale x:1.5, y:1.5
```

DRAWBM and DRAWBRM

DRAWBM draws multiple boxes with square corners. **DRAWBRM** draws multiple boxes with rounded corners. The boxes are outlined and/or filled according to the GEPkey. These commands also support strings as operands allowing you to use DBM variables as operands.

Note: Use **DRAWC** to draw circles or ellipses.

Syntax

```
Xpos Ypos width height GEPkey repeat Xdispl Ydispl DRAWBM
```

```
Xpos Ypos width height GEPkey radius repeat Xdispl Ydispl DRAWBRM
```

Where:

Xpos and Ypos	are the positions of the upper left corner of the first box
width	is the width.
height	is the height.
radius	is the corner radius, width=height=2xradius generates a circle.
repeat	gives the number of boxes to draw.
Xdispl and Ydispl	are added to the Xpos and Ypos of the previous box to provide the position of the upper left corner for the next box to be drawn. Ydispl displacement is up when ORIBL is used or down when using ORITL. Negative numbers invert the displacement direction. The default is ORIBL.

Examples

```
120 360 1160 45 LT 20 0 100 DRAWBM % zebra
```

Modes

These commands are applicable in all modes.

Related commands

DRAWC

DRAWC

DRAWC draws a circle or an ellipse. The circle is outlined and filled according to the GEPkey. This command supports strings as operands allowing you to use DBM variables as operands.

Syntax

```
Xpos Ypos width height GEPkey DRAWC
```

Where

Xpos and Ypos	are the positions of the upper left corner of the bounding box of the circle or ellipse in current units.
width and height	are the width and height of the circle or ellipse in current units. When the width and height are equal, a circle is drawn. When they differ, an ellipse is drawn.
GEPkey	is the Graphical Element Property key used to draw the circle.

Examples

```
710 1250 900 900 S1 DRAWC          % outlined circle
710 1250 900 900 XLT_S1 DRAWC      % shaded and outlined circle
710 1250 900 600 S1 DRAWC          % outlined horizontal ellipse
710 1250 600 900 XLT DRAWC         % shaded vertical ellipse
```

Modes

This command is applicable in all modes.

Related commands

- SETGEP
- DRAWB and DRAWBR
- TPATH
- SHPATH
- OTCLIP and ITCLIP

DRAWCRV

DRAWCRV draws a curve chart. This command places the bottom left origin of the chart at the current secondary print position.

Syntax

```
[ label/value list ] width height DRAWCRV
[ label/value list ] width height option DRAWCRV
ddg_index width height DRAWCRV
ddg_index width height option DRAWCRV
```

Where:

label/value list	is a list of label/value pairs.
width	is the width of the curve chart area.
height	is the height of the curve chart area.
option	is detailed in Parameter Descriptions. All of the parameters have default values and can be omitted. All parameters set here can override temporarily the default value set by SETPARAMS and only apply to that command. The default values can be restored for subsequent commands.
ddg_index	refers to a list of label/values captured by an RPE entry (only zero is currently supported). Refer to FROMLINE align parameter.

Multiple charts drawn with **DRAWCRV** can be stacked using the /Stack parameter. When drawing stacked charts the following rules must be followed:

- /MaxVal and /MinVal parameters can be set to ensure a common scale for the stacked charts.
- All stack charts can share the same width and height.
- Labels can be specified only on the first chart.

Examples

```
[ (FF) 100 (US$) 250 (DM) 150 ] 300 200 DRAWCRV
[ (FF) 100 (US$) 250 (DM) 150 ] 300 200 /83 DRAWCRV
```

This example shows how to draw three stacked charts:

```
[/3D true /MaxVal 1000 /MinVal 500 /BGColor XLGREEN ] SETPARAMS
[ (1998) 623 (1999) 556 (2000) 690 ] 1600 900 [ /ColorTable [RED] ] DRAWCRV
[ () 720 () 656 () 840 ] 1600 900 [ /Stack true /ColorTable [GREEN] ] DRAWCRV
[ () 840 () 956 () 590 ] 1600 900 [ /Stack true /ColorTable [BLUE] ] DRAWCRV
```

Modes

This command is applicable in all modes.

Related commands

- SETPARAMS
- RPEKEY

- FROMLINE
- DRAWPIE
- DRAWBAR

DRAWPAR

DRAWPAR is a chart command used to draw Pareto charts. A Pareto chart combines a bar chart and a line chart using the same set of values. It is used to display the relative importance of the differences between groups of data. The bar chart displays the set of values as would **DRAWBAR**. The line chart represents the cumulative percentage related to the sum of the values. The values are expected in descending order.

Syntax

```
[ lab1 val1 ... labN valN ] width height DRAWPAR
[ lab1 val1 ... labN valN ] width height option1 DRAWPAR
[ lab1 val1 ... labN valN ] width height option1 option2 DRAWPAR
```

Where:

option1	is related to the bar chart
option2	is related to the line chart.

All other parameters are similar to those of the **DRAWBAR** command.

Examples

```
/VAR_SAMPLE
[ (Traffic) 92 (Child Care) 55 (Public Transportation) 42
  (Weather) 30 (Overslept)  25 (Emergency)  15 (Other)  7
] SETVAR

VAR_SAMPLE 2500 1638 DRAWPAR            % using default parameters

VAR_SAMPLE 2500 1638
[ /3D true
  /BGColor GR_SUN1H
  /BGLineColor WHITE
  /PrintValue true
  /ColorTable [XLGREEN]
  /BarSpace .1
]
[ /BarSpace .1 ]
DRAWPAR
```

The following default parameters are forced for Pareto charts:

- Bar chart
  - /MaxVal sum_of_values
  - /ScaleStep (10)
  - /PrintScale 1
  - /ColorTable [MBLUE]
  - /BarSpace .03
  - /SliceSepWidth 0
  - /BGLineColor LIGHT
  - /3DThickness .1 /Format (@@@@@@@@@@#)
- Line chart:

- /BGColor null
- /OriLine 21
- /PrintScale 2
- /MaxVal 100
- /PlotSymbol [(l) current_size RED]
- /ColorTable [RED]
- /BGLineColor WHITE
- /Format (@@# %)

Modes

This command is applicable in all modes.

Related commands

- DRAWBAR
- DRAWPIE
- DRAWCRV
- DRAWRDR

DRAWPATH and DRAWPATHR

DRAWPATH draws a path using a combination of straight lines and/or Bezier curves. **DRAWPATHR** draws the path with rounded corners.

Syntax

```
[Xpos Ypos [Point1] [Point2] ... [PointN]] GEPkey  DRAWPATH

[Xpos Ypos [Point1] [Point2] ... [PointN]] GEPkey  radius  DRAWPATHR

[ Xpos1 Ypos1 [Point11] [Point12] ... [Point1N}     % path 1
  Xpos2 Ypos2 [Point21] [Point22] ... [Point2N}     % path 2
  ...
] GEPkey  DRAWPATH
```

Where

Xpos Ypos	are the coordinates in current units of the origin of the path.
PointX	defines consecutive points joining the previous point with either a straight line or a Bezier curve: [Xpos1 Ypos1] are coordinates of the end of a straight line segment. [Xpos1 Ypos1 Xpos2 Ypos2 Xpos3 Ypos3] are coordinates of the Bezier curve control points.
GEPkey	defines the Graphical Element Property key to outline or fill the path.
radius	is the corner radius.

Examples

```
[200 200 [200 1200] [1400 200 300 300 400 450] [200 200]] S1 DRAWPATH
```

Modes

This command is applicable in all modes.

Related commands

- DRAWPOL
- DRAWB and DRAWBR
- SETGEP
- TPATH
- SHPATH
- OTCLIP and ITCLIP

DRAWPFF

DRAWPFF draws and inserts a PDF form field in a PDF document at the current secondary print position. PDF form fields are intended to be filled by a user (recipient) and the PDF sent back (submitted) to a process designed to extract the field contents and store them in a database.

The command is really effective when the VIPP® job is rendered into a PDF document. When the job is simply imaged on screen or on paper, only the initial image of the field as displayed when the PDF document is opened is drawn.

DRAWPFF can create various different types of form fields as detailed in the syntax. Form fields are included in the PDF using the AcroForm specifications. For more details, refer to Adobe Interactive Forms documentation.

Syntax

```
/type width height align [options] DRAWPFF
```

Where

/type	/TL - Text field (single line)
	/TB - Text box (multi lines)
	/CB - Check box
	/RB - Radio button
	/PL - Pull down list
	/CL - Combo box (free typing allowed)
	/LB - List box
	/RS - Reset form button
	/PR - Print form button
	/SF - Submit form button
	/SI - Digital Signature
width	is the width of the field rectangle in current units.
height	is the height of the field rectangle in current units.

align Indicates which point of the field can be aligned on the secondary print position

0 — top left (default)

1 — top right

2 — top center

10 — bottom left

11 — bottom right

12 — bottom center

20 — center left

21 — center right

22 — center center

options is an array with the following optional key/value pairs:

/FName Field name

Type string

Default value none

Description A name to identify the field at extraction time. Must be different for each field except for a set of radio buttons which must share the same name

/FDesc Field alternate description

Type string

Default value none

Description An alternate field name to be used in place of the actual field name wherever the field can be identified in the user interface (such as in error or status messages referring to the field). This text is also useful when extracting the contents of document in support of accessibility to users with disabilities or for other purposes.

/FMap Field mapping name

Type string

Default value none

Description The mapping name to be used when exporting PDF form field data

/ReadOnly Read only flag

Type integer

Default value 0

Description 0 - field can be changed

1 - field cannot be changed

/Required Required flag

Type integer

Default value 0

Description 0 - field can be left empty

1 - field cannot be left empty

/NoExport Export flag

Type integer

Default value 0

Description 0 - field will be exported

1 - field will not be exported

/NoPrint Print flag

Type integer

Default value 0

Description 0 - field can be printed

1 - field cannot be printed

/MulSel Multiple selection flag (LB only)

Type integer

Default value 0

Description 0 - only one selection allowed

1 - multiple selection allowed

/FValue Field value (TL, TB, PL, CL, LB only)

Type string

Default value none

Description Preset value displayed when the PDF is opened.

/DValue Default field value

Type string

Default value none

Description Value displayed when pressing the reset button.

/IState Preset state for check boxes and radio buttons (must be coded only with the first radio button in a set).

Type integer

Default value 0

Description 0 - off state

1 - on state for check box or first radio button

>1 - on state for other radio buttons

(button order number)

/FChoices Array of value choices (PL, CL, LB only)

Type Array of strings OR array of [export display] strings

Default value none

Description List of choices presented to the user. When [export display] format is used the export strings are used for export and the display strings are used for presentation on screen.

/TAlign Field text alignment in the field

Type integer

Default value 0

Description 0 - left

1 - right

2 - center

/BColor Field border color

Type color key (Gray, RGB and CMYK color only)

Default value none (transparent)

Description Color of the field border

/FColor Field background color

Type color key (Gray, RGB, CMYK, Gradient and pattern color only)

Default value none (transparent)

Description Color of the field border

/VColor Field value color

Type Color key (Gray, RGB and CMYK color only)

Default value none (transparent)

Description Color of the field value

/BStyle Field border style

Type [width /style]

Default value [1 /s]

Description Border style:

- border width in points
- border style is one of

/S solid

/D dashed

/B embossed

/I engraved

/U underline

/FCaption Field caption (CB, RB, RS and SF only)

Type string

Default value check sign (CB) or bullet (RB)

Description For check box and radio button:

<hex code> (see caption table below) for reset and submit buttons: (text string)

/TSplit Split text field (TL, TB only)

Type integer

Default value 0

Description 0 - no split

>0 - number of splits and max length

/To Submission destination (SF only)

Type URL string

Default value none

Description Email or web server URL where the form extraction (in FDF or PDF format) is sent.

/SubmitPDF Submission format (SF only)

Type integer

Default value 0

Description 0 - submit as FDF format

1 - submit as PDF format

2- submit as HTML format

3- submit as XFDF format

/TFName A font name for field value. Limited to the following list: /Helv /HeBo /HeOb /HeBO(Helvetica) /TiRo /TiBo /Tilt /TiBI (Times-Roman) /Cour /CoBo /CoOb /CoBO (Courier).

Type /name

Default value /Helv

/TFSize A font size for field value

Type integer (points)

Default value 0

Description 0 - auto-scalable

>0 - fixed font size

/FImage A field background image

Type String (image name of a TIFF, JPEG or EPS)

Default value none

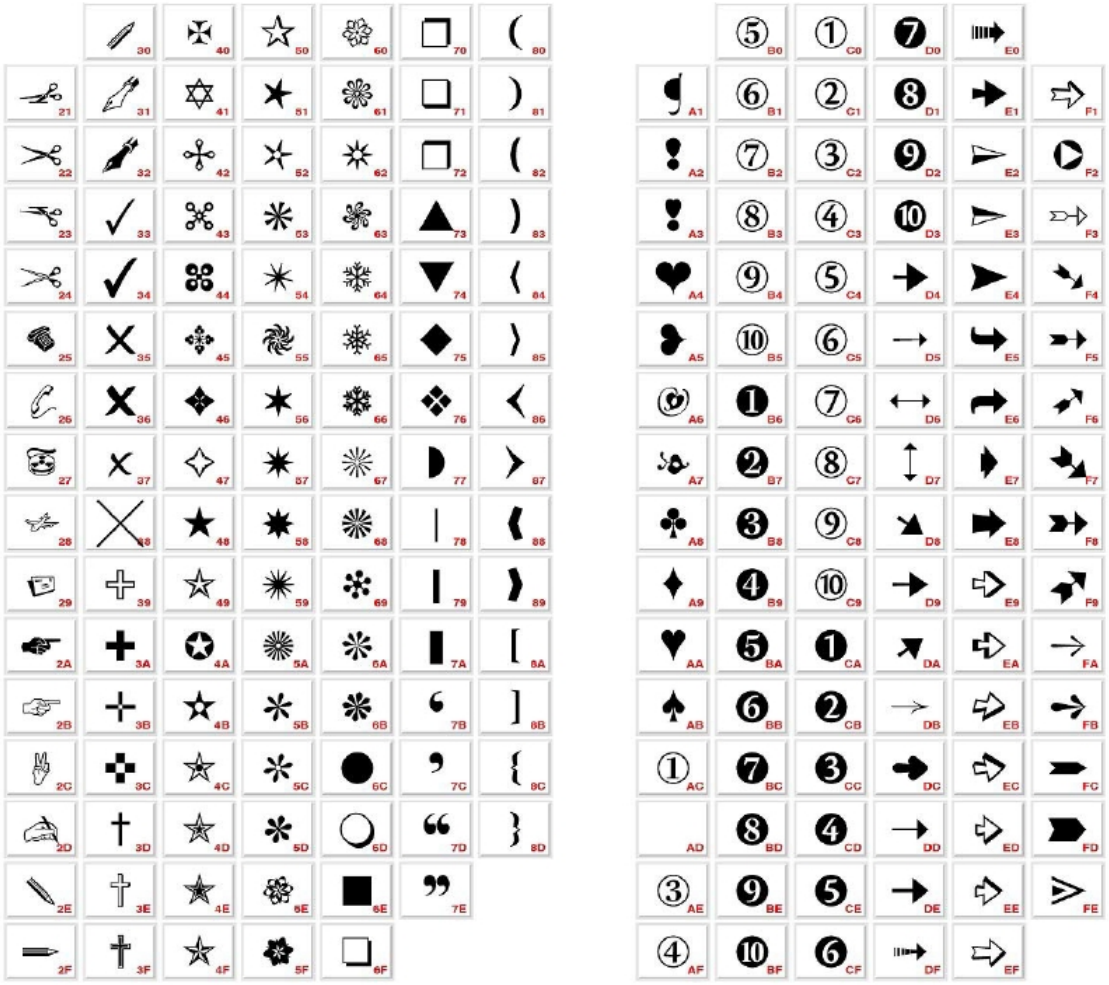
Description Image to be used as field background (alternative to FColor)

Examples

```
% text box
/TL 600 100 0
[ /FName (FirstName)
  /TAlign 2
  /FValue (John)
  /BColor ORANGE
  /BStyle [1 /B]
] DRAWPFF

% check box:
/CB 100 100 0
[ /FName (CheckBox1)
  /BStyle [1 /B]
  /FColor XLGREEN
  /VColor ORANGE
  /IState 1
] DRAWPFF
```

Caption table for FCaption option:



Modes

This command is applicable in all modes.

Related commands

SETPIF

DRAWPIE

DRAWPIE draws a pie chart. This command places the center of a pie chart at the current secondary print position.

Syntax

```
[ label/value list ] radius DRAWPIE
[ label/value list ] radius option DRAWPIE
ddg_index radius DRAWPIE
ddg_index radius option DRAWPIE
```

Where:

label/value list	is a list of label/value pairs.
radius	is the radius of the pie chart. When /FitSpace > 0, then radius is the maximum acceptable radius.
option	is detailed in Parameter Descriptions. All of the parameters have default values and can be omitted. All parameters set here will temporarily override the default value set by **SETPARAMS** and only apply to that command. The default values can be restored for subsequent commands.
ddg_index	refers to a list of label/values captured by an RPE entry (only zero is currently supported). Refer to FROMLINE align parameter.

Examples

```
[ (FF) 100 (US$) 250 (DM) 150 ] 200 DRAWPIE
[ (FF) 100 (US$) 250 (DM) 150 ] 200 [/3D true /3DThickness .8
/ColorTable [BLUE GREEN RED] ] DRAWPIE
[ (FF) 100 (US$) 250 (DM) 150 ] 200 /29 DRAWPIE
```

Modes

This command is applicable in all modes.

Related commands

- SETPARAMS
- FROMLINE
- RPEKEY
- DRAWBAR
- DRAWCRV

DRAWPOL

DRAWPOL draws a polygon. This command supports strings as operands, allowing you to use DBM variables as operands.

Syntax

```
[ Xpos1 Ypos1 Xpos2 Ypos2 ... Xposn Yposn ] GEPkey DRAWPOL
```

Where:

Xpos and Ypos	provide the subsequent coordinates of a polygon, which is outlined, filled, or both according to the GEPkey.

Examples

```
[ 200 200 1240 3300 2280 200 ] LMED_S1 DRAWPOL % draws a triangle
```

Modes

This command is applicable in all modes.

Related commands

- DRAWB and DRAWBR
- TPATH
- SHPATH
- OTCLIP and ITCLIP
- DRAWPATH and DRAWPATHR

DRAWRDR

DRAWRDR is a chart command used to draw radar charts.

Syntax

```
[ lab1 val1 lab2 val2 … labN valN ] radius DRAWRDR
[ lab1 val1 lab2 val2 … labN valN ] radius option DRAWRDR
[ lab1 [val11 val12 ..val 1M] … labN [valN1 valN2 ..val NM] ] radius DRAWRDR
[ lab1 [val11 val12 ..val 1M] … labN [valN1 valN2 ..val NM] ] radius option DRAWRDR
```

Where:

labX

is the string label for a radar item. Variables can be used. Labels do not word wrap.

valueX

is the value, string or real, for a radar item. Variables can be used. Refer to the /Format parameter for formatting options.

[valX1 valX2 ..valXM]

is the set of string values, string or real, for multiple radar items. Variables can be used.
When a set of values is provided, equivalent points are drawn along the same axis. The number of values must be the same in all sets.
The number of colors in the /ColorTable parameter must match the number of values in a set. Spot labelling for these colors can be provided using the /SpotLabels parameter. Refer to the /Format parameter for formatting options.

Examples

```
[ (ItemA)  20
  (ItemB)  11
  (ItemC)  42
  (ItemD)  35
  (ItemE)  95
  (ItemF)  63
}  800 DRAWRDR
```

The following default parameters are forced for radar charts:

- `/BGLineColor DARK`
- `/ColorTable [ RED ]`
- `/PlotSymbol [(u) current_size+4 null]`

Modes

This command is applicable in all modes.

Related commands

- DRAWBARDRAWPIE
- DRAWCRV

- DRAWPAR

DUPLEX_off

DUPLEX_off disables duplex printing. This is the default.

To delay switching from duplex to simplex, and avoid throughput deterioration in jobs that frequently switch from duplex to simplex, use **SETPARAMS** to set the parameter MixPlexCount using these values:

- When in mix-plex mode and the MixPlexCount value is greater than 0, the value is the number of pages that can be printed with blank back pages after the execution of the **DUPLEX_off** command. True simplex mode can only be entered after that number of pages.
- When in mix-plex mode and the MixPlexCount value is equal to 0, true simplex mode is entered immediately after the execution of **DUPLEX_off**. This is the default value for FreeFlow Print Server printers.
- When the MixPlexCount value is equal to -1, true simplex mode is never entered in mix-plex mode. For backward compatibility, this is the default value for NPS printers. When this value is set, the print speed for the document remains the same as in duplex mode because a blank page is imaged on the back of every simplex page.

Syntax

```
DUPLEX_off
```

Modes

This command is applicable in all modes.

Related commands

- BCALL
- DUPLEX_on
- ENDIMP
- TUMBLEDUPLEX_off
- TUMBLEDUPLEX_on

DUPLEX_on

DUPLEX_on enables duplex printing. The default is **DUPLEX_off**.

Syntax

```
DUPLEX_on
```

Modes

This command is applicable in all modes.

Related commands

- BCALL
- DUPLEX_off
- ENDIMP
- TUMBLEDUPLEX_off
- TUMBLEDUPLEX_on

ENDARBM

Use the **ENDARBM** command to end an bi-directional merge definition, **BEGINARBM**.

Syntax

```
ENDARM
```

Modes

This command is applicable in all modes.

Related commands

FCALL, MOVETO, SCALL

ENDARBT

Use the **ENDARBT** command to end an bi-directional context definition (BEGINARBT).

Syntax

```
ENDARBT
```

Modes

This command is applicable in all modes.

Related commands

BEGINARBT

ENDBOOKLET

Use the **ENDBOOKLET** command to end a booklet. You can code this command after the **PAGEBRK** command on the last page of a booklet.

Syntax

```
ENDBOOKLET
```

Modes

This command is applicable in all modes.

Related commands

SETPARAMS, STARTBOOKLET

ENDCASE

ENDCASE can close the CASE list. Use this command only with the **CASE** command.

Syntax

```
ENDCASE
```

Related commands

CASE

ENDCLIP

ENDCLIP cancels the clipping area defined by a previous command using a CLIP GEPkey.

Syntax

```
ENDCLIP
```

Modes

This command is applicable in all modes.

Related commands

COLLATE_dbm

ENDIFALL

ENDIFALL is an **RPE** sub-command used when conditions are nested at any level. /ENDIFALL provides a facility to close all pending /IF statements in one command rather than having to code all matching /ENDIF statements. Refer to SETRCD and SETPCD. For more information, also refer to RPE Command Information and to other related RPE commands.

Examples

```
1 FROMLINE
        /IF_CND1
           [ ....  rpe entry 1  .... ]
           [ ....  rpe entry 2  .... ]
           /IF_CND2
           [ ....  rpe entry 3  .... ]
           /ELSE
               [ ....  rpe entry 4  .... ]
           /ENDIF
        /ELSE
           [ ....  rpe entry 5  .... ]
           [ ....  rpe entry 6  .... ]
           /IF_CND3
                ....  rpe entry 7  .... ]
           /ENDIF
        /ENDIF

10 FROMLINE
        /IF_CND4
        [ ....  rpe entry 11  .... ]
           [ ....  rpe entry 12  .... ]
        /ELSE /IF_CND5
           [ ....  rpe entry 13  .... ]
        /ELSE /IF_CND6
           [ ....  rpe entry 14  .... ]
        /ELSE /IF_CND7
           [ ....  rpe entry 15  .... ]
           [ ....  rpe entry 16  .... ]
        /ELSE
           [ ....  rpe entry 17  .... ]
        /ENDIFALL
```

Modes

This command is applicable in line mode.

Related commands

None

ENDIMP

ENDIMP turns off the imposition feature initiated by **BEGINIMP**. These 2 commands can always be coded as a pair.

Syntax

```
BEGINIMP
call to document tiff files
ENDIMP
```

Examples

```
TWOUP
TUMBLEDUPLEX_on
ENDIMP
BEGINIMP (report1.ps) RUNDD
```

Modes

This command is applicable in all modes except line mode.

Related commands

BEGINIMP

ENDJOB

Use **ENDJOB** to define a sequence of actions to execute at the end of a job.

Syntax

```
{additional actions} ENDJOB
```

Where:

{additional actions}	is a sequence of VIPP® commands that can be executed at the end of the job. Additional actions can include additional marking on the last page or additional pages depending upon if and how **PAGEBRK** is used in the sequence.

Examples

```
{ PAGEBRK
  /NHE 30 SETFONT
  200 3000 MOVETO
  (This job printed $$PPCOUNT. pages) VSUB 0 SHP
  PAGEBRK
} ENDJOB
```

Modes

This command is applicable in line mode and database mode.

Related commands

STARTDBM, STARTLM

ENDOFRUN

ENDOFRUN acts as a subset of a set delimiter. When inserted in the print file, this command can be placed at the beginning of the last page of the subset.

Syntax

```
ENDOFRUN
```

Code **ENDOFRUN** in the data stream (in an NMP in line mode,) or in a **BEGINPAGE** procedure based on any condition detected on the page as illustrated in this example.

```
/LASTPAGE 1 60 0 --length-- /eq (endofrun_marker) SETPCD
{IF LASTPAGE { ENDOFRUN } ENDIF } BEGINPAGE
```

This allows you to place the `endofrun_marker` at any location in the last page of the set. **SETPBRK** can be used to set up a page delimiter that combines an `endofpage_marker` with an `endofrun_marker`. When the form feed is used as an `endofpage_marker`, combine the above setting with this example.

```
<0C> 021 SETPBRK
```

The data stream displays as follows:

```
%!
(xyz.jdt)STARTLM
            first page
<FF>
            2nd page
<FF>
            last page of run
<FF>endofrun_marker
            first page of new run
```

In this example, the `endofrun_marker` is part of the last page of the run. This may help when migrating LCDS applications.

Note: This feature is effective only on DocuPrint NPS systems with the MultiSet feature enabled and on FreeFlow Print Server. For a specific action on the subset, code the **SETFINISHING** commands in the JDT or at the beginning of the job. For FreeFlow Print Server printers, the **ENDOFSET** and **ENDOFRUN** commands are equivalent. There is no differentiation between a set and a run.

Modes

This command is applicable in all modes.

Related commands

BEGINPAGE,SETPCD,SETFINISHING

ENDOFSET

ENDOFSET acts as a set delimiter. When inserted in the print file, this command can be placed at the beginning of the last page of the set.

Note: This command can be used on a FreeFlow Print Server (FFPS) without using the **SETFINISHING** command. To set finishing at the FFPS, use a queue with the Subset Output option. Set the Subset Output option to `Retrieved from PDL`.

Syntax

```
ENDOFSET
```

Code **ENDOFSET** in the data stream (in an NMP in line mode), or in a **BEGINPAGE** procedure, based on any condition detected on the page as follows.

```
/LASTPAGE 1 60 0 –length– /eq (endofset_marker) SETPCD

{IF LASTPAGE { ENDOFSET } ENDIF } BEGINPAGE
```

This allows you to place the endofset_marker at any location in the last page of the set. **SETPBRK** helps to set up a page delimiter that combines an `endofpage_marker` with an `endofset_marker`.

When the form feed is used as an `endofpage_marker`, combine the above setting with this example.

```
<0C> 021 SETPBRK
```

The data stream displays as follows:

```
%!
(xyz.jdt)STARTLM
.......... first page
<FF>
.......... 2nd page
<FF>
.......... last page of set
<FF>endofset_marker
.......... first page of new set
```

In this example the `endofset_marker` is part of the last page of the set. This may help migrating LCDS applications.

Note: This feature is effective only on DocuPrint NPS systems with the MultiSet feature enabled, and on FreeFlow Print Server. For a specific action on the subset, code /ON **SETFINISHING** commands in the JDT or at the beginning of the job. For FreeFlow Print Server printers, the **ENDOFSET** and **ENDOFRUN** commands are equivalent. There is no differentiation between a set and a run.

Modes

This command is applicable in all modes.

Related commands

- BEGINPAGE
- SETPCD
- SETFINISHING,

- STARTOFSET

ENDPAGE

Use **ENDPAGE** to define actions that the system performs at the end of each page. The default is no actions.

Syntax

```
{end of page actions} ENDPAGE
{end of page actions} count ENDPAGE
{end of page actions} /P ENDPAGE
```

Where

count	is an optional operand defining the number of consecutive pages on which the actions can be executed. The default for the count operand is 999999.

The first two syntax examples define actions that the system executes before the page is imaged. In general, these consist of final markings, such as printing data that was previously captured using **GETFIELD**.

The third syntax example defines actions that the system executes after the page is imaged. In general, use this syntax to change the setup for subsequent pages, depending on the conditions evaluated by **SETPCD**. Marking commands cannot be used in this syntax.

Examples

```
{ /NHE 20 SETFONT
    100 3000 MOVETO
    VAR.CUST SHC
} ENDPAGE

{ IF BANNER
        { /VARjdt 0 5 8 /BANNER GETFIELD
        ($$VARjdt...jdt) VSUB SETJDT
        }
    ENDIF
} /P ENDPAGE
```

Modes

This command is applicable in all modes.

Related commands

- BEGINPAGE
- SETPCD
- GETFIELD
- SLIPSHEET

ENDPCC

ENDPCC ends a PCC definition. It can be coupled with **BEGINPCC**.

Syntax

```
ENDPCC
```

Modes

This command is applicable in line mode.

Related commands

BEGINPCC, SETPCC, SETVFU

ENDRPE

ENDRPE ends an RPE library definition. It can be coupled with **BEGINRPE**. Refer to RPE Command Information and to other related RPE commands.

Syntax

```
ENDRPE
```

Modes

This command is applicable in line mode.

Related commands

- BEGINRPE
- FROMLINE
- INDEXRPE
- RPEKEY

ENDTABLE

Use **ENDTABLE** to terminate a table.

Syntax

```
ENDTABLE
```

Examples

```
[ /Margins [10 10 10 10] /TableStroke B_S1 ] BEGINTABLE
[ [ /Width 200 /CellText (R1/C1 Hello World) /CellFill RED ]
[ /Width 300 /CellText (R1/C2 Hello Earth) /CellFill YELLOW]
[ /Width  80 /CellText (R1/C3 Hello Man)   /CellFill GREEN]
] SHROW
[ [ /Width 200 /CellText (R2/C1 Hello World) /CellFill RED ]
[ /Width 300 /CellText (R2/C2 Hello Earth) /CellFill YELLOW]
[ /Width  80 /CellText (R2/C3 Hello Man)   /CellFill GREEN]
] SHROW
ENDTABLE
```

Modes

This command is applicable in all modes.

Related commands

- ENDRPE
- FROMLINE
- INDEXRPE
- RPEKEY
- STARTLM
- SHROW
- BEGINTABLE

ENDXPD

ENDXPD ends an XML Processing Definition (XPD) table in an XML Job Ticket (XJT) file. It can be coupled with an **BEGINXPD** command.

Syntax

```
ENDXPD
```

Modes

This command is applicable in XML mode.

Related commands

- BEGINXPD
- BTA
- BTS
- ETA
- ETS
- STARTXML

ETA

ETA ends a tag action definition. It can be coupled with an **BTA** command.

Syntax

```
ETA
```

Modes

This command is applicable in line mode.

Related commands

- BEGINXPD
- BTA
- BTS
- ETS
- STARTXML

ETCLIP

Use **ETCLIP** to clear a path previously set by the **TPATH**, **OTCLIP** or **ITCLIP** GEPkeys.

Syntax

```
ETCLIP
```

Modes

This command is applicable in all modes.

Related commands

TPATH, OTCLIP and ITCLIP

ETS

ETS ends a tag substitution definition. It can be coupled with an **BTS** command.

Syntax

```
ETS
```

Modes

This command is applicable in XML mode.

Related commands

- BEGINXPD
- ENDXPD
- BTA
- BTS
- ETA
- STARTXML

EXIST

The **EXIST** command tests for the existence of an external or embedded VIPP® resource. The **EXIST** command requires use with the **IF** command.

Syntax

```
IF (resource_name) /rec_type EXIST
```

Where:

resource_name	Is the name of the resource that is checked. If resource_name is an empty string, the **EXIST** command returns `false`.
rec_type	Is one of the following: **frm**: VIPP® form files FRMs and EPS files in `xgfc/formlib`. **dbm**: VIPP® Data Base Master Files (DBMs) in `xgfc/formlib`. **seg**: VIPP® PostScript Segments in `xgfc/formlib`. **img**: TIFF and JPEG files in `xgfc/formlib` **jdt**: VIPP® Job Description Ticket files in `xgfc/formlib`. **enc**: Font-encoding tables in `xgfc/encoding`. **fnt**: Font files. **mis**: Miscellaneous files. The path is set by the **SETMPATH** command. **dd**: Decomposition files.

Note: The paths for all files are defined in the `xgf/src/xgfunix.run` or `xgf/src/xgdos.run`.

Examples

```
IF (truk.tif) /img EXIST
{ 100 200 MOVETO (truk.tif) 1 0 ICALL }
ENDIF
```

In the following example, when the condition is `true`, the truk.tif image is placed. Otherwise, a default.tif image is placed.

```
IF (truk.tif) /img EXIST
{ 100 200 MOVETO (truk.tif) 1 0 ICALL
}
ELSE
{ 100 200 MOVETO (default.tif) 1 0 ICALL
}
ENDIF
```

Modes

This command is applicable in all modes.

Related commands

- ICALL
- SETFORM

- SCALL
- SETJDT
- SETENCODING
- RUN
- RUNDD

EXIT

Use **EXIT** to exit from a REPEAT loop.

Syntax

```
EXIT
```

Examples

This example can print all Decomposition Services documents named in order `docu001.txt`, `docu002.txt`, and so on and exit on the first named file that does not exist, up to 999 documents.

```
{ /VAR1 RPCOUNT (###) FORMAT SETVAR
    IF (docu$$VAR1..txt) VSUB /dd EXIST
        {  BEGINIMP
            (docu$$VAR1..txt) RUNDD
            ENDIMP
        }
    ELSE
        {EXIT}
    ENDIF
} 999 REPEAT
```

Modes

This command is applicable in all modes.

Related commands

REPEAT, EXIST, IF/ELSE/ELIF/ENDIF

FBIND

FBIND is used exclusively before **XGFRESDEF** when embedding a VIPP® form in a VIPP® job.

Syntax

```
{ form contents } FBIND XGFRESDEF
```

Modes

This command is applicable in all modes.

Related commands

XGFRESDEF

FCALL

FCALL executes a segment in the current context such as font, print, or position.

Syntax

```
(Segmentname) FCALL
```

Segments must be coded in VIPP® native mode and stored in one of the formlib libraries defined by **SETFPATH** in the `/usr/xgf/src/xgf` file. Use of the `.seg` extension is recommended.

Note: Unlike the **SCALL** command, the **FCALL** command is not encapsulated. When you replace the Segmentname parameter in **FCALL** with the contents of the segment, you get the same result. Be aware of possible side effects. Inefficient PostScript code can affect performance.

The **FCALL** command can execute a **PAGEBRK** as part of the segment definition, without affecting performance.

Tip: **FCALL** can be used to store pieces of frequently used VIPP® code or to print paragraphs down the page.

Modes

This command is applicable in all modes.

Related commands

SCALL, SETFPATH

FILLOMR

FILLOMR fills in the bubbles and response boxes of an Optical Mark Reading (OMR) grid.

Syntax

```
(response string) FILLOMR
(response string) [ parameters ] FILLOMR
```

Where:

[parameters]	can be one of the parameters listed here: **/OMRMap (map string)** **/OMRDir /H or /V** **/OMRHskip integer** **/OMRVskip integer** **/OMRHdisp integer** **/OMRVdisp integer** **/OMRSlugFont /Fontname** **/OMRSlugSize integer** **/OMRSlugChar string** **/OMRWriteResp boolean** **/OMRMode integer**

Parameters are described in detail in Parameter Descriptions.

All the parameters have built-in defaults that can be altered using **SETPARAMS**. The FILLOMR array can be provided with only those parameters that differ from the defaults.

The response string can be made up of characters from the `OMRMap` string. The current font set by **SETFONT** or a font index is used for printing the response string.

The current font set by **SETFONT** or a font index is used for printing the response string.

Examples

```
/Helvetica 15 SETFONT                % Font for response string
100 1000 MOVETO                      % The origin of the top left corner of the grid.
                                     % More specifically, it is the upper left corner
                                     % of the grid cell (OMRHdisp X OMRVdisp) in
                                     % which the first grid bubble is centered.

(398400562874234)  [ /OMRMap (0123456789)
            /OMRDir /H
            /OMRHskip 1
            /OMRVskip 2
            /OMRHdisp 6
            /OMRVdisp 5
            /OMRSlugFont /XOMR
            /OMRSlugSize 12
            /OMRSlugChar (A)
            /OMRWriteResp true
  ] FILLOMR
100 2000 MOVETO

(4523456) [ /OMRMode 1 /OMRMap (8421) ] FILLOMR
(234523456) [ /OMRMode 2 ] FILLOMR
```

Modes

This command is applicable in all modes.

Related commands

SETPARAMS, MOVETO, SETFONT

FOREACH

FOREACH applies **GETITEM** to each entry of a variable array and executes the supplied procedure. The variable array is previously defined and populated with **SETVAR** and **ADD**.

Syntax:

```
{proc } [ table ] FOREACH
```

```
{proc } [ table ] /MF FOREACH
```

Where:

{ sequence of VIPP commands }	is any sequence of native mode commands that produces a portion of a page, a complete page, or several pages. Variable names defined in the variable array can be used in this sequence.
variable_array_name	is a name referencing a variable array in the format suited for the **GETITEM** command.
/MF	When the /MF option is used, **SETLKF**, **FOREACH** and **SHROW** can interact with each other when **SHROW** is called in the **FOREACH** loop. The total size of all **SHROW** called in the **FOREACH** procedure is evaluated. If it does not fit in the frame, the rows are not imaged, a **NEWFRAME** is called, and the procedure is executed a second time with the same table entry values. This option is intended to address multiple frames with different layouts, so that on the second execution a different **SHROW** is possibly called.

Note: In a VIPP® native mode job, place a **FOREACH** loop and a **PAGEBRK** command in a **BCALL** statement or in a processed linked-frame mode.

```
{ x y MOVETO
  { ... PAGEBRK ... } table_name FOREACH
} BCALL
```

```
[[ x y w h 0 ]] SETLKF
{ ... } table_name FOREACH
```

Examples

These examples will print:

```
John Smith
Paul Martin
Mary White
```

```
/VAR_names
[ [ /VAR_fname /VAR_name ]
[ (John) (Smith) ]
[ (Paul) (Martin)]
[ (Mary) (White) ]
] SETVAR

{ ($$VAR_fname. $$VAR_name.) VSUB SHL } VAR_names FOREACH
```

Modes

This command is applicable in all modes.

Related commands

- GETITEM
- SETVAR
- ADD
- REPEAT
- UPDATE
- SORT

FORMSHIFT

Use **FORMSHIFT** to adjust the origin in a VIPP® form.

Syntax

```
X Y FORMSHIFT
```

Where:

X	is the horizontal adjustment in current units. It can be negative.
Y	is the vertical adjustment in current units. It can be negative.

Note: **FORMSHIFT** is in effect only inside a form definition, between the braces {}, and after the orientation command, if any.

Modes

This command is applicable in all modes.

Related commands

SHIFT

FROMLINE

FROMLINE sets an RPE definition to be applied from linenumber. For more information on a discussion of how to extend both the **FROMLINE** and **RPEKEY** commands using conditional processing, fixed text, and the align procedure, and to other related RPE commands, refer to RPE Items.

Syntax

```
linenumber FROMLINE
[align rotate Xinit Xdispl Yinit Ydispl recpos length
  /font Colorkey]

[align rotate Xinit Xdispl Yinit Ydispl field_nr /FN
  /font Colorkey]

[ {proc} rotate Xinit Xdispl Yinit Ydispl recpos length
  /font Colorkey]

[ {proc} rotate Xinit Xdispl Yinit Ydispl field_nr /FN
  /font Colorkey]

[ {proc} rotate Xinit Xdispl Yinit Ydispl 0 (resource_name)
  /font Colorkey]

[ align rotate Xinit Xdispl Yinit Ydispl 0 (my text here)
  /font Colorkey]
```

Where:

linenumber	refers to the record position in the print file from the last page delimiter. The **FROMLINE** command is then followed by one or several RPE entries encapsulated between square brackets [ ].

Each RPE entry has ten parameters that define the field processed in the record and the layout attributes, as follows:

align can include:

0 align left

1 align right

2 align center

[Colwidth 3] justify on Colwidth

4 align left with heading blanks strip

5 align right with trailing blanks strip

[(string) 6] align on string (for example, decimal point)

7 show on (append to previous string).

When the align parameter is extended with a decimal, the related RPE field can be captured for later use in a Data Driven Graphic (DDG). For more information, refer to DRAWPIE,DRAWBAR, or DRAWCRV. When extending align with a decimal:

x.1 - capture a value field (mandatory)

x.2 - capture a label field (optional)

For example, 1.1 can capture a value field and print it aligned right, [(.) 6.2] cancapture a label field and align it on the decimal point.

The align parameter can also be replaced by a procedure body {proc}. Refer to Align procedure in Programming Tips.

rotate is the rotation angle in degrees (positive is counterclockwise).

Xinit is the initial horizontal position on the page computed from the left margin.

Xdispl is added to Xinit for consecutive records using the same RPE definition.

Yinit is the initial vertical position on the page computed from the top margin. In an RPE definition, the origin is always in the top left corner of the page. This parameter can also be expressed as a variable, /YINIT.

Ydispl is added to Yinit for consecutive records using the same RPE definition. This parameter can also be expressed as a variable, /LSP.

recpos is the record position of the field to select (starting with 0).

length is the length of the field. length can be replaced with a string value.

field_nr is the field number starting with 0. It applies to records with a field delimited structure. /FN indicates that the previous parameter is a field number. The field delimiter is defined by **SETDBSEP**. The default is :.

font is the font index as defined by **INDEXFONT**. /CRFT is a reserved index name that can be used to refer to the font used in the previous line. It gives an initial value using **INDEXFONT**.

Colorkey is defined in the `/usr/xgf/src/xgf.gep` file, for more information, refer to Standard lists, tables, keys, and attributes in the *FreeFlow VI Compose User Guide*.

resource_name is the name of the resource to be used with an Align procedure, refer to Programming Tips.

my text here is the Fixed text to be printed, refer to Programming Tips.

Examples

In this example, the value Smith is obtained by this RPE entry from the record that follows.

```
/ADR0
[ 0 0 100 null 200 50 3 /FN /F1 BLACK ]
ADR0:Robert:W.:Smith:New York
```

This example shows how to capture data (CASH/32.70 and SHARE/45.28) printed with a FROMLINE definition and use the data to create a data driven graphic.

```
Data File
CASH          37,586            32.70
SHARE         26,879          45.28
              -----

/GraphData   14  5  /eq  (-----) SETRCD
2 BEGINRPE
1 FROMLINE
[ 0.2  0  150 null  150  50  0  10  /F4 BLACK ]
[ 0.1  0  600 null    0   0 24   5  /F4 BLACK ]

/GraphData
[ {SCALL} 0 1500 null 300 50 0 (PIE0) /F3 BLACK ]
/ENDIF
ENDRPE

/PIE0 {  0  300  DRAWPIE } XGFRESDEF
```

Using FROMLINE to Compute Print Position

The print position for consecutive lines formatted by the same RPE definition, or RPE group when RPEKEY is used, is computed using one of these methods:

- Fixed line spacing
- Variable line spacing

Fixed line spacing provides backward compatibility. When the null keyword is not used in any Xdispl or Ydispl field in the RPE definition, the print position is computed as follows:

Xpos = Xinit+Xdispl*LN

Ypos = Yinit+Ydispl*LN

Where:

LN	is the number, starting from zero, of the line in the current group. When the group changes, LN is reset to zero
Xpos and Ypos	are re-initialized with the Xinit and Yinit values of the new group.
Xinit, Yinit, Xdispl, and Ydispl	generally can be identical in all entries of an RPE group. Xinit or Yinit can change only when the corresponding X or Y displacement is set to zero.

Examples

This is an example of fixed line spacing.

```
% RPE definition
5 BEGINRPE

% align rot. Xinit  Xdispl  Yinit  Ydispl  recpos  length  font    color

1   FROMLINE
    [   2   0       835     0     300     0       00      99      /F4     BLACK ]
2   FROMLINE
    [   2   0       615     0     445     0       00      99      /F1     WHITE ]
3   FROMLINE
    [   2   0      1199   318     445     0       00      99      /F1     WHITE ]
9   FROMLINE
    [   2   0      3140     0     445     0       00      99      /F5     WHITE ]
10  FROMLINE
    [   0   0       230     0     560    75       00      33      /F2     BLACK ]
    [   1   0      1345     0     560    75       33      12      /F2     BLACK ]
    [   1   0      1658     0     560    75       45      12      /F2     BLACK ]
    [   1   0      1976     0     560    75       57      12      /F2     BLACK ]
    [   1   0      3290     0     560    75      105      12      /F3     BLACK ]
    [   1   0      2286     0     560    75       69      12      /F2     BLACK ]
ENDRPE
```

With variable line spacing, Xdispl and Ydispl can vary in an RPE definition or RPE group. When processing lines match an RPE definition or RPE group, the print position (X and Y) resulting from the last RPE entry is kept in memory and is used as the initial print position for the next RPE entry. These values are initialized with Xinit and Yinit the first time an RPE definition or RPE group is involved in a page. In all subsequent access to any entry in this RPE definition, the print position is computed as follows:

- When the keyword null is used for either Xdispl or Ydispl, the corresponding Xinit or Yinit values of the current RPE entry are used.
- When the keyword null is not used for either Xdispl or Ydispl, the Xdispl or Ydispl values are added to the values kept in memory. When PCC processing is enabled, Xdispl or Ydispl values are added before or after printing the record portion, depending on the PCC definition.

Therefore, when a record is split into several fields that can be printed with the same horizontal position, Yinit and Ydispl are only relevant in the first RPE entry. All other entries can have these parameters set to zero. In addition, Xinit varies to reflect the different horizontal positions of the fields and Xdispl is always null.

This is an example of variable line spacing in which LFA1 records are using a font (F2) requiring a line spacing (Ydispl = 80) larger than the one (Ydispl = 50) required with the font (F1) used for LFA0 records.

```
%             Xinit  Xdispl  Yinit  Ydispl
/LFA0 RPEKEY
[   0   0     70    null    910     50       5       2      /F1     BLACK]
[   0   0    130    null      0      0       7      22      /F1     BLACK]
[   1   0    920    null      0      0      29       6      /F1     BLACK]
[   1   0   1170    null      0      0      35       9      /F1     BLACK]
[   1   0   1420    null      0      0      44       8      /F1     BLACK]
[   1   0   1620    null      0      0      52       6      /F1     BLACK]
[   1   0   1870    null      0      0      58       9      /F1     BLACK]
[   1   0   2100    null      0      0      67       7      /F1     BLACK]
[   1   0   2410    null      0      0      74      11      /F1     BLACK]
/LFA1 RPEKEY
[   0   0     70    null    910     80       5      24      /F2     BLACK]
[   1   0   2410    null      0      0      74      11      /F2     BLACK]
```

LFA0 and LFA1 records in the data file can be intermixed in any order, the line spacing adjusts automatically.

This example shows the use of the subcommand NEWPOS. Use NEWPOS when a new print position is necessary for a group of RPE entries to print the same field at different locations on the page,

```
%          Xinit  Xdispl  Yinit  Ydispl
/LFA0 RPEKEY
[  0  0  70    null  910  50  0  30  /F1  BLACK]
NEWPOS
[  0  0  1200  null  200  80  0  30  /F3  BLACK]
```

NEWPOS forces a new independent set of print position values to be kept in memory.

In addition, with this new syntax, records belonging to the same RPE group can be nonconsecutive in the data file. Old and new syntaxes are exclusive in a specified RPE definition set by an RPEKEY or FROMLINE command. However, RPE definitions with both syntaxes can be mixed in an RPE library composed of several RPEKEY or FROMLINE commands placed between BEGINRPE and ENDRPE.

Tip: TLGRID can be a useful tool when setting up RPE definitions.

Modes

This command is applicable in line mode.

Related commands

- BEGINRPE
- COPYRANGE
- ENDRPE
- INDEXRPE
- LSP
- RPEKEY
- SETPAT
- SETPCD
- SETRCD
- SETTXC

FSHOW

Use **FSHOW** to submit a for mas a normal print file, and to obtain a sample of the form either as printed output or on screen.

FSHOW enables automatically caching when the form is invoked by **SETFORM** or **SETBFORM**.

FSHOW takes FORMSHIFT into account when the form is submitted on its own.

Note: To invoke automatic caching of forms that do not contain variable elements, use the **FSHOW** command. When a form is cached with **FSHOW** and the form contains variable elements, only the first variable object is in the cache or used in subsequent calls.

Syntax

```
%!

{ form contents } FSHOW
```

Modes

This command is applicable in all modes.

Related commands

None

GETDATE

GETDATE sets or updates these date- and time-related VIPP® variables:

D_DWL	Day of week (long)	Sunday, Monday, Tuesday, and so on
D_DWS	Day of week (short)	Sun, Mon, Tue, and so on
D_DD	Day of month	1–31
D_MOL	Month (alpha long)	January, February, March, and so on
D_MOS	Month (alpha short)	Jan, Feb, Mar, and so on
D_MO	Month number	01–12
D_YY	Year (2 digits)	00–99
D_YYYY	Year (4 digits)	1970–9999
T_HH	Hours (24)	00–23
T_HH2	Hours (12)	1–12
T_MM	Minutes	00-59
T_SS	Seconds	00-59
T_AMPM	AM/PM	am, pm
T_TZN	TimeZone	PST, PDT
D_DOY	DayOfYear	1–366

Syntax

```
GETDATE
```

GETDATE is initially called in `xgf.def` and then implicitly called at each page initialization. In most cases, you need not to call **GETDATE** explicitly unless you want to force an immediate update of the date variables.

The date variables can be used in conjunction with **VSUB** to construct an appropriate time stamp. The following example will print `01/24/2003 16:16:56`:

```
($$D_MO./$$D_DD./$$D_YYYY. $$T_HH.:$$T_MM.:$$T_SS.) VSUB SH
```

The behavior and result of **GETDATE** are impacted by these parameters:

/TimeZone	set time zone, +- minutes from UTC (-480 = PST)
/DaylightSaving	set start, end times for Daylight Saving Time
/DaysLong	list of day names (long)
/DaysShort	list of day names (short)
/MonthsLong	list of month names (long)
/MonthsShort	list of month names (short)

/TimeZoneName	time zone names: Standard time, Daylight Saving time
/AmPm	AM/PM designations, for example: [(a.m.) (p.m.)]
/DefaultDate	date used when no file system is available
/DefinedDate	when present, this date overrides the system date

Note: Specify the hours in 24hour time for **DefaultDate** and **DefinedDate**.

For more details on these parameters, refer to Parameter Descriptions.

Default values for these parameters, except /DefinedDate, are defined in `xgf.def`. To localize the time- and date-related variables for the area, and to return values in a language other than English, you must edit the `xgf.def` file. The example shown contains values for installations in the western United States.

```
/TimeZone    -480                % Pacific Standard
/DaylightSaving [                % set start, end times for Daylight Saving Time
    [ 2003  60  96 120 299 120 ] % year adj. sday stime  endday  endtime
    [ 2004  60  95 120 305 120 ] % 2004 +1Hr day95 0200Hr day305 0200Hr (4  Apr - 26 Oct)
    [ 2005  60  93 120 303 120 ] % 2005 +1Hr day93 0200Hr day303 0200Hr (3  Apr - 30 Oct)
    [ 2006  60  92 120 302 120 ] % 2006 +1Hr day92 0200Hr day302 0200Hr (2  Apr - 29 Oct)
    [ 2007  60  70 120 308 120 ] % 2007 +1Hr day70 0200Hr day308 0200Hr (11 Mar - 4  Nov)
    [ 2008  60  69 120 307 120 ] % 2008 +1Hr day69 0200Hr day307 0200Hr (9  Mar - 2  Nov)
    [ 2009  60  67 120 305 120 ] % 2009 +1Hr day67 0200Hr day305 0200Hr (8  Mar - 1  Nov)
    [ 2010  60  73 120 311 120 ] % 2010 +1Hr day73 0200Hr day311 0200Hr (14 Mar - 7  Nov)
    [ 2011  60  72 120 311 120 ] % 2011 +1Hr day72 0200Hr day310 0200Hr (13 Mar - 6  Nov)
    [ 2012  60  71 120 311 120 ] % 2012 +1Hr day71 0200Hr day309 0200Hr (11 Mar - 4  Nov)
  ]
/DaysLong [(Sunday)(Monday)(Tuesday)(Wednesday)(Thursday)(Friday)(Saturday)]
/DaysShort    [(Sun)(Mon)(Tue)(Wed)(Thu)(Fri)(Sat)]
/MonthsLong   [(January)(February)(March)(April)(May)(June)
               (July)(August)(September)(October)(November)(December)]
/MonthsShort  [(Jan)(Feb)(Mar)(Apr)(May)(Jun)(Jul)(Aug)(Sep)(Oct)(Nov)(Dec)]
/TimeZoneName [(PST)(PDT)]
/AmPm         [(a.m.) (p.m.)]
/DefaultDate  [ 2003 1 1 00 00 00 0 ] % 2003 Jan 1 00:00:00 Std time
```

Modes

This command is applicable in all modes.

Related commands

SPOOLNAME, SETPARAMS, SHIFTDATE

GETFIELD

The **GETFIELD** command captures a record portion or a field of a specific line and assigns its value to the specified variable.

Note: When PCC bytes are used in **FROMLINE** or **RPEKEY** entries, where `recpos=0` refers to the first data byte that skips the PCC, column one of the original data file is not treated as user data. The original data file is used as the PCC index. However, this is not true for **GETFIELD**, **SETRCD**, and **SETPCD** commands, where `recpos=0` always refers to the first byte of the record, whether the **SETPCC** command is coded or not coded.

Syntax

```
/VARname line_nr recpos length GETFIELD
/VARname line_off recpos length /PCDkey GETFIELD
/VARname line_nr field_nr /FN GETFIELD
/VARname line_off field_nr /FN /PCDkey GETFIELD
```

Where:

line_nr	is the number of the line from which to capture data.
recpos	is the position of the record portion to capture.
length	is the length of the record portion to capture.
field_nr	is the field number to capture in the record.
PCDkey	identifies the line from which to capture data.
line_off	is an offset to the line identified by **PCDkey**. For example: **0** selects the line on which **PCDkey** is true **1** selects the following line **-1** selects the previous line

When the condition is false, or when `line_nr` or `line_off` is out of range, **GETFIELD** returns an empty string.

Note: **GETFIELD** strips left and right blanks from the extracted field before assigning the field to the variable.

Examples

This example illustrates how to capture the data field following **FORM=** on the banner page and use it to activate a JDT on the subsequent pages. (FORM=INV01 calls INV01.jdt)*./BANNER* must be true to assign */VARjdt* the value captured by **GETFIELD**.

```
/BANNER 10 9 0 5 /eq (FORM=) SETPCD
{ IF BANNER
  { /VARjdt 0 5 8 /BANNER GETFIELD
    ($$VARjdt..jdt) VSUB 1 SETJDT
  }
  ENDIF
} /B BEGINPAGE
```

Modes

This command is applicable in line mode.

Related commands

- BEGINPAGE
- ENDPAGE
- SETPCD
- SLIPSHEET

GETITEM

Use the **GETITEM** command to assign a particular set of values referred to by an index to a set of field names. Field names and values are stored in a table previously defined by **SETVAR**. The first entry of the table holds the field names, subsequent entries hold the sets of values. **GETITEM** is used in a Data Base Master to avoid a long list of **SETVARs**. Typically this command is used to assign multi-lingual variables depending on an index in a DBF field.

Syntax

```
VAR_itemtable index GETITEM
```

Where:

VAR_itemtable is a table previously defined with this syntax:

```
/VAR_itemtable
[   [ /VAR_name1 /VAR_name2 /VAR_name3 .../VAR_nameN ]
    [ (value11) (value12) (value13) .... (value1N) ]
    [ (value21) (value22) (value23) .... (value2N) ]
    .......
    [ (valueM1) (ValueM2) (valueM3) .... (valueMN) ]
] SETVAR
```

index is an integer in the range 1 to M (length of the table).

Examples

This example shows how to build and use a multi-lingual table for a multi-lingual mailing. In the table LCODE is the language code ranging from 1 to 4. GENDER is the gender ranging from 1 to 3.

```
/VAR_LANGUAGE
[ [ /VAR_G1 /VAR_G2 /VAR_G3 /VAR_LETTER ]
[(Dear Sir,)(Dear Madam,)(Dear Miss,)(letter_en.ps)]                    % 1. English
[(Querido Señor,)(Querida Señora,)(Querida Señorita,)(letter_fr.ps)]  % 2. Spanish
[(Cher Monsieur,)(Chère Madame,)(Chère Mademoiselle,)(letter_sp.ps)]  % 3. French
[(Sehr geehrter Herr,)(Sehr geehrte Frau,)(Sehr geehrtes Fraulein,)
   (letter_ge.ps)]                                                      % 4. German ]
/INI SETVAR
VAR_LANGUAGE LCODE GETITEM
VAR_LETTER CACHE SETFORM
x y MOVETO
(VAR_G$$GENDER.) VSUB2 SHL
```

Modes

This command is applicable in all modes.

Related commands

- SETVAR
- REPEAT
- IF/ELSE/ELIF/ENDIF
- CASE

GOTOFRAME

Use the **GOTOFRAME** command with a frame number to start placing all subsequent elements into the specified frame. The frame number can be greater than the current frame number and less than or equal to the maximum frame number, otherwise the request can be ignored.

Syntax

Where:

frame_number	is the frame number for the destination frame (starting with 1).

Examples

This example starts the placement of a paragraph of text in frame 2.

```
2 GOTOFRAME
(My paragraph of text) 0 SHP
```

Modes

This command is applicable in native and database mode.

Related commands

FRCOUNT,NEWFRAME,SETLKF

ICALL

ICALL images a 6.0 TIFF file or JFIF/JPEG file (baseline encoded format) at the current secondary print position. Bi-level or Grayscale TIFF images use the current color set by SETTXC.

Syntax

```
(imagename) scale rotation ICALL
() scale rotation ICALL
(imagename) scale rotation align ICALL
() scale rotation align ICALL
```

Where:

imagename	is the name of the TIFF or JPEG file to image.
scale	is the scaling factor (1 = 100 % or no scaling). A special value (2D3) is available to print 200 DPI scanned images at 300 DPI (a 66,666... % reduction) with the best performance. The default is no scaling.
rotation	is the rotation value in degrees (positive is counterclockwise).
align	indicates which point of the image can be aligned on the secondary print position using these values: **0** top left (default) **1** top right **2** top center **10** bottom left **11** bottom right **12** bottom center **20** center left **21** center right **22** center center

Images can be stored in one of the libraries defined by **SETIPATH**. Use of the .tif or .jpg extension is recommended.

This support is provided for TIFF or JPEG files:

- One image for each file. Only the first image is processed
- Bi-level, transparent imaging, uncompressed or compressed with CCITT G4/G3 or PackBits encoding schemes
- Grayscale or full color, uncompressed or compressed with LZW encoding schemes, opaque imaging
- Two-dimensional (2D)
- Mono-strip or multi-strip images
- FillOrder (tag 266) value can be 1 (MSB). A FillOrder value of 2 (LSB) is supported on Docuprint NPS printers
- Support for CIE color space (`PhotometricInterpretation=8`).

- Support for density `parameters=0` in JPEG files
- Support for resolution available in the Exif marker of certain JPEG files
- JPEG baseline is supported, progressive encoding is not supported

Important: The JPEG/Exif marker contains information about the resolution that was ignored until FreeFlow version 4.0. In previous versions VIPP® may have defaulted to 300 dpi and possibly rendered the image at an incorrect size. Since the image can now be rendered at the correct size existing VIPP® jobs referencing JPEG files with Exif markers can experience a backward incompatibility issue. To fix this the scale option of the **ICALL** command can be changed. When the change impacts production, the old behavior can be re-instantiated by temporarily adding the following statement in the JDT or in `/usr/xgf/src/xgf.def`: `[ /ProcessExif false ] SETPARAMS`

Multi-strip, grayscale, color images, scaling, and rotation will negatively affect printing performance.

XGFRESDEF syntax is recommended.

Examples

This is an example of how to call a TIFF file embedded in the data stream.

```
500 900 MOVETO
() 1 0 ICALL
<TIFF file contents>
%%%%%%%%%%
200 300 MOVETO ...
```

On DocuPrint NPS, the /var/db/PS.prefix.read file contains paths to the image libraries or to their parents.

Although not normally supported by the TIFF specifications, TIFF files with resolution tags (282 and 283) set to zero, default to 300 dpi. The 262 tag (`PhotometricInterpretation`) defaults to zero. Different XY resolution tags are supported.

Note: To cache images, use **SCALL** instead of **ICALL** because **CACHE** cannot be combined with **ICALL**.

Because **ICALL** and **SCALL** default alignments differ, when converting an **ICALL** syntax without alignment option the **SCALL** syntax should specify alignment option 0 to ensure the correct placement of the image. For example:

```
(myfile.tif) 1 0 ICALL
```

must be converted to:

```
(myfile.tif) CACHE 1 0 0 SCALL
```

Modes

This command is applicable in all modes.

Related commands

- BCALL
- ENDIMP
- IGNOREBT_off
- IGNOREBT_on

IF/ELSE/ELIF/ENDIF

The **IF**, **ELSE**, and **ENDIF** commands introduce native mode and Data Base Master conditional logic. **ELIF** complements the **IF/ELSE/ENDIF** conditional statements set. It combines **ELSE** and IF in one statement to avoid complex nesting.

Syntax

```
IF condition { ... true action ... } ENDIF

IF condition { ... true action ... }
ELSE         { ... false action ... }
ENDIF

IF condition1   { ... action if condition1 is true ... }
ELIF condition2 { ... action if condition2 is true ... }
...
ELIF conditionN { ... action if conditionN is true ... }
ENDIF

IF condition1   { ... action if condition1 is true ... }
ELIF condition2 { ... action if condition2 is true ... }
...
ELIF conditionN { ... action if conditionN is true ... }
ELSE            { ... action if non of the above is true ... }
ENDIF
```

Where

The **condition**: can be one of these:

- **item1 item2 test_op**
- **PCDkey**
- **condition1 condition2 bool_op**
- **item1and item2** can be fixed strings, integers, real numbers, names of variables defined by **SETVAR** or **GETFIELD**, database field names, or XML tags.
- **test_op** is a test operator such as:
  - **eq** equal
  - **ne** not equal
  - **gt** greater than
  - **ge** greater than or equal
  - **lt** less than
  - **le** less than or equal
- **CIEQ** Case Insensitive equal
- **CINE** Case Insensitive not equal

- **HOLD** searches for a string anywhere within the selected field or defined area or characters on a line.

  For more information, refer to Test Operators and Conditional Expressions.
- **bool_op** is a boolean operator such as:
  - **or** true if either condition is true
  - **and** true if both conditions are true

    For more information, refer to Test Operators and Conditional Expressions.
- **PCDkey** is a condition defined by SETPCD

Examples

This example prints `Your gift will be a pocket razor` when the contents of FIELD1 is `Sir`, and `Your gift will be a hand bag mirror` when `FIELD1 contains any other text`.

```
IF FIELD (Sir) eq
{ (Your gift will be a pocket razor) SHL }
ELSE
  { (Your gift will be a hand bag mirror) SHL }
ENDIF
```

Use this example to print the string `Is life good or what!!` for only those people from California who are over 40 and are male.

```
IF State (California) eq
Age (40) ge and
Gender (Male) eq and
{ Is life good or what!!) 0 SHP }
ENDIF
```

Modes

These commands are applicable in native mode, line mode, and database mode.

Related Commands

SETVAR,GETFIELD

IGNOREBT_off

IGNOREBT_off

`IGNOREBT_off` causes the **ICALL** command to abort the job on corrupted or unsupported TIFF files. **IGNOREBT_off** is the default.

IGNOREBT

`IGNOREBT` stands for Ignore Bad TIFFs.

Syntax

```
IGNOREBT_off
```

Modes

This command is applicable in all modes.

Related Commands

IGNOREBT_on, ICALL

IGNOREBT_on

`IGNOREBT_on` causes corrupted or unsupported TIFF files to be ignored by the **ICALL** command. The related TIFF file is not printed, and the job finishes. A list of all ignored TIFF files is produced on the error sheet at the end of the job. By default, the **ICALL** command aborts the job on corrupted or unsupported TIFF files.

The default is `IGNOREBT_off`.

Syntax

```
IGNOREBT_on
```

Modes

This command is applicable in all modes.

Related Commands

IGNOREBT_off, ICALL

ILAND

`ILAND` sets inverse landscape orientation. Inverse landscape orientation is obtained by rotating a portrait sheet 90 degrees clockwise.

Syntax

```
ILAND
```

Modes

This command is applicable in all modes.

Related Commands

LAND,IPORT,PORT

INDEXALIGN

INDEXALIGN associates an align value with an ALIindex. When defined by INDEXALIGN, an **ALIindex** follows the same rules and behaviors as any index that is defined by an **INDEX** command. However, **INDEXALIGN** is only effective inside text blocks printed with **SHP**.

For more information, refer to INDEXCOLOR.

Syntax

```
/ALIindex align INDEXALIGN
```

Where

ALIindex

Is an alphanumeric string.

Align

is one of these alignment codes:

- **0** left
- **1** right
- **2** center
- **3** justify with last line aligned left
- **4** justify with last line aligned right
- **5** justify with last line centered
- **6** justify all lines

Examples

```
(//) 2 SETFTSW
/F1 /NHE 12 INDEXFONT
/A2 2 INDEXALIGN
/A3 3 INDEXALIGN
(Centering the following word
//A2CENTERED
//A3in the middle of a justified text block) 23 SHP
```

Modes

This command is applicable in all modes.

Related Commands

SETLSP,SHP and SHp,SETFTSW

INDEXBAT

INDEXBAT associates a BATkey with a BATindex.

Once defined by INDEXBAT, a BATindex follows the same rules and behaviors as a Colorindex defined by INDEXCOLOR. For more information, refer to INDEXCOLOR.

Syntax

```
/BATindex /BATkey INDEXBAT
/BATindex null INDEXBAT
```

Where

BATindex

Is an alphanumeric string.

BATkey

Is a **BATkey** defined by **SETBAT**.

null

Defines a **BATindex** to cancel the current one.

For more information on pre-defined **BATkeys**, refer to *Standard lists, tables, keys, and attributes* in the *FreeFlow VI Compose User Guide*.

Examples

This example prints `Switching from underlined text to regular`.

```
/U /UNDL INDEXBAT
/N null INDEXBAT
(Switching from //U underlined text//N to regular) 0 SHMF
```

Modes

This command is applicable in all modes.

Related Commands

- INDEXFONT
- INDEXCOLOR
- SHMF, SHMf, and SHmf
- SHX
- SETFTSW

INDEXCOLOR

INDEXCOLOR associates a color and pattern with the **Colorindex** key.

When defined by **INDEXCOLOR**, **Colorindex** can be used externally as a stand-alone command between subsequent SHX commands, or following a color switch prefix inside printable data processed by **SHMF** or **SHP**. The color switch prefix is defined by **SETFTSW** (the default is //).

When used externally, **Colorindex** can be any alphanumeric string starting with an alphabetic character. When used inside printable data, **Colorindex** can be an alphanumeric character string whose length is defined by **SETFTSW** (the default is 1).

Colorkeys are defined in `/usr/xgf/src/xgf.gep`.

Syntax

```
/Colorindex Colorkey INDEXCOLOR
/Colorindex /Colorkey INDEXCOLOR
/Colorindex [ Colorkey PATkey ] INDEXCOLOR
/Colorindex null INDEXCOLOR
/Colorindex (ColorKey~Tintlevel) INDEXCOLOR
/Colorindex (ColorKey#TRlevel) INDEXCOLOR
```

Where:

Colorindex	is an alphanumeric string.
Colorkey	is a Colorkey defined in /usr/xgf/src/xgf.gep.
PATkey	is a pattern key defined by SETPAT.
null	leaves the area transparent instead of being filled with a color.
Tintlevel	is a real number (between 0 and 1) specifying the tint level.
TRlevel	is a real number (between 0 and 1) specifying the transparency level. TRlevel is only effective in VI eCompose and APPE RIP. On a PS RIP it is emulated by an opaque tint.

Note: For more information on pre-defined color and pattern keys, refer to *Standard Lists, Tables, Keys, and Attributes* and *Color Tints* in the *FreeFlow VI Compose User Guide*.

Examples

This example illustrates how to apply tint and transparency color indexes.

```
/T (RED~.5) INDEXCOLOR
/R (RED#.5) INDEXCOLOR
(//TText using tinted red //RTest using transparent red) 0 SHMF
```

This example shows two text strings being printed using a combination of font indexes, color indexes, tint and transparency indexes.

```
/1 /NHE 18 INDEXFONT
/2 /NHE 24 INDEXFONT
/3 /NHE 18 INDEXFONT
/A BLACK INDEXCOLOR
/B BLUE INDEXCOLOR
/Y (RED ~.5) INDEXCOLOR       % sets 50% tint
/Z (BLUE #.75) INDEXCOLOR     % sets 75% transparency


200 200 MOVETO
(//1//AText using font 1 black //2//BSwitch to font 2 blue //3and Font 3) 0 SHMF


200 400 MOVETO
(//1//AText using font 1 black //2//YSwitch to font 2 red 50% tint //3//2and Font 3 with 75% Transparency) 0 SHMF
```

Modes

This command is applicable in all modes.

Related commands

- INDEXBAT
- SHMF, SHMf, and SHmf
- SETFTSW
- SETPAT
- SETTXC
- SHP and SHp
- SETCOL
- SETTRAN

INDEXFONT

INDEXFONT associates a font name and a font size with **Fontindex**.

Once defined by **INDEXFONT**, Fontindex can be used externally as a stand-alone command between subsequent **SHx** commands, or following a font switch prefix inside printable data processed by **SHMF** or **SHP**. The font switch prefix is defined by **SETFTSW** (the default is //). For more information, refer to SHMF, SHMf, and SHmf.

When used externally, **Fontindex** can be any alphanumeric string starting with an alphabetic character. When used inside printable data, **Fontindex** can be an alphanumeric character string whose length is defined by **SETFTSW** (default is 1).

For more information, refer to Kerning.

Syntax

```
/Fontindex /Fontname size INDEXFONT
```

```
/Fontindex /Fontname sizex sizey INDEXFONT
```

Where:

Fontindex	is an alphanumeric string. The built-in font index, /CRFT, refers to the current font and is intended for RPE tables.
Fontname	is one of the following: The name of a font chosen from the VIPP® font lists enabled by **SETENCODING** in the `/usr/xgf/src/xgf.def` file. For more information, refer to "Standard lists, tables, keys, and attributes" in the *FreeFlow VI Compose User Guide*. A special Fontname defined to facilitate changing the font face to a different member of the same font family without having to re specify the Fontname: **/~REG** for regular **/~BLD** for bold **/~ITL** for italic **/~BDI** for bold-italic **/~CUR** for current font For example, when the current font is Helvetica 12 point as set by **/NHE 12 SETFONT**, an **INDEXFONT** specifying the special Fontname /~BLD changes the face to Helvetica-Bold. When the current font is Times Italic, the same **INDEXFONT** definition changes the face to Times Bold. The special Fontname /~CUR can be used to change the size of a font but keep the current Fontname unchanged. Specifying a font size as null keeps the point size. For more information, refer to Applying Attributes to Fonts.
size	is one of the following: **An integer or real number** specifying the font size in units of 1/72 inches (points). When a size of 0 is given, the font is automatically scaled according to the margins and grid defined by **SETMARGIN** and **SETGRID**. In this case, a fixed font must be used (for example, Courier).

	Null, specifying the current point size should be used.
sizex	allows specification of the font size in the X direction in points. The values are the same as defined in size, however, a value of 0 is not allowed.
sizey	allows specification of the font size in the Y direction in points. The values are the same as defined in size, however, a value of 0 is not allowed.

When the second syntax is used, the font is scaled with different values on the X and Y axis.

Note: The built in variables, *GLT* and *MPR*, which are used in Specialty Imaging, can be used as options in this command.

For more information, refer to Specialty Imaging.

Examples

This example shows the usage of a three-character font index with an **SHMF** command.

```
(//) 3 SETFTSW
/H10 /NHE 10 INDEXFONT
/H12 /NHE 12 INDEXFONT
/BLD /~BLD null INDEXFONT
/CUR /~CUR 10 INDEXFONT
(//H10 use Helvetica 10 //H12 use Helvetica 12) 0 SHMF
(//BLD use Helvetica Bold 12 //CUR use Helvetica Bold 10) 0 SHMF
```

Modes

This command is applicable in all modes.

Related commands

- INDEXBAT
- SETENCODING
- SETFONT
- SHMF, SHMf, and SHmf
- SHP and SHp
- SETFTSW

INDEXKERN

INDEXKERN associates kerning options with a **Kernindex** key.

When defined by **INDEXKERN**, **Kernindex** can be used externally as a stand-alone command between subsequent **SHx** commands, or following a switch prefix inside printable data processed by **SHMF** or **SHP**. The switch prefix is defined by **SETFTSW** (the default is //).

When used externally, **Kernindex** can be any alphanumeric string starting with an alphabetic character. When used inside printable data, **Kernindex** can be an alphanumeric character string whose length is defined by **SETFTSW** (the default is 1).

Syntax

```
/Kernindex [ PW_opt TG_opt TK_deg ] INDEXKERN
```

Where:

Kernindex	is an alphanumeric string.
PW_opt	defines the pair-wise kerning option. PW_opt can take one of these values: **0** disable pair-wise kerning. **not 0** enable pair-wise kerning by multiplying the pair-wise kerning values provided by the AFM file (KP, KPX or KPY entries). The recommended value is 1 and may be a real number. **null** keep the current pair-wise kerning option in effect.
TG_opt	defines the generic track kerning option. TG_opt can take one of these values: **0** disable generic track kerning **not 0** enable track kerning by multiplying generic track kerning values defined by VI Compose. The recommended value range is -3 to +3, and may be a real number. **null** keep the current generic track kerning option in effect.
TK_deg	defines the track kerning degree. TK_deg can take one of these values: **0** disable track kerning degree. **not 0** enable track kerning by selecting values from the closest track kern degree defined in the TrackKern entries of the AFM file. Track kerning degrees generally range from -3 to +3, they must be an integer. **null** keep the current track kerning degree in effect.

Modes

This command is applicable in all modes.

Related commands

SHMF, SHMf, and SHmf, SHP and SHp, SETKERN

INDEXLSP

INDEXLSP associates a line spacing value with an **LSPindex**. Once defined by **INDEXLSP**, an **LSPindex** follows the same rules and behaviors as any index defined by an **INDEX** command. For more information, refer to INDEXCOLOR.

INDEXLSP is mainly intended to be used inside text blocks printed with **SHP** in conjunction with fonts of different sizes.

Syntax

```
/LSPindex LSPval INDEXLSP
```

Where:

LSPindex	is an alphanumeric string.
LSPval	is a line spacing value. Real number is in current units.

Examples

```
POINT SETUNIT
(//) 2 SETFTSW
/F1 /NHE 12 INDEXFONT
/L1 14.4 INDEXLSP
/F2 /NHE 20 INDEXFONT
/L2 24 INDEXLSP
(//F1//L1Switching from small to //F2//L2larger text
 //F1//L1and back to small) 0 SHP
```

Modes

This command is applicable in all modes.

Related commands

SETLSP,SHP and SHp, SETFTSW

INDEXOTL

INDEXOTL associates text outline parameters with an **OTL** index. When defined by **INDEXOTL**, an **OTL** index follows the same rules and behaviors as any index defined by an **INDEX** command. For more information, refer to INDEXCOLOR.

Syntax

```
/OTLindex [ LineWidth Colorkey ] INDEXOTL
```

Where:

LineWidth	is the width of the line in current units.
Colorkey	defines the color of the line

Examples

```
/OTL1 [ 2 RED ] INDEXOTL
```

Modes

This command is applicable in all modes.

Related commands

SETOTL,SHP and SHp, SETFTSW

INDEXPIF

INDEXPIF associates a **PIF** destination or note with a **PIF** index. When defined by **INDEXPIF**, a **PIF** index follows the same rules and behaviors as a **Colorindex** defined by **INDEXCOLOR**. For more information, refer to INDEXCOLOR.

Syntax

Where:

PIFindex	is an alphanumeric string.
PIFtype	can be one of these: • **PAGE** a page in the document • **DEST** a named destination defined by PDFDEST • **XPAGE** a page in another PDF document • **XDEST** a named destination in another PDF document • **FILE** a non-PDF document • **URI** an Internet/Intranet site or document • **NOTE** a note
paramX	depending on PIFtype these parameters must be supplied: • [/PAGE pagenum view] • [/DEST /destname] • [/XPAGE (fileref) pagenum view] • [/XDEST (fileref) /destname] • [/FILE (fileref)] • [/URI] • [/URI (URIstring)] • [/NOTE (title) (contents)] • [/NOTE (title) (contents) notetype color option]

Note: For a list of possible values, refer to SETPIF.

null	defines a PIF index to cancel the current PIF index.

Note: Such an index is useful only when a PIF index is used on a fragment of text inside a string that is printed by SHP or SHMF. For all other usages, the PIF index is canceled automatically when it has been associated with an element.

Note: Single byte data can be encoded using ISO-8859-1. Multi-byte data can be encoded using UTF8. It can be converted automatically into UTF16 by VI Compose for insertion in the PDF because this is the only multi-byte encoding supported by the PDF format. To trigger the conversion of UTF8 data to UTF16 the current font selected by **SETFONT** or **INDEXFONT** can have an UTF8 encoding.

Examples

This example shows how to associate a URL with a logo and a bookmark, using a PIF index:

```
/LX [ /URI (http://www.xerox.com) ] INDEXPIF
/LX [ /URI (http://www.xerox.com) ] INDEXPIF
LX (Xerox web site) BOOKMARK
```

Modes

This command is applicable in all modes.

Related commands

- PDFDEST
- SETPIF
- BOOKMARK
- PDFOPEN
- PDFINFO

INDEXRPE

INDEXRPE registers the current RPE under a key.

In the syntax example below, **INDEXRPE** registers the current **RPE** under the **RPEname** used later as a self-execute command in a JDT or Native Mode Prefix (NMP). **INDEXRPE** can be coded immediately after **ENDRPE**.

The last **RPE** defined by **ENDRPE** remains active.

For more information, refer to RPE Command Information and to other related **RPE** commands.

Syntax

```
/RPEname INDEXRPE
```

Modes

This command is applicable in line mode.

Related commands

- BEGINRPE
- ENDRPE
- FROMLINE
- RPEKEY

INDEXSST

INDEXSST associates an SST_param sequence with an **SSTindex** . When defined by **INDEXSST**, an **SSTindex** follows the same rules and behaviors as a **Colorindex** defined by **INDEXCOLOR**.

This index is used to activate Subscript and superscript attributes. For more information, refer to SETTXS.

Syntax

```
/SSTindex sst_param INDEXSST
```

```
/SSTindex [ Ydispl Fshrink ] INDEXSST
```

Where:

SSTindex	is an alphanumeric string.
sst_param	is defined in **SETTXS**. Refer to **SETTXS** for a description of this operand.
Ydispl	defines the vertical offset as a factor of the current font size. It can be positive, shown as superscript, or negative, shown as subscript. The expected value range is [-1 +1].
Fshrink	defines the shrink factor to apply to the current font. The expected value range is [>0 +1].

Examples

Both of these examples print `March, 17th`.

```
/1 /NHE 16 INDEXFONT
/2 /NHE 8 INDEXFONT
/3 20 INDEXSST
/4 null INDEXSST
100 3000 MOVETO
(//1March, 17//2//3th//4//1) 0 SHMF

/S /SUP INDEXSST
/N null INDEXSST
/NHE 16 SETFONT
100 3000 MOVETO
(March, 17//Sth//N) 0 SHMF
```

This example sets a superscript with a vertical offset equal to 40 % of the current font size, and current font shrink at 60 %.

```
/ss1[.4.6]INDEXSST
```

Modes

This command is applicable in all modes.

Related commands

- SETFTSW
- SETTXS
- SHMF, SHMf, and SHmf
- SHP and SHp

IPORT

IPORT sets inverse portrait orientation. Inverse portrait orientation is obtained by rotating a portrait sheet by 180 degrees.

Syntax

```
IPORT
```

Modes

This command is applicable in all modes.

Related commands

ILAND, LAND, PORT

IREVERSE_off

IREVERSE_off causes bi-level images to print in normal mode. This is the default.

Bi-level images are transparent. White parts do not overlay previous marks on the page.

Syntax

```
IREVERSE_off
```

Modes

This command is applicable in all modes.

Related commands

ICALL, IREVERSE_on

IREVERSE_on

IREVERSE_on causes bi-level images to print in reverse mode. The default is **IREVERSE_off**.

Bi-level images are transparent. White parts do not overlay previous marks on the page.

Syntax

```
IREVERSE_on
```

Modes

This command is applicable in all modes.

Related commands

ICALL, IREVERSE_off

JOG_on and JOG_off

Use **JOG_on** and **JOG_off** to offset pages on a page-by-page basis, rather than on a set-by-set basis as with **ENDOFSET**. When **JOG_on** is coded at the beginning of a page, the next, and all subsequent pages, are offset from each other. **JOG_off** stops offsetting beginning with the next page. The deprecated command, **OFFSET_on**, cannot be used when using **JOG_on**.

Using **JOG_on** and **JOG_off** is more efficient than using **ENDOFSET**.

Note: These commands are only effective on DocuPrint NPS and FreeFlow Print Server systems.

Syntax

```
JOG_on

JOG_off
```

Examples

Use this example to end a multi-page **DBM** to offset each **DBM** document from the next.

```
....
PAGEBRK                 % end of last-but-one page
JOG_on
......                  % last page
PAGEBRK
JOG_off
```

Modes

These commands are applicable in all modes.

Related commands

ENDOFRUN, ENDOFSET

LAND

LAND sets landscape orientation. Landscape orientation is obtained by rotating a portrait sheet 90 degrees counterclockwise. The default orientation is **PORT** and is set in the `/usr/xgf/src/xgf.def` file.

Syntax

```
LAND
```

Modes

This command is applicable in all modes.

Related commands

ILAND,IPORT, PORT

LMSKIP

Use **LMSKIP** to skip the beginning of line mode data by a number of lines, a number of characters, or an array of bytes. You can place the **LMSKIP** command in the JDT or before the **STARTLM** command.

Syntax

```
item option LMSKIP
```

Where:

item	is: • A number of lines or bytes for options /L or /C • An array of ASCII numbers for option /B
option	is: • **/L** skip lines • **/C** skip characters • **/B** skip bytes

Examples

```
3 /L LMSKIP                          % skip first 3 lines
10 /C LMSKIP                         % skip first 10 characters
[ 16#0A 16#0D 16#0C ] /B LMSKIP      % skip all bytes whose ASCII
                                     % value is either 10, 13 or 12
[ 10 12 13 ] /B LMSKIP               % skip all bytes whose ASCII value
                                     % is either 10, 13 or 12
```

Modes

This command is applicable in line mode.

Related commands

STARTLM

MAKEVMFILE

MAKEVMFILE stores data, generally the contents of a file, in memory as a virtual file. Use **MAKEVMFILE** in combination with **XGFRESDEF** to embed external files used by **RUN**, **SETLMFILE**, **SETDLFILE** in a self-contained VIPP® PostScript file.

Syntax

```
MAKEVMFILE
data .........
%%EOD_XGF
```

Where:

data	represents the contents of a file.
%%EOD_XGF	can be appended at the end to indicate the end of the data.

Examples

This example shows how to embed a PostScript file in a self-contained VIPP® file.

```
/doc1.ps
MAKEVMFILE
data .........
%%EOD_XGF
XGFRESDEF
```

Modes

This command is applicable in all modes.

Related commands

XGFRESDEF

MAKEVMFORM

MAKEVMFORM creates a procedure suitable for use with the **SETFORM** or **SETBFORM** commands. Use **MAKEVMFORM** to avoid encapsulation between braces ({..... }). This creates VIPP® forms out of PostScript files generated by document processing applications. When using this format, all limitations of brace encapsulation are avoided except for memory limitations.

Syntax

```
MAKEVMFILE
data .........
%%EOD_XGF
```

Where:

data	represents the contents of a file.
%%EOD_XGF	must be appended at the end to indicate the end of the data.

Examples

This example shows how to create a VIPP® form from a PostScript file.

```
%!
%%Title: form1.ps
MAKEVMFORM
PS code ......
%%EOD_XGF
```

Modes

This command is applicable in all modes.

Related commands

SETFORM, SETBFORM

MAXICODE

MAXICODE creates and images a **MaxiCode** barcode based on the specified string and parameter data. No special fonts are required.

Note: **PAGEBRK** is not allowed between the barcodes of a structured append sequence.

Syntax

```
(msgdata) mode MAXICODE
(msgdata) mode align MAXICODE
(msgdata) mode [ posval totval ] MAXICODE
(msgdata) mode [ posval totval ] align MAXICODE
(msgdata) mode scale rotate align MAXICODE
(msgdata) mode [ fit-in-width ] rotate align MAXICODE
(msgdata) mode [ posval totval ] scale rotate align MAXICODE
(msgdata) mode [ posval totval ] [ fit-in-width ] rotate align MAXICODE
```

Where:

(msgdata) is a string that contains the data to be encoded. For more information, refer to msgdata formatting requirements

mode defines the structure of the barcode data and error correction within the symbol. There are two obsolete, and five supported modes:

0 obsolete.

1 obsolete.

2 creates a US structured carrier message that is used as a destination sortation symbol by carriers in the shipping industry.

3 creates an international structured carrier message that is used as a destination sortation symbol by carriers in the shipping industry.

4 creates a symbol that encodes information for purposes other than the shipping industry. Mode 4 encodes a maximum of 93 characters or 138 digits, but it contains less error correction/detection than mode 5.

5 creates a symbol that encodes information for purposes other than the shipping industry. Mode 5 encodes a maximum of 77 characters or 113 digits, but it provides more error correction or detection than mode 4.

6 creates a symbol that encodes a message used to program barcode readers (scanners). Mode 6 encodes a maximum of 93 characters or 138 digits.

Modes 2, 3, 4, and 6 use Enhanced Error Correction (EEC) for the primary message and Standard Error Correction (SEC) for the secondary message. Mode 5 uses EEC for both the primary and secondary messages.

align	indicates which point of the barcode can be aligned on the secondary print position, using these values: **0** top left (default) **1** top right **2** top center **10** bottom left **11** bottom right **12** bottom center **20** center left **21** center right **22** center center
[posval totval]	is the optional structured append array. This array is specified when more than one and up to eight MaxiCode barcodes are to be appended in a structured format. posval is the position of the barcode. totval is the total number of appended barcodes.

msgdata Formatting Requirements

The format of msgdata is dependent upon the mode of the barcode as explained below.

All modes:

When this optional message header is specified, it must be positioned at the beginning of msgdata:

```
[)><RS>01<GS>YY
```

(In this header, YY is a two-digit representation of the year.)

Modes 2 and 3 only:

msgdata contains a postal code, reference mode 2 only and mode 3 only for specific, 3-digit country code, and 3-digit service class in that order. The postal code follows the message header, when specified in msgdata; otherwise the postal code can be positioned at the beginning of msgdata.

In a structured append sequence:

Each barcode of a structured append sequence contains the same mode, postal code, country code, and service class.

Mode 2 only:

The postal code can be 2 to 9 digits. No dashes or other characters are allowed. A 5-digit postal code with a country code of 840 can be right filled with 4 zeroes. All other postal codes with a length less than 9 digits can be left filled with zeros.

Mode 3 only:

The postal code can be 2 to 6 characters, but a postal code with less than 6 characters can be space filled to equal 6 characters. Each character can be a number or uppercase letter.

Modes 4, 5, 6 only:

msgdata does not contain a postal code, country code, or service class.

In a structured append sequence:

Each barcode of a structured append sequence contains the same mode.

Examples

This is an example of a mode 2 barcode:

```
([\)><RS>01<GS>96152382802<GS>840<GS>001<GS>1Z00004951<GS>UPSN
<GS>06X610<GS>159<GS>1234567<GS>1/1<GS><GS>Y<GS>634 ALPHA DRIVE
<GS>PITTSBURGH<GS>PA<RS><EOT>) 2 MAXICODE
```

Where:

<GS>	is a special character sequence used to separate fields
<RS>	is a special character sequence indicating end of format
<EOT>	is a special character sequence indicating end of transmission
<FS>	is a special character sequence used to separate the primary and secondary address numbers not shown in above example.

Note: The brackets in the special character sequences are necessary to differentiate them from normal uppercase characters.

[)><RS>01<GS>96	message header, transportation data, and format header; 96 is the last 2 digits of the year
152382802	Postal code
840	Country code, can be 3 digits
001	Service class, can be 3 digits
1Z00004951	Tracking number
UPSN	SCAC
06X610	UPS account number
159	Julian day of pickup
1234567	Shipment ID #
1/1	Package n/x
Y	Address Validation
634 ALPHA DRIVE	Ship To Street Address
PITTSBURGH	Ship To City
PA	Ship To State

This is an example of a mode 3 barcode:

```
([\)><RS>01<GS>V6J5G3<GS>124<GS>001<GS>1Z00004951<GS>UPSN
<GS>06X610<GS>159<GS>1234567<GS>1/1<GS><GS>Y<GS>1090 W PENDER ST
<GS>VANCOUVER<GS>BC<RS><EOT>) 3 MAXICODE
```

That the structure of msgdata for mode 2 and 3 barcodes is very similar. Only the postal code

requirements are different.

This is an example of mode 2 barcodes in a structured append sequence:

```
100 700 MOVETO
([\)><RS>01<GS>96152382802<GS>840<GS>001<GS>1Z00004951<GS>UPSN<GS>
06X610<GS>159<GS>1234567<GS>1/1<GS><GS>Y<GS>634 ALPHA DRIVE<GS>
PITTSBURGH<GS>PA<RS><EOT>) 2 [1 2] MAXICODE
100 1100 MOVETO
([\)><RS>01<GS>96152382802<GS>840<GS>001<GS>05<GS>400123456789<RS>
<EOT>) 2 [2 2] MAXICODE
```

This is an example of mode 5 barcodes in a structured append sequence:

```
100 700 MOVETO
(This is message 1 of a structured append sequence) 5 [1 8] MAXICODE
100 1100 MOVETO
(This is message 2 of a structured append sequence) 5 [2 8] MAXICODE
100 1500 MOVETO
(This is message 3 of a structured append sequence) 5 [3 8] MAXICODE
100 1900 MOVETO
(This is message 4 of a structured append sequence) 5 [4 8] MAXICODE
500 700 MOVETO
(This is message 5 of a structured append sequence) 5 [5 8] MAXICODE
500 1100 MOVETO
(This is message 6 of a structured append sequence) 5 [6 8] MAXICODE
500 1500 MOVETO
(This is message 7 of a structured append sequence) 5 [7 8] MAXICODE
500 1900 MOVETO
(This is message 8 of a structured append sequence) 5 [8 8] MAXICODE
```

A **MOVETO** command is required between the **MAXICODE** commands of a structured append sequence to prevent the overlapping of images.

Modes

This command is applicable in all modes.

Related commands

- AZTEC
- DATAMATRIX
- PDF417
- QRCODE
- MOVEH
- MOVEHR
- MOVETO

MOVEH

MOVEH sets the secondary horizontal print position (PP) with optional dot leading.

Syntax

```
hpos MOVEH
[ hpos GEPkey ]MOVEH
```

Where:

hpos	is the secondary horizontal position in current units. When using **MOVEH**, hpos is measured from the left edge of the page.
GEPkey	is the **GEPkey** used to define the appearance of the dot leading. When a **GEPkey** is present, a line is drawn from the current secondary horizontal print position to the left of the next text item imaged with an **SHx** command.

Examples

This example prints:

```
Introduction . 1
100 MOVEH (Introduction) SH [1500 D1] MOVEH (1) SH
```

Modes

This command is applicable in all modes.

Related commands

- MOVETO
- MOVEHR
- NL
- SETLKF

MOVEHR

The **MOVEHR** command sets the secondary horizontal print position relative to the last horizontal main position as defined in the last MOVETO or the current frame with optional dot leading.

Syntax

```
hrpos MOVEHR

[ hpos GEPkey ] MOVEHR
```

Where:

hrpos	is the distance to move along the horizontal axis from the last horizontal main position defined in the last **MOVETO** or the current frame. When not specified, the initial PP is 0,0. hrpos can also be a string or variable. This allows you to use DBM variables as hrpos operands. The origin (0,0) is located at the bottom left corner of the page, or at the top left corner when **ORITL** is specified.
GEPkey	is the **GEPkey** used to define the appearance of the dot leading. When a **GEPkey** is present, a line is drawn from the current secondary horizontal print position to the left of the next text item imaged with an **SHx** command.

Examples

This example prints:

```
Introduction ..............................1

(Introduction) SH [1500 D1] MOVEHR (1) SH
```

Modes

This command is applicable in all modes.

Related commands

- MOVEH
- MOVETO
- NL
- ORIBL
- ORITL
- SETLKF

MOVETO

The **MOVETO** command sets both the main and secondary print positions. In the syntax example below, **MOVETO** sets the main and secondary PPs to hpos and vpos.

The **MOVETO** command causes the page to be initialized.

Syntax

```
hpos vpos MOVETO
```

Where:

hpos	is along the horizontal axis.
vpos	is along the vertical axis.

When not specified, the initial print position is 0,0. *vpos* and *hpos* can also be strings or variables. This allows you to use **DBM** variables as *vpos* and *hpos* operands.

The origin, 0,0 is located at the bottom left corner of the page or at the top left corner when **ORITL** is specified.

Modes

This command is applicable in all modes.

Related commands

- MOVEH
- NL
- ORIBL
- ORITL
- SETLKF

MSPP_on

In Multi-Up mode, the **MSPP_on** command allows **ENDOFSET**, **ENDOFRUN**, and **JOG_on/JOG_off** to be placed at the beginning of the last physical page rather than the beginning of the last logical page.

By default, VI Compose forces the last logical page on a new physical page to execute the **ENDOFx** command. Although the set remains consistent, this may lead to undesirable page splitting. **MSPP_on** avoids this.

Syntax

```
MSPP_on
```

Note: When using **MSPP_on**, it is the responsibility of the customer application to insert commands in the position that identifies the first logical page of the last physical page of the set.

Modes

This command is applicable in all modes.

Related commands

- ENDOFRUN
- ENDOFSET
- TWOUP
- SETMULTIUP
- JOG_on and JOG_off

MUL

MUL multiplies a numeric variable defined by **SETVAR**, or an XML variable by a number.

Syntax

`/VARname number MUL`

`/^XMLname number MUL`

Where:

/VARname	refers to a numeric variable previously initialized by **SETVAR**.
/^XMLname	refers to an XML variable. In general XML variables do not need to be explicitly initialized. VI Compose initializes all XML variables to an empty string, which is equivalent to a numeric string equal to zero.
number	is the number by which the variable is multiplied. It can be an integer, a real, or a numeric string. When large numbers are involved a numeric string is mandatory.

Numeric strings accommodate large numbers up to 40 digits, 25 digits for the integer part and 15 digits for the decimal part. In a numeric string the negative sign and the decimal delimiter are defined by the parameters /DecimalPoint and /NSign and can occur anywhere in the string.

It is mandatory to set these parameters with appropriate values to ensure accurate results. Defaults are defined in the file `/usr/xgf/src/xgf.def`. Characters in the numeric string other than these two plus the digits 0-9 are ignored.

The initial length of the string defined by **SETVAR** is extended automatically up to 40 digits when needed.

Reals and integers can be used only for small values <= 99999. For instance the implementation of a counter. The decimal delimiter, if present, is always the point (.). The negative sign, if present, is always the minus (-) and can be the first character.

Examples

```
/VAR.CNT1 123 SETVAR
/VAR.CNT1 12 MUL
/VAR.CNT1 -3 MUL
/VAR_SUM (1,234,890,566,00-) SETVAR
/VAR_SUM (.15) MUL
```

Modes

This command is applicable in all modes.

Related commands

- ADD
- SUB
- DIV
- SETVAR
- ++ and –

NEWBACK

In duplex mode, **NEWBACK** forces the current page to print on the next available back of a sheet. You can code **NEWBACK** after a page delimiter, such as **PAGEBRK**, Form Feed, or Skip to channel one.

Syntax

```
NEWBACK
```

In Multi-Up mode, an implicit **NEWSIDE** is also performed.

Modes

This command is applicable in all modes.

Related commands

- DUPLEX_on
- NEWFRONT
- NEWSIDE
- PAGEBRK
- SETMULTIUP
- TWOUP

NEWFRAME

Use the **NEWFRAME** command to start placing subsequent elements in the next available frame. Using this command can cause a page transition.

Syntax

```
NEWFRAME
```

Modes

This command is applicable in native and database mode.

Related commands

FRCOUNT, GOTOFRAME, SETLKF

NEWFRONT

NEWFRONT forces the current page to print on the front of a new sheet. You can code **NEWFRONT** after a page delimiter such as **PAGEBRK**, **Form Feed**, or **Skip** to channel one.

Syntax

```
NEWFRONT
```

In Multi-Up mode, an implicit **NEWSIDE** is also performed.

Modes

This command is applicable in all modes.

Related commands

- DUPLEX_on
- NEWBACK
- NEWSIDE
- PAGEBRK
- SETMULTIUP
- TWOUP

NEWGROUP

NEWGROUP is an **RPE** sub command that allows you to create **RPE** Groups inside an **RPE** definition. The command is used when a data stream uses prefixes that do not follow the last digit rule for line grouping. All **RPEKEY** definitions belonging to a group can be placed together and preceded by a **NEWGROUP** command.

Refer to RPE Command Information and to other related **RPE** commands.

Examples

```
6 SETRPEPREFIX
10 BEGINRPE
  NEWGROUP
    /LINECR RPEKEY [ .... ]
    /LINEDB RPEKEY [ .... ]
  NEWGROUP
    /NAME01 RPEKEY [ .... ]
    /NAME02 RPEKEY [ .... ]
    /ADRE01 RPEKEY [ .... ]
    /ADRE01 RPEKEY [ .... ]
    /ZPCITY RPEKEY [ .... ]
ENDRPE
```

Modes

This command is applicable in line mode.

Related commands

BEGINRPE, ENDRPE

NEWPOS

NEWPOS, an **RPE** sub command, forces a new independent set of print position values to be kept in memory. It is used when a new print position is necessary for a group of **RPE** entries to print the same field at different locations on the page.

Examples

In addition, with this new syntax, records belonging to the same RPE group can be nonconsecutive in the data file. Old and new syntaxes are exclusive in a specified **RPE** definition set by an **RPEKEY** or **FROMLINE** command. However, **RPE** definitions with both syntaxes can be mixed in an RPE library composed of several **RPEKEY** or **FROMLINE** command placed between **BEGINRPE** and **ENDRPE**.

For more information, refer to RPE Command Information and to other related RPE commands.

Modes

This command is applicable in line mode.

Related commands

RPEKEY

NEWSIDE

In Multi-Up mode, **NEWSIDE** forces the current logical page to print on the next physical page. You can code **NEWSIDE** after a page delimiter, such as **PAGEBRK**, **Form Feed**, or **Skip** to channel one.

Syntax

```
NEWSIDE
```

Modes

This command is applicable in all modes.

Related commands

- NEWBACK
- NEWFRONT
- PAGEBRK
- SETMULTIUP
- TWOUP

NEWSTACK

This command is only available when using the generic mode of **ZSORT**. Refer to ZSORT in this document.

This command ends the current **ZSORT** stack with all records processed so far and begins a new stack. It is intended to be used inside a conditional statement to start a new stack when the condition is true.

NEWSTACK can be placed before page initialization before any **MOVETO** or mark on the page.

Syntax:

```
NEWSTACK
```

Example

```
IF Fieldx NEXT_FieldX ne { NEWSTACK } ENDIF
```

Modes

This command is applicable in all modes when combined with generic **ZSORT**.

Related Commands

ZSORT

NL

The **NL** command resets the horizontal print position (PP) and forwards the vertical print position.

In the syntax example below, NL resets the main and secondary horizontal print positions to the last values specified by **MOVETO** and forwards the vertical print position by the **SETLSP** or **LSPval** values.

Syntax

```
NL
LSPval NL
```

Where:

LSPval	is the value in current units that can be added to the current vertical position. When not present, the default value set by **SETLSP** is used.

Modes

This command is applicable in all modes.

Related commands

MOVEH,MOVETO

NMP_off

NMP_off disables Native Mode Prefix (NMP) records processing.

Syntax

```
NMP_off
```

Modes

This command is applicable in line mode and database mode.

Related commands

% % XGF, SETNMP

OMRINIT

OMRINIT initializes **OMR** code processing. An **OMR** code is a sequence of small vertical bars used to drive an automated mailing system. The height, thickness, spacing and configuration of the bars can be defined through the /OMRconfig parameter.

Once initialized, the **OMR** code can be printed on each page using the **OMRSHOW** command.

This command can be coded in a native mode job, in a **JDT** or in an **NMP**. It can be invoked only once in a job or repeated several times when settings need to be changed during the job, for example, between sets.

OMRINIT must be coded before the page is initialized, prior to any marking command, including **MOVETO**.

Syntax

```
/plex feed_count fold _count annexes OMRINIT
```

Where:

/plex	can be one of the following: **/F OMR** code is intended on front pages **/B** OMR code is intended on back pages
feed_count	can be one of the following: **1–255** feed envelope after that number of pages **0** no feed for the next pages The feed bar is placed at the position I or I in the configuration string.
fold_count	can be one of the following: **1–255** intermediate fold after that number of pages **0** no intermediate fold for the next pages The fold bar is placed at the position F in the configuration string.
annexes	is an integer, the power of 2 component of which represents a bar intended to trigger an insertion or a side action. Example: **+1** insert annex 1 **+2** insert annex 2 **+4** insert annex 3 **+8** do not close envelope **+16** and so on

Note: Bar +1 is placed at the position X or x in the configuration string and is followed by the other bars. The maximum number of bars depends on the mailing system configuration.

Examples

This example initializes the **OMR** code to be printed on the front page, feed 7 pages in the envelope,

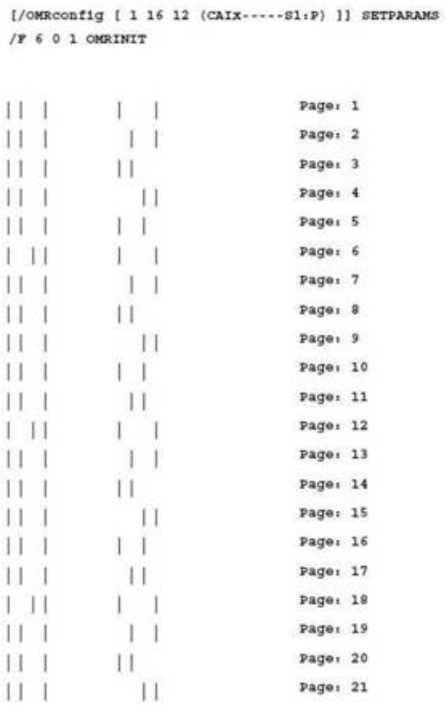
fold each 2 pages and insert annexes 1 and 3.

```
/F 7 2 5 OMRINIT
```

Note: The `feed_count` and `fold_count` parameters express a number of data pages. In simplex or duplex jobs with a back form, this count is equivalent to the number of sheets. In duplex jobs without a back form, this count is equivalent to the number of face prints.

/B is mandatory when **OMRSHOW** is called on a back form.

The graphic below is an example of /OMRconfig and **OMRINIT** settings followed by a collection of **OMR** codes as they would be printed on 21 consecutive pages according to those settings:



OMRconfig Parameter

Each mailing system requires a different OMR configuration described in its documentation. This parameter can be used to define the configuration of the OMR code. Since a given site generally has only one type of mailing system, it can be placed in xgf.def so that it is defined only once. It can also be placed in the job itself (JDT, XJT, and so on.)

Syntax

```
[/OMRconfig [ width height spacing (config_string) ]] SETPARAMS
```

Where:

width	is the width of a bar in point.
height	is the height of a bar in point.

spacing	is the space between bars in points.
config_string	is a string that can include the following symbols: **C** control bar (present on each OMR code) **I** insert on last page **i** insert on first page **A** alimentation (when no insert) **P** odd parity **p** even parity **Sab** A rolling sequence number (3 positions) **Where**: **S** fixed can be S **a 0** roll number between 0 and 7 **1** roll number between 1 and 7 **7** roll number between 7 and 1 **b -** no reset **+** reset on first page **:** reset on last page **F** fold **L** last page **X** first bar for annexes (on insert page only) **x** first bar for annexes (on all pages) **-** empty space or place holder

With the exception of C and - the symbols can be used only once in a config string. The pairs, P/p, I/i, and X/x, are mutually exclusive.

Examples

This example defines a 16 barcode with insert/alim/fold positions, odd parity, 1-7 rolling sequence with no reset, and up to 6 annexes (+1 to +32).

```
[ /OMRconfig 1 16 12 (C-CPAIS1-FX———) ] SETPARAMS
```

Modes

This command is applicable in all modes.

Related commands

OMRSHOW,DUPLEX_on,DUPLEX_off

OMRSHOW

OMRSHOW prints an OMR code previously initialized by **OMRINIT** at the main print position. It can be included in a form definition.

Syntax

```
OMRSHOW

{ imaging proc } /BC OMRSHOW
```

Where:

The first syntax prints a sequence of OMR bars.

The second syntax prints the OMR sequence as a series of zeros and ones, possibly using a bar code font or 2D barcode.

Examples

To print a sequence of bars:

```
{ 1450 3330 MOVETO OMRSHOW } SETFORM
```

To print the OMR sequence as a 2OF5 bar code:

```
{ /2of5_font size SETFONT
{ 2OF5 SH } /BC OMRSHOW
} SETFORM
```

To print the OMR sequence as a Datamatrix symbol:

```
{ { DATAMATRIX } /BC OMRSHOW } SETFORM
```

Modes

This command is applicable in all modes.

Related commands

OMRINIT, SETFORM, SETBFORM

ONEUP

ONEUP resets one-up mode. One-up mode permits a single logical page to be imaged on each physical page. **ONEUP** is the default.

Syntax

```
ONEUP
```

Modes

This command is applicable in all modes.

Related commands

SETMULTIUP,TWOUP

ORIBL

ORIBL sets the origin of all coordinates, except **RPE**, to the bottom left corner of the page. This command can be the first command in a VIPP® file. **ORIBL** is the default.

Syntax

```
ORIBL
```

Modes

This command is applicable in all modes.

Related commands

ORITL

ORITL

ORITL sets the origin of all coordinates at the top left corner of the page. It should be the first command in a VIPP® file.

Syntax

```
ORITL
```

Modes

This command is applicable in all modes.

Related commands

ORIBL

OVERPRINT_on

OVERPRINT_on enables overprint processing in line mode and data base mode. Use this command to print lines of data over each other when each line ends with a single Carriage Return (CR) instead of **LF** or **CR/LF**. When placed before **STARTDBM** in conjunction with **QSTRIP_on LF** or **CR/LF** can be embedded in fields surrounded by quotes. Those fields can then be printed using **SHP** with option +20.

Syntax

```
OVERPRINT_on
```

Important: Use this command only when necessary as it can affect performance.

Modes

This command is applicable in line mode and data base mode.

Related commands

STARTLM

PAGEBRK

PAGEBRK prints the current page and resets the main and secondary print positions (PP) to 0,0. This command is the only end of page marker in native mode. In line mode, end of page also occurs when Form Feed or channel skip are encountered. For further information, refer to *VIPP® data streams* in the *FreeFlow VI Compose User Guide*.

In Multi-Up mode, **PAGEBRK** skips to the next logical page. A physical page only prints when the last logical page is reached, unless **NEWFRONT**, **NEWBACK**, **NEWFRAME**, or **NEWSIDE** is used.

Syntax

```
PAGEBRK
```

When nothing is imaged on the page, **PAGEBRK** does not produce a blank page. To produce a blank page, you must use, at a minimum, **NL PAGEBRK**.

Modes

This command is applicable in all modes.

Related commands

- BCALL
- ENDIMP
- NEWBACK
- NEWFRAME
- NEWFRONT
- NEWSIDE
- SETLKF

PAGERANGE

PAGERANGE specifies the range of pages to print for the current job.

Note: Execute this command at the beginning of the job, before any page is imaged.

Syntax

```
startpage stoppage PAGERANGE
```

Where:

startpage	is the starting page number.
stoppage	is the ending page number.

Examples

This example only prints pages 50 to 100

```
50 100 PAGERANGE
```

Modes

This command is applicable in all modes.

Related commands

SETPARAMS

PDF417

PDF417 creates and images a **PDF417** barcode based on the specified strings and parameter data. No special fonts are required.

Syntax

```
[/TC (string) /BC (string) /NC (string)] PDF417
[/TC (string) /BC (string) /NC (string)] align PDF417
[/TC (string) /BC (string) /NC (string)] [/Rows RValue /Columns CValue
/ARatio [HValue WValue] /ELevel EValue ] PDF417
[/TC (string) /BC (string) /NC (string)] [/Rows RValue /Columns CValue
/ARatio [HValue WValue] /ELevel EValue ] align PDF417
[/TC (string) ...] scale rotate align PDF417
[/TC (string) ...] [/Rows RValue...]scale rotate align PDF417
[/TC (string) ...] [fit-in-width] rotate align PDF417
[/TC (string) ...] [/Rows RValue...] [fit-in-width] rotate align PDF417
[/TC (string) /BC (string) /NC (string)] [/Rows RValue /Columns CValue /ARatio
[HValue WValue] /ELevel EValue /Truncate TValue] PDF417
[/TC (string) /BC (string) /NC (string)] [/Rows RValue /Columns CValue /ARatio
[HValue WValue] /ELevel EValue /Truncate TValue] align PDF417
```

Where:

/TC is text compaction mode. The allowable characters are printable characters with decimal values 32 - 127 and the following control characters:

(\n) new line or line feed

(\r) carriage return

(\t) horizontal tab

(\\) backslash

Note: To use characters outside the 32–127 range, use /BC mode. The barcode is larger and allows the full range of codes.

/BC is byte compaction mode. It allows the encoding of any 8-bit value from 0 to 255.

/NC is numeric compaction mode. It should be used to encode long strings of consecutive numeric digits. Although numeric compaction mode can be invoked at any digit length, Xerox recommends that it be used when there are 13 or more consecutive digits.

(string) is either text, byte, or numeric data depending on the preceding compaction mode.

align indicates which point of the barcode will be aligned on the secondary print position, using these values:

0 top left (default)

1 top right

2 top center

10 bottom left

	11 bottom right
	12 bottom center
	20 center left
	21 center right
	22 center center
/Rows	is the number of rows in the barcode.
RValue	must be within the range of 3–90.
/Columns	is the number of data columns in the barcode.
CValue	must be within the range of 1–30.
/ARatio	is the aspect ratio of the barcode.

Note: The default /ARatio is [1 1]. An /ARatio of [1 1] does not produce a square barcode. The number of columns per barcode is WValue plus four, for the start and stop columns, and the left and right row indicators.

The /ARatio necessary to create a square barcode depends on many factors, such as the values given to the **SETPARAMS** parameters, DotsPerModule, and RowHeight, the error correction and detection value, the number of rows or columns, and the number of string characters. The best way to create a specific barcode size is through trial and error. When an /ARatio of 1 to 1 creates a barcode that is wider than it is tall, as happens when the default parameters are used, to create a square barcode, try to increase the /HValue, or decrease the /WValue, or both. For example: `/ARatio [5 1]`.

HValue	is the height or number of rows in the barcode.
WValue	is the width or the number of data columns.
/ELevel	is the error correction/detection level.
EValue	must be within the range 0–8, where 0 is the minimum and 8 is the maximum amount of correction.
/Truncate	specifies the truncated **PDF417** barcode.
/TValue	is the status of the truncated **PDF417** barcode:
	true = enabled
	false = disabled

/Rows, /Columns, and /ARatio are optional. Both /Rows and /Columns cannot be specified in the same command. When both are specified, only /Columns can be accepted and /Rows can be recalculated. When neither /Rows nor /Columns is specified, both will be automatically determined based on the default or specified by /ARatio.

/ELevel is also optional. The error level will default to a minimum of two and a maximum of five depending on the amount of data encoded.

One **PDF417** barcode can encode approximately 1850 text characters, 1110 bytes, or 2710 digits at an error level of 0. For an error level of 8, the limits are approximately 830 text characters, 498 bytes, or 1215 digits.

The **PDF417** barcode also supports two new parameters in **SETPARAMS**:

```
[ /DotsPerModule integer
/RowHeight integer
] SETPARAMS
```

Where:

/DotsPerModule	is the number of dots per bar or space in the barcode. The default value is 3.
/RowHeight	is the height of one row in the barcode. The default value is 4 if the error level is low given the amount of data encoded and 3 if the error level is appropriate for the amount of data encoded.

These parameters are modifiable to support a wide range of printers and scanners. However, Xerox recommends that the parameters remain at their default values unless printer resolution or scanner problems require enlarging the bars.

Examples

```
[ /BC <01 02 03 04 05 06 07 08 09> /TC (Test) /NC (00246812345678) ] PDF417
[ /BC (PDF417 symbology) ] [ /ARatio [1 2] /ELevel 1 ] PDF417
[ /NC (12345678901234567890123 45) ] [ /Rows 3 ] PDF417
[ /TC (John Doe\n1405 Ocean Drive\nEl Segundo, CA 90245) ] [ /Columns 3 ]
PDF417
```

Modes

This command is applicable in all modes.

Related commands

- AZTEC
- DATAMATRIX
- MAXICODE
- QRCODE
- MOVEH
- MOVEHR
- MOVETO

PDFBOUND

PDFBOUND enables the creation of optional page boundary boxes in a PDF output created by a VIPP® job. This command does not alter the PDF contents other than by adding the additional bounding boxes in the page. Applications rendering or processing the PDF downstream in the workflow can decide to process or ignore them. When the output of the VIPP® job is not a PDF file the command is ignored and has no effect.

PDFBOUND must be placed before page initialization before any **MOVETO** or mark on the page.

Syntax:

```
/xxxBox [top bottom left right] PDFBOUND
```

Where:

/xxxBox	is one of: • /CropBox • /BleedBox • /TrimBox • /ArtBox
top, bottom, left and right	are offsets to the center of the page (in current units) from MediaBox as defined by **SETPAGESIZE**.

Example:

```
/CropBox [50 50 50 50] PDFBOUND
```

Mode

This command is applicable in all modes.

Related Commands

SETPAGESIZE

PDFDEST

PDFDEST defines a named destination. A named destination can be referenced in a PIF definition of type DEST or XDEST.

Syntax

```
/destname PDFDEST
```

```
/destname [ pagenum view ] PDFDEST
```

Where:

destname	is the destination name (alphanumeric string).
pagenum	is the page number of the destination starting with 1. The default is the current page number. It may be a variable.
view	defines how to adjust the view for the destination. Refer to SETPIF for a list of possible values.

Examples

This example shows how to set a link to a destination (/NOTES) defined later in the job:

```
[ /DEST /NOTES [/Fit] ] SETPIF
100 500 MOVETO (Go to Notes) 2 SHmf
....
PAGEBRK
....
/NOTES PDFDEST
```

Modes

This command is applicable in all modes.

Related commands

- BOOKMARK
- SETPIF
- INDEXPIF
- PDFOPEN
- PDFINFO

PDFFORMOCG

PDFFORMOCG enables PDF optional content at the form level.

PDF optional content is the capability provided by some PDF viewers like Adobe Acrobat and Reader to show or hide certain parts of the PDF document when viewed or printed. These parts appear under the Layer tool bar when a PDF is opened.

PDFFORMOCG is used to map each VIPP® form **planenumber** with a PDF Layer name so it can appear in the Layer tool bar. A typical use case is to allow a background form to be present in the PDF but disabled when the document is printed on preprinted paper.

Syntax

```
[ [ (layerName1) istate1 ] ... [ (layerNameN) istateN ] ] PDFFORMOCG
```

Where:

[(layerNameX) istateX]	is the name and initial state for each form plane number. It can be replaced by null if the layer is not subject to optional content.
(LayerNameX)	is a string containing the name of the layer
istateX	is a boolean indicating the initial state of the layer when opening the document: - true visible - false hidden

Examples

```
Assign an optional layer to 2 of 3 form planes:
(bvr.frm) 0 SETFORM
(bill.frm) 1 SETFORM
(copy.frm) 2 SETFORM
[ [(BVR) true] [(Invoice) true] null ] PDFFORMOCG
```

Modes

This command is applicable in all modes.

Related commands

PDFOCG,SETFORM,SETMAXBFORM

PDFINFO

PDFINFO populates the Document Summary section of the PDF file with information.

Syntax

```
[ /Author (Author of the document)
/Creator (Creator of the document)
/Title (Title of the document)
/Subject (Subject of the document)
/Keywords (List of keywords)
] PDFINFO
```

Where:

Author of the document	contains the author name.
Creator of the document	contains the document creator name.
Title of the document	contains the document title.
Subject of the document	contains the document subject.
List of Keywords	contains the document keywords.

Note: Single byte data can be encoded using ISO-8859-1. Multi-byte data can be encoded using UTF8. It can be converted automatically into UTF16 by VI Compose for insertion in the PDF because this is the only multi-byte encoding supported by the PDF format. To trigger the conversion of UTF8 data to UTF16 the current font selected by **SETFONT** or **INDEXFONT** can have an UTF8 encoding.

Examples

```
[ /Author (John Smith)
/Creator ()
/Title (VIPP PIF Samples)
/Subject (Samples of VIPP Interactive PDF features)
/Keywords (VIPP PDF Interactive PIF)
] PDFINF
```

Modes

This command is applicable in all modes.

Related commands

PDFOPEN

PDFOCG

PDFOCG enables PDF optional content at the element level.

PDF optional content is the capability provided by some PDF viewers like Adobe Acrobat and Reader to show or hide certain parts of the PDF document when viewed or printed. These parts appear under the Layer tool bar when a PDF is opened.

PDFOCG is used to encapsulate any portion of the VIPP® code containing marking commands and assign it a name so it can appear in the Layer tool bar.

Syntax

```
(LayerName) istate PDFOCG % begin optional content group

null PDFOCG % end optional content group
```

Where:

(LayerName)	is a string containing the name of the layer
istate	is a boolean indicating the initial state of the layer when opening the document: - true visible - false hidden

Examples

```
Place the logo in a specific optional layer:
(Logo) true PDFOCG
(xslogo.tif) .4 0 1 ICALL
null PDFOCG
```

Modes

This command is applicable in all modes.

Related commands

PDFFORMOCG

PDFOPEN

PDFOPEN defines how a PDF document can be opened. When associated with a PIF destination it also defines a destination to be used when the document is opened.

Syntax

```
/openmode PDFOPEN
```

Where:

openmode

Selects the mode used when the document is opened. It may be one of these:

- **/UseOutlines** open and display the bookmarks
- **/UseThumbs** open and display thumbnail images
- **/UseNone** open and display none of the above (default)
- **/FullScreen** open in full screen mode

Examples

This example shows how to use **PDFOPEN** to display bookmarks when the PDF file is opened:

```
/UseOutlines PDFOPEN
```

Modes

This command is applicable in all modes.

Related commands

PDFINFO

PORT

PORT sets portrait orientation. Portrait orientation indicates that the short edge of the sheet is on the horizontal (X) axis and the long edge is on the vertical (Y) axis. This is the default.

Syntax

```
PORT
```

Modes

This command is applicable in all modes.

Related commands

- ILAND
- IPORT
- LAND
- SETPAGESIZE

PRECACHE

PRECACHE enables resource pre-caching. It is only effective on a PostScript interpreter that behaves according to Xerox specifications for pre-caching. Consult a Xerox representative for more details.

Pre-cached resources are created on the printer controller (DFE) external storage. **PRECACHE** does not produce any mark on the page. Only subsequent calls to the resource, possibly in subsequent jobs, using **SCALL** or **SETFORM/SETBFORM** will image the resource on the page.

Note: **PRECACHE** is not supported on all controllers. To determine if the controller supports this command, contact a Xerox representative.

Syntax

```
(rname) [rot1 scale1 rot2 scale2...rotN scaleN] PRECACHE
```

Where:

rname	can be a VIPP® segment, EPS, PostScript, or TIFF file.
rotN scaleN	pairs list the rotation and scaling combinations that need to be pre-rendered.

PRECACHE can use files located in these VIPP® resource directories:

- formlib (as defined by **SETFPATH**)
- imglib (as defined by **SETIPATH**)
- mislib (as defined by **SETMPATH**)

Examples

```
(car1.eps) [0 1 90 1 0 .5 90 .5]PRECACHE
```

Related commands

SETFPATH, SETIPATH, SETMPATH

PROCESSDJDE

Use **PROCESSDJDE** to simplify processing **DJDE** (LCDS data stream) jobs by VI Compose. **PROCESSDJDE** eliminates **DJDEs**from the printable data and calls a user-defined procedure for each keyword/parameter pair in the **DJDE** line.

Syntax

```
{ djde_proc } position (djde_prefix) option PROCESSDJDE
```

Where:

djde_proc	is a procedure that can be executed for each keyword/parameter pair in the **DJDE** line. It is a VIPP® command sequence that takes appropriate action depending on **DJDECMD** containing the **DJDE** keyword and **DJDEPAR** containing the related DJDE parameter. **CASE** and/or **IF/ELSE/ENDIF** commands are expected in this procedure.
position	is the position of the **DJDE** prefix in the record starting with 0.
djde_prefix	is the **DJDE** prefix string.
option	is a number that may combine these values: **+1** forces new page on first **DJDE** in a **DJDE** packet **+2** cancels pre-skip on first data line after **DJDE** **+4** cancel line pre-skip on **DJDE** line **+8** cancel post-skip on **DJDE** line

Examples

```
{ CASE DJDECMD {}% default action = none
    (JDL)  {($$DJDEPAR..JDT) VSUB SETJDT }
    (FORM) {($$DJDEPAR..FRM) VSUB SETFORM }
    (FEED) {CASE DJDEPAR {}
               (AUX){(Preprinted) SETMEDIA}
               (MAIN){(Plain) SETMEDIA}
               ENDCASE
           }
    ENDCASE
} 0 ($DJDE$) 3 PROCESSDJDE
```

This example can process the DJDE statement by exclusively assigning the values that follow:

```
$DJDE$ JDL=JDL23, FORM=BILL2. FEED=MAIN, END;
DJDECMD=(JDL), DJDEPAR=(JDL23)
DJDECMD=(FORM), DJDEPAR=(BILL2)
DJDECMD=(FEED), DJDEPAR=(MAIN)
```

Modes

This command is applicable in line mode.

Related commands

CASE, IF/ELSE/ELIF/ENDIF, DJDEBEGIN

QRCODE

QRCODE creates and images a QR Code Model 2 barcode based on the specified strings and parameter data. No special fonts are required.

Syntax

```
[/AC (string) /BC (string) /KC (string) /MC (string) /NC (string)] QRCODE
[/AC (string) /BC (string) /KC (string) /MC (string) /NC (string)]
scale rotate align QRCODE
[/AC (string) /BC (string) /KC (string) /MC (string) /NC (string)]
[/ELevel EValue /SAppend [Posval Totval] /QRver version] QRCODE
[/AC (string) /BC (string) /KC (string) /MC (string) /NC (string)]
[/ELevel EValue /SAppend [Posval Totval] /QRver version] scale rotate align QRCODE
[ /cmode (data)...] [fit-in-width] rotate align QRCODE
[ /cmode (data)...] [/ELevel EValue...] [fit-in-width] rotate align QRCODE
```

Where:

/AC	is alphanumeric data compaction mode. Data is encoded at a density of 2 characters per 11 bits. There are 45 allowable characters: **10** numeric digits (0–9) **26** alphabetic characters (A–Z) and **9** symbols (SP, $, %, *, +, -, ., /, :)
/BC	is 8-bit byte data compaction mode. This mode encodes the 8-bit Latin/Kana character set in accordance with JIS X 0201 (character values 00 to FF hex) at a density of 8 bits per character.
/KC	is Kanji data compaction mode. The Kanji mode encodes Kanji characters in accordance with the Shift JIS system based on JIS X 0208 at a density of one two-byte character per 13 bits. The Shift JIS values are shifted from the JIS X 0208 values.
/MC	is mixed-data stream compaction mode. This mode encodes sequences of data in a combination of any of the compaction modes.
/NC	is numeric data compaction mode. This mode encodes data from the decimal digit set (0–9) at a density of 3 data characters per 10 bits.
(string)	is either alphanumeric, byte, Kanji, numeric, or mixed data depending on the preceding compaction mode.

Note: Data compaction mode and string pairs can be specified in any order, although a mode is required before the corresponding string.

scale	changes the size of the barcode (default is 1). When the scale is 1 the module (cell or the smallest component of the barcode) size is .254 mm, 0.01 in., or 6x6 dots on a 600 dpi printer.

rotate	changes the orientation of the barcode by angle degrees (default is 0, no rotation). A positive value rotates the barcode counterclockwise with respect to its former orientation. For example, when rotate = -90, the barcode can rotate clockwise 90 degrees.
align	indicates which point of the barcode can be aligned on the secondary print position, using these values: **0** top left (default) **1** top right **2** top center **10** bottom left **11** bottom right **12** bottom center **20** center left **21** center right **22** center center

The optional array [/ELevel EValue /SAppend [Posval Totval] /QRver version] contains a list of key or value pairs that can be used to specify the following options:

/ELevel	is the error correction/detection level.
EValue	can be within the range 1–4, where 1 is the minimum and 4 is the maximum amount of correction. The range 1–4 corresponds to levels L, M, Q, and H respectively. When ELevel is not specified, level 2 or M is the default.
/MPattern	is a performance enhancement parameter. It specifies a subset of the otherwise eight mask patterns that are used to evaluate and create the QR Code barcode.
MValue	is a single integer in the range 1 through 8 or an array of 1 to 8 integers.

Although the QR Code specification calls for applying 8 different mask patterns to barcode data and selecting the mask that results in the most desirable barcode, tests indicate that each mask pattern resulted in a readable barcode. Since evaluating mask patterns is a time consuming task, the option of using only a subset of the 8 mask patterns is provided to improve performance.

/QRopt	is a performance enhancement parameter. It enables or disables shortcuts in the analysis of mask patterns.
opt	is -1 (default) or any other integer. -1 enables the shortcuts; other numbers disable the shortcuts.

When shortcuts are enabled, a smaller percentage of barcode data is used in the evaluation of mask patterns. Since tests indicate that every mask pattern results in a readable barcode, in-depth analysis of each mask pattern is unnecessary.

Examples

```
[/AC (AC-42)] [/ELevel 4 /MPattern 1] QRCODE % improve performance

[/AC (AC-42)] [/ELevel 4 /MPattern [2 3 6]] QRCODE % improve performance
```

```
[/AC (AC-42)] [/ELevel 4 /QRopt 1] QRCODE % disable shortcuts, % no
performance improvements
```

/SAppend	is the optional structured append array that is used when more than one and upto 16 QR barcodes are to be appended in a structured format. The array contains two elements that can be used as follows:
Posval	is the position of the barcode within the set of appended barcodes.
Totval	is the total number of appended barcodes.
/QRver	is the size or version of the QR Code barcode.
version	is a number from 1 to 40. When a version number is not specified, the most suitable version can automatically be determined from the number of encoded data characters and the value of **EValue**. When **QRver** and **ELevel** are specified but **QRver** does not have the capacity to hold the given data at that **ELevel**, an error can be reported. When **QRver** is specified and **ELevel** is not, an error can be reported when **QRver** does not have the capacity to hold the given data at the default **ELevel**.

Note: These character sets are supported in the QR Code barcode.

```
JIS X 0201 (JIS 8-bit)
JIS X 0208-1997 (Japanese Graphic, Annex 1 Shift Coded Representation)
ANSI X 3.4 (7-bit).
```

QRCODE and PAGEBRK

QRCODE does not support a QR barcode that will span a **PAGEBRK**. **PAGEBRK** is not allowed between the barcodes of a structured append syntax.

Kanji character 5C

When the second byte of a Kanji character is 5C in hexadecimal, the backslash character, it must be preceded with another 5C or it will be treated as an escape character and stripped before reaching VI Compose. The additional 5C must be inserted into the data string by the application that generates the VIPP® code.

Limitations

QR Code Model 1 is not supported. Model 1 is the original, more restricted version of Model 2.

FNC1 is not supported as it is scarcely used. FNC1 is a mode indicator that identifies symbols encoding messages formatted according to specific predefined industry or application specifications.

Examples

```
[/AC (HTTP://WWW.FUJIXEROX.CO.JP) ] QRCODE
[/NC (12345678901234567890) ] 1 -90 0 QRCODE
[/BC (ハンカクモジ) ] 1 0 0 QRCODE
[/KC ( 富士ゼロックス一申\し込み ) ] [/ELEVEL 3 /QRver 8 ] 1 0 0 QRCODE
[/MC (ABCD1234567 ハンカクモジ富士ゼロックス一申\し込み ) ] [/ELEVEL 2 /QRver 4 ] 2 0 0 QRCODE
[/AC (THIS IS MESSAGE 1 OF A STRUCTURED APPEND SEQUENCE.)] [/SAppend [1 2]] QRCODE
[/AC (THIS IS MESSAGE 1 OF A STRUCTURED APPEND SEQUENCE.)] [/SAppend [2 2]] QRCODE
```

Modes

This command is applicable in all modes.

Related commands

- AZTEC
- DATAMATRIX
- PDF417
- MAXICODE
- QRCODE
- MOVEH
- MOVEHR
- MOVETO

QSTRIP_on

QSTRIP_on strips first and last double quotes or single quotes, when present, from every delimited field in database mode. Quoted and non-quoted fields can be mixed within a record. Field delimiters, as defined by **SETDBSEP**, in quoted fields are not stripped, they are retained as part of the field contents.

Syntax

```
QSTRIP_on
```

Do not code global commands such as **QSTRIP_on**, **DUPLEX_on**, **SETDBSEP**, and **SETBUFSIZE** in the Data Base Master. Place global commands at the beginning of the database file prior to the **STARTDBM** command. As an alternative, global commands can be placed in an external Job Descriptor Ticket file referenced by a **SETJDT** command placed in the database file prior to the **STARTDBM** command.

Examples

```
%!
DUPLEX_on
(;) SETDBSEP
BSTRIP_off
QSTRIP_on
(cas.dbm) STARTDBM
....

%! database file
(cas.jdt) SETJDT
(cas.dbm) STARTDBM
....

%!PS-Adobe-2.0
%%Title: cas.jdt
%%Creator: CAS/RXCH
....
DUPLEX_on
(;) SETDBSEP
QSTRIP_on
BSTRIP_off
....

%!
XGF
QSTRIP_on
(,) SETDBSEP
(dbm1.dbm) STARTDBM
FNAME,NAME,ADDRESS
“John”,“Martin”,“23, Wall Street”
.....
```

This example assigns the variables as follows:

FNAME=John

NAME=Martin

ADDRESS=23, Wall Street

The field names may also be defined by using quotes as follows:

FNAME, NAME, ADDRESS

John, Martin, 23, Wall Street

Modes

This command is applicable in database mode only.

Related commands

STARTDBM

RELVAR

RELVAR removes a previous **SETVAR** definition so that it can be set again by a **SETVAR** statement containing the */INI* option.

Syntax

```
/VARxyz RELVAR
```

Modes

This command is applicable in all modes.

Related commands

SETVAR

REPEAT

Use **REPEAT** to execute a sequence of commands numerous times. **REPEAT** can be used in a Data Base Master when the **SETCYCLECOPY** command cannot be used. **REPEAT** performs the equivalent of the **SETCYCLECOPY** command while in database mode.

Syntax

```
{sequence of VIPP commands} count REPEAT
```

Where:

{sequence of VIPP commands}	is any sequence of native mode commands that produce a portion of a page, a complete page, or several pages.
count	is the number of times the procedure is executed. This operand can be a field name from the database file when the field name contains a numeric value.

In normal operation **REPEAT** will end when the count is reached. The **EXIT** command can be used in a conditional test to exit the **REPEAT** loop.

Use the two VIPP® integer variables, **RPCOUNT** and **RPLEFT**, in the **REPEAT** procedure to check which iteration is being executed.

For control purposes use these variables with **IF/ELSE/ENDIF**:

RPCOUNT	indicates the number of current iterations in the **REPEAT** command
RPLEFT	indicates the number of remaining iterations in the **REPEAT** command

Note: Using RSAVE and RESET in REPEAT is not recommended as unpredictable results may occur.

Examples

This example uses **REPEAT** to print multiple copies of the Data Base Master form for each record contained in the database file, it also changes the number of copies from record to record and triggers an **ENDOFSET** on the last page.

```
IF RPLEFT 1 eq
  {ENDOFSET}
ENDIF
{ 0 2400 MOVETO
 ($$GREETING..tif) VSUB 1 0 ICALL
 PAGEBRK
} NUMBER REPEAT
```

This example shows how to change the font for the last iteration of a **REPEAT** loop.

```
300 2500 MOVETO
{
(I am looping) 0 SHP
IF RPCOUNT 9 gt
   { /NCRB 30 SETFONT } ENDIF
} 10 REPEAT
```

This example shows how to exit a loop when the horizontal position has reached the edge of the page.

```
/VARHoz 300 SETVAR
/VARVer 300 SETVAR
/VARNumber 0 SETVAR

{
VARHoz VARVer MOVETO
VARNumber 50 2 SHP
/VARNumber ++
/VARHor 50 ADD
IF VARHoz 2550 ge    { EXIT} ENDIF
} 24 REPEAT
```

Modes

This command is applicable in all modes.

Related commands

- IF/ELSE/ELIF/ENDIF
- STARTDBM

RESET

RESET restores the VIPP® context to the initial default value or to the value stored by the last **RSAVE**. It cancels all settings forms, fonts, and so on, since the last **RSAVE**.

Syntax

```
RESET
```

Placement

Code this command only after a page delimiter command, for example, **PAGEBRK**, Form Feed, or channel skip.

Do not use

Do not use **RESET** in a **REPEAT** command, or after the **SETCYCLECOPY** command as unpredictable results can occur.

Modes

This command is applicable in all modes.

Related commands

RSAVE

RPEDEF

RPEDEF provides a fast way to build simplified RPE definitions that will process records with a font index. Simplified RPE definitions have no rotation, left alignment, and no field selection.

Use **RPEDEF** in NMP records (%%XGF) when the conversion of **LCDS DJDE FONTS=** statements is required. **RPEDEF** may also be coded in Job Descriptor Tickets.

For more information on other related RPE commands, refer to RPE command information.

Syntax

```
[ [ /font0 Ydispl0 ] [ /font1 Ydispl1 ]... ] RPEDEF
```

Where:

fontN	is a VIPP® font index previously defined by **INDEXFONT**, which will be used with the record font index N.
YdisplN	is the line spacing, in current units, associated with this font.
Colorkey	is a VIPP® Colorkey.
Fheight	specifies a font height value expressed in current units for each font in the list. This causes the top margin set by **SETMARGIN** to refer to the top of the cell rather than to the baseline the default for the first line on the page. This option is primarily used to convert LCDS data streams using font index.

Examples

This example is equivalent to the RPE coding below.

```
[ [ /F0 50 ] [ /F1 80 RED ] [ /F2 [ 60 40 ] ] RPEDEF
1 SETRPEPREFIX
2 BEGINRPE
/0 RPEKEY [ 0 0 0 null 0 50 1 buf_size /F0 BLACK ]
/1 RPEKEY [ 0 0 0 null 0 80 1 buf_size /F1 RED ]
/2 RPEKEY [ 0 0 0 null 40 60 1 buf_size /F2 BLACK ]
ENDRPE
```

This information must be defined in the xgf.def file, in the JDT, or in a previous NMP record:

- Initial position (value of **DJDE BEGIN=**) by **SETMARGIN**
- buf_size by **SETBUFSIZE**
- /font0, /font1, ... with **INDEXFONT**

The value of the first byte in the record, or the second byte when PCC mode is active, is used to select the simplified **RPEKEY** defined by **RPEDEF**. The selection is based on the position of the RPE definition in the **RPEDEF** command.

This example prints hello world using the /F1 font and a vertical displacement of 20 units. *I am still alive* prints using the /F2 font with a vertical displacement of 30.

```
%%XGF [ [/F1 20 ] [/F2 30] ] RPEDEF
0hello world
1I am still alive
```

Font index values start with zero and range from 0 to 9 and then A to Z. When the customer list starts

with 1, the **RPEDEF** list starts with a dummy entry.

When the font index is not the first byte in the record, or the second byte in the record when a PCC is used, the **SETRPEPREFIX** can be coded before **RPEDEF**. When an **EBCDIC** data stream is transformed, the **INDEXFONT** command used to define the font can be coded before the **RPEDEF** command.

In this example, the portion extracted from the record as printable data (0 to 131) ends just before the font index byte.

```
[ 1 131 ] SETRPEPREFIX
[ .......... ] RPEDEF
```

Related commands

- INDEXFONT
- SETBUFSIZE
- SETMARGIN
- SETPAT
- SETTXC
- SETRPEPREFIX
- RPEKEY

RPEKEY

RPEKEY starts an RPE definition that will be invoked by the line prefix when **SETRPEPREFIX** is used or by a **SETRPE** command in the data stream. RPE key names that match, in all but the last character, form a group. Refer to **SETRPEPREFIX** for further information, also refer to RPE command information and to other related RPE commands.

Syntax

```
/rpekeyname RPEKEY
[ align rotate Xinit Xdispl Yinit Ydispl recpos length /font Colorkey ]
...
[ align rotate Xinit Xdispl Yinit Ydispl recpos length /font Colorkey ]
```

Where:

rpekeyname	is a line prefix matching the **SETRPEPREFIX** definition. When the prefix contains one or more spaces the syntax must be rpekeyname instead of /rpekeyname.

Note: Whenever an RPE prefix contains spaces, it requires coding between the parentheses. For example: `(REC1 45 C) RPEKEY [ ... RPE entry ... ]`

For more information, refer to FROMLINE for a complete description of the RPE entry parameters and Extending FROMLINE and RPEKEY commands.

Examples

```
% set RPE fonts
/F1  /NHEB  9  13  INDEXFONT
/F2  /NHEB  10 15  INDEXFONT
/F3  /NHEB  13 15  INDEXFONT
/F4  /NHEB  18     INDEXFONT
/F5  /NHEB  13     INDEXFONT

% RPE definition
5 BEGINRPE

% almt  rot. Xinit  Xdispl Yinit  Ydispl  recpos  length   font   color
/HDR0 RPEKEY
 [  2   0    835    0    300    0    00    99    /F4   BLACK ]
/CL10 RPEKEY
 [  2   0    615    0    445    0    00    99    /F1   WHITE ]
/CL20 RPEKEY
 [  2   0   1199   318   445    0    00    99    /F1   WHITE ]
/CL30 RPEKEY
 [  2   0   3140    0    445    0    00    99    /F5   WHITE ]
/BODY RPEKEY
 [  0   0    230    0    560   75    00    33    /F2   BLACK ]
 [  1   0   1345    0    560   75    33    12    /F2   BLACK ]
 [  1   0   1658    0    560   75    45    12    /F2   BLACK ]
 [  1   0   1976    0    560   75    57    12    /F2   BLACK ]
 [  1   0   2286    0    560   75    69    12    /F2   BLACK ]
 [  1   0   2610    0    560   75    81    12    /F2   BLACK ]
 [  1   0   2945    0    560   75    93    12    /F2   BLACK ]
 [  1   0   3290    0    560   75   105    12    /F3   BLACK ]
ENDRPE
```

Modes

This command is applicable in line mode.

Related commands

- BEGINRPE
- COPYRANGE
- ENDRPE
- FROMLINE
- INDEXRPE
- SETPAT
- SETPCD
- SETRCD
- SETRPEPREFIX
- SETTXC

RSAVE

RSAVE saves the current context, which is restored at a later time using the **RESET** command.

Syntax

```
RSAVE
```

Placement

This command can occur only before the first marking command (for example, **SHx**, **DRAWB**, **DRAWBR**, **DRAWBM**, **DRAWBRM**, **ICALL**, and **SCALL**) on a page and after a page delimiter when it is not the first page. Otherwise the effects are limited to the current page.

Do not use

Do not use **RSAVE** in a **REPEAT** command, or after the **SETCYCLECOPY** command, as unpredictable results can occur.

Modes

This command is applicable in all modes.

Related commands

% % EOF and RESET.

RUN

The **RUN** command executes the VIPP® or PostScript code contained in the referenced file. The referenced file is stored in one of the libraries defined by **SETMPATH**, or in a VIPP®-enabled FreeFlow Makeready repository.

Syntax

```
(file name) RUN
```

```
(file name) option RUN
```

Where:

option	can include: **0** no special action default **1** save/restore encapsulation to protect memory consumption **2** save/restore + basic VIPP®/PostScript interaction **3** save/restore + full VIPP®/PostScript interaction limited to PostScript files created by some drivers.

0 no special action default

save/restore encapsulation to protect memory consumption

save/restore + basic VIPP®/PostScript interaction

save/restore + full VIPP®/PostScript interaction limited to PostScript files created by some drivers.

option 0 (or no option) is used with files containing VIPP® code and is the only possible option with these files.

For applications that protects the **RUN** execution, option 1 can be specified. In addition, options 2 and 3 enable an interaction between VIPP® and PostScript where page-related features such as **SETFORM**, **SETPAGENUMBER**, and **SETFRAME** are active during execution of the PostScript job.

option 2 applies VI Compose features at the end of the PostScript page execution. For this reason, **SETMEDIA** is not allowed with this option.

option 3 applies VI Compose features at the beginning of the PostScript page execution. **SETMEDIA** can be used. This option only supports PostScript files generated by some PostScript drivers. Apple LaserWriter NTX II or Adobe drivers are recommended.

In multi-up mode when a form is used, will be imaged on the next logical page following the last page if the sheet side is not entirely filled up.options 2 and 3 also update information pertaining to the context, such as page number, and front/back flag for **NEWFRONT/NEWBACK**.

There is no guarantee that options 2 and 3 will work with every PostScript file, due to the nature of PostScript, but they should work with most of them.

RUN and Demographics

To use the RUN command with the Demographics function, you can use option 2 or 3.

Run and Normalize

PostScript code called by RUN cannot be Normalized. For more information, refer to the *FreeFlow VI Compose User Guide*.

By using an empty reference, the RUN command can be placed after the VIPP® settings and before the beginning of a PostScript document, so that a separate VIPP® submission file is not required. When subsequent VIPP® code is appended after the PS code the **%%EOD_XGF** marker is placed between the end of the PS code and the VIPP® code in order to properly terminate the RUN command and resume regular VIPP® code execution.

Examples

This example illustrates an empty reference in the **RUN** command.

```
%!
XGF
(copy.frm) SETFORM
() 3 RUN
%!PS_Adobe-3.0
.....
```

Note: Inefficient PostScript code can affect performance.

Modes

This command is applicable in all modes.

Related commands

RESET, SETMPATH

RUNDD

RUNDD prints a document previously processed by the Decomposition Service in DocuPrint NPS or FreeFlow Print Server controllers.

Syntax

```
(document_name) RUNDD
(document_name) /ENDOFSET RUNDD
(document_name) /VARxyz RUNDD
```

Where:

(document_name)	is the name of a document previously processed by the decomposition service.
/ENDOFSET	applies the finishing previously defined by **SETFINISHING** to the document.
/VARxyz	refers to a procedure previously defined with **SETVAR** that will be executed at the beginning of the last page of the document.

RUNDD locates the initial page relative to the document in the libraries currently defined by **SETIPATH** or **SETFPATH**; then locates all consecutive pages in the same library associated with the document.

RUNDD can be combined with **SETFORM**, and **SETMEDIA**, usually with the cycle syntax of these commands, to map specific forms and media with specific pages of the document.

Use **SETFORM** or **ENDPAGE** to place static or variable data on every page of the decomposed document. **SETFORM** prints the data under the decomposed page (white areas are transparent), **ENDPAGE** prints them on top. To place specific data on specific pages, implement a counter and use IF or **CASE** statements in the form or **ENDPAGE** procedure. More complex layout coding can also be done using SETPAGEDEF.

RUNDD can be used with **SETCYCLECOPY/COLLATE_off**.

DocuPrint NPS and FreeFlow Print Server application compatibility

The **RUNDD** command will search in the **SETIPATH** libraries for TIFF images named according to this DocuPrint NPS Decomposition Services convention:

```
document_name.pnnnn.c.tif
```

Where:

document_name	is the name of the file submitted to Decomposition Services.
nnnn	is the four digit page number.
c	is either: **b** black **h** highlight

When TIFF images are not found, this FreeFlow Print Server Decomposition Services naming convention is used to search again:

```
document_name_dir/document_name.pnnnnnnnn.tif
```

Where:

document_name	is the name of the file submitted to Decomposition Services.
nnnnnnnn	is the eight-digit page number.
document_name_dir	is the name of the subdirectory containing the TIFF images.

In both cases, when TIFF images are found, **RUNDD** prints each of them on a separate page using the ICALL command.

DocuPrint NPS only

All `.b.tif` pages are printed black, 0 **setgray** is forced, and `.h.tif` images are overlaid on the black image using the color previously defined by **SETTXC**.

When TIFF images are not found in the prior steps, forms named according to this DocuPrint NPS Decomposition Services convention are searched in the **SETFPATH** libraries:

```
document_name.pnnnn.ps
```

Where:

document_name	is the name of the file submitted to Decomposition Services.
nnnn	is the four digit page number.

When forms are found, **RUNDD** prints each of them on a separate page using the exec PostScript operator. All form pages are printed with black and the current highlight color.

The forms option is supported only on DocuPrint NPS and is required when Byte Compressed **SaveMaskBC** decomposition is used.

FreeFlow Print Server only

When the default location for FreeFlow Print Server decomposition images is changed, the new location/path must be added to **SETIPATH** in xgfunix.run. DO NOT change or remove the original path to the default location.

Examples

This example prints the decomposed document doc1 using, in sequence, the set of forms and media defined by the **SETFORM** and **SETMEDIA** commands as follows:, doc1(page1)+form1+med1, doc1 (page2)+form2+med2, ...

```
[ (form1) (form2) .... (formn) ] SETFORM
[ (med1) (med2) .... (medn) ] SETMEDIA
(doc1) RUNDD
```

When using a decomposed document as a form, combine **RUNDD** with **SETFORM** to eliminate printing differences between the DocuPrint NPS family of printers, and FreeFlow Print Server controllers. Example of syntax (assuming `doc1.ps` is a one page document):

```
{ (doc1.ps) RUNDD } SETFORM
```

This solution allows decomposed forms to print the same across all three printer families.

This example prints This page belongs to name (field from the database file) on every page of the

decomposed document:

```
{
1275 50 MOVETO
(This page belongs to $$NAME. ) VSUB 1200 2 SHP
} SETFORM
(Doc1) RUNDD
```

This is an example of using **ENDPAGE** when printing `This page belongs to name` on the first three pages only.

```
{
1275 50 MOVETO
(This page belongs to $$NAME. ) VSUB 1200 2 SHP
} 3 ENDPAGE
(Doc1) RUNDD
```

The 3 is user-defined, and represents the number of pages in which the procedure remains active.

This example prints the Name and Address fields from the database file on page 2 and 5 of the decomposed document.

```
/VARpage (00) SETVAR
{ /VARpage ++
   IF VARpage (02) eq VARpage (05) eq or
     { 1275 50 MOVETO
        NAME  SHL
       ADDRESS SHL
     }
   ENDIF
} SETFORM
(Doc1) RUNDD
```

This is an example of printing a sequential page number on a decomposition page.

```
%!
XGF
DUPLEX_on
PORT
/VARPageCount (00001) SETVAR  %  Starting number for page.
/NHEB 10 SETFONT
{
1275 50 MOVETO
(Page $$VARPageCount.) VSUB 900 2 SHP
} SETFORM
```

This is an example of set functionality. This staples each set.

```
/Staple /ON SETFINISHING
(FirtsSection.ps) /ENDOFSET RUNDD
(SecondSection.ps) /ENDOFSET RUNDD
(ThirdSection.ps) /ENDOFSET RUNDD
```

This example will staple all sections together, however, the cover page is not stapled.

```
150 3000 MOVETO
(Cover Page) SH
PAGEBRK
/Staple /ON SETFINISHING
(FirtsSection.ps) RUNDD
(SecondSection.ps) RUNDD
(ThirdSection.ps) /ENDOFSET RUNDD
```

Modes

This command is applicable in native mode and database mode.

Related commands

- BCALL
- ENDIMP
- ENDOFRUN
- ENDOFSET
- SETFPATH
- SETIPATH
- RUNPDF
- RUNTIF

RUNPDF

RUNPDF prints a PDF document containing one or more pages and supports mixed page sizes and orientations. It prints each page of the PDF file on a separate logical page similar to **RUNDD**.

Syntax

```
(document1.pdf) RUNPDF
```

Note: On PostScript RIPs the PDF pages are rendered as a rectangle filled with the PDF name unless EPS information has been embedded in the PDF file contents. This is automatically performed when the PDF file is used as a resource in *VI Design Pro* or *VI Design Express*. On PDF RIPs VI eCompose, normalize this step is not required.

Modes

This command is applicable in all modes.

Related commands

- RUNDD
- RUNTIF
- BEGINIMP
- ENDIMP

RUNTIF

RUNTIF prints a document from a single TIFF file that contains one or more pages. This command prints each page of a TIFF file on a separate logical page.

Syntax

```
(document1.tif) RUNTIF
```

```
(document1.tif) startpage stoppage RUNTIF
```

Where:

startpage and stoppage	specify the range of pages to print.

Use the same techniques as described in the **RUNDD** command to apply media, forms and variable data on the pages of the TIFF document printed with **RUNTIF**.

Note: Use the **ENDPAGE** command, including the incrementation of a page count variable and a CASE statement referring to it, to place variable data on various pages of the document.

For example:

```
/VARpc 0 SETVAR
{  /VARpc ++
   CASE VARpc {}
   5  { .. code to place variable data on page 5 }
   13 { .. code to place variable data on page 13 }
   ENDCASE
} ENDPAGE
(mydoc.tif) RUNTIF
```

Built in variables are available in forms or **ENDPAGE** procedures when **RUNTIF** is used:

TIFPAGE	page number of the current page
TIFLAST	page number of the last page

Examples

Refer to RUNDD examples.

To print on the last page of the TIFF the following can be coded:

```
{ IF TIFPAGE TIFLAST eq { (last page) SH } ENDIF } ENDPAGE
(multipages.tif) RUNTIF
```

Modes

This command is applicable in native and line mode.

Related commands

- BCALL
- ENDIMP
- RUNDD
- RUNPDF

SAVEPP

SAVEPP saves the current secondary print position for later use by **HDISP**, **VDISP**, **SHPOS**, and **SVPOS**. There are many uses for this command, but the most common use of **SAVEPP** is to draw variable boxes or lines between variable length paragraphs.

Syntax

```
SAVEPP
```

Examples

Use this example to draw a line between paragraphs using **SAVEPP**.

```
150 3000 MOVETO
400 0 SHP   %This is a variable length paragraph which could have many lines
25 NL
SAVEPP
SHPOS SVPOS 400 0 S1 DRAWB
25 NL
400 0 SHP   %This is the next paragraph
```

To draw a box instead of a line, use the **VDISP** and **HDISP** commands.

Modes

This command is applicable in all modes.

Related commands

- HDISP
- SHPOS
- SVPOS
- VDISP

SCALL

SCALL images an object or segment on the current page. The origin (0,0) of the object or segment is placed at the secondary print position. An object or segment can be:

- A segment coded using VIPP® native mode or simple PostScript
- A TIFF, JPEG, PostScript, Encapsulated PostScript (EPS), or XObject file
- A single page PDF file.
- On PostScript RIPs, the PDF is rendered as a rectangle and filled with the PDF name unless EPS information has been embedded in the PDF file contents. This action is performed automatically performed when the PDF file is used as a resource in VI Design Pro or VI Design Express. On PDF RIPs such as VI eCompose or Normalizer, this step is not required.
- Syntax:
  - When calling a segment, all syntax variants shown can be used with or without the **CACHE** command.
  - When calling any other format such as TIFF, JPEG, EPS, PS, or PDF with the **CACHE** command, all syntax variants can be used.
  - When calling any other format such as TIFF, JPEG, EPS, PS, or PDF without the **CACHE** command, the three options scale, rotate, and align are mandatory.
- Location:
  - For any syntax, TIFF and JPEG are required in imglib with the **SETIPATH** command, or in project directories with the **SETPPATH** command.
  - For a segment without the **CACHE** command and without the align option, the segment is required in formlib with the **SETFPATH** command, or in project directories with the **SETPPATH** command.
  - In any other case, the location is required in imglib with the **SETIPATH** command, in formlib with the **SETFPATH** command, and in mislib with the **SETMPATH** command, or in project directories with the **SETPPATH** command.
- For information about image format support and limitations, refer to the **ICALL** command.

Syntax

```
(Segmentname) SCALL

(Segmentname) scale SCALL

(Segmentname) scale rotation1 SCALL

(Objectname) scale rotation1 align1 SCALL

(Objectname) [width height] rotation2 align2 SCALL

(Objectname) CACHE scale rotation1 align1 SCALL

(Objectname) CACHE [width height] rotation2 align2 SCALL

{ segment contents } SCALL

{ segment contents } scale SCALL

{ segment contents } scale rotation1 SCALL
```

Where:

Segmentname	is the name of a segment.
Objectname	is the name of a segment, TIFF, JPEG, PostScript, EPS file, PDF, or XObject.
scale	is either: A **single isomorphic scaling factor** (real number, can be negative) An **array** of the form:, [scaleH scaleV /A] **Where**: **scaleH scaleV** are 2 anamorphic scaling factors (real numbers, can be negative). **Negative numbers** are used for mirroring (either horizontally, vertically or both)
rotation1	is the rotation angle of the segment (counterclockwise) around its alignment point. Default is 0.
align1	is the alignment option defining which point of the image must be aligned on the secondary print position. (Use the BBOX marker for in-line segments.) It must be one of these: **0** top left **1** top right **2** top center **10** bottom left (default) **11** bottom right **12** bottom center **20** center left **21** center right **22** center center
width height	are the size, in current units, of a box into which the object will be forced to fit. This is called the fit-in-box option. The bottom left corner of the box is placed at the secondary print position.
rotation2	is the rotation of the object inside the box (only 0, 90, 180 and 270 are supported).

align2	indicates how the image will be aligned inside the box. Values are the same as align1. Three optional weights can be combined with align2 options: **+100** fill in box option. When this option is enabled the image is scaled isomorphically so that it fills the entire box. When the width/height ratios of the box and the image differ, the part of the image that falls outside of the box is cropped. In this case the alignment options are used to choose which part of the image (corners, left, right, top, bottom, or center) is kept visible in the box. **+200** stretch in box option. When this option is enabled the image is scaled anamorphically so that it fills the entire box. When the width/height ratios of the box and the image differ, the image is stretched either vertically or horizontally. In this case the alignment options are useless. **+300** place in box. When this option is enabled the image is not scaled. It is placed in the box according to the alignment options. The part of the image that falls outside of the box, if any, is cropped.
segment contents	represents a small segment built in the data stream. Such a segment, which does not reference an external file, is called an in-line segment. When using this syntax, do not use the **CACHE**. Segments must be coded using VIPP® native mode or simple PostScript and stored in one of the libraries defined by **SETFPATH**. Use of the .seg extension is recommended.

CACHE is optional after (segmentname) in the first three syntax examples. When the CACHE command is used with a segment you must code a **%%BoundingBox** statement at the beginning of the segment to define the size of the image and its position relative to its origin.

Inside a segment, print positions are relative to the origin. The initial print position is set by default to 0,0, that is the origin of the segment. In other words **0 0 MOVETO** is implicitly executed at the beginning of a segment.

Single page PostScript and EPS files created by document processing applications using common drivers, typically `.prn` files, as well as TIFF and JPEG files can be imaged by **SCALL**. These objects must be stored in a library referenced by the **SETFPATH**, **SETIPATH**, **SETMPATH** or **SETPPATH** commands.

Syntax for use with PDF files:

The following are the only supported syntaxes for use with PDF files:

```
(xyz.pdf) CACHE SCALL
```

```
(xyz.pdf) scale rotate align SCALL
```

For example:

```
(xyz).pdf SCALL
```

must be changed to:

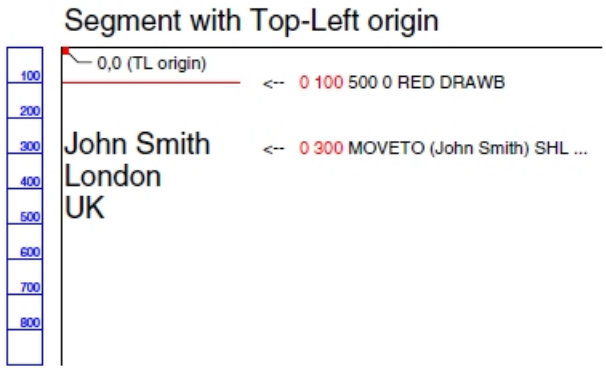
```
(xyz.pdf) 1 0 10 SCALL
```

Segment origin for segments coded using the VIPP® language:

The origin of all placements inside a segment coded with VIPP® is the secondary print position at the time **SCALL** is executed.

That position is then considered:

- bottom-left (vertical placements are measured upwards) by default or when **ORIBL** is coded
- top-left (vertical placements are measured downwards) when **ORITL** is coded



Bounding Box:

If **SCALL** is used in conjunction with **CACHE** the bounding box statement

```
(%%BoundingBox: llx lly urx ury)
```

is used to figure out the size and clipping of the segment image. **llx lly** is considered as the origin of the image and coincides with the secondary print position.

Examples

```
(dcxlogo.seg) SCALL
```

```
(doc1.eps) CACHE .6 SCALL
```

```
{ /NHE 18 SETFONT (XY Corp) SHc 25 NL (xy.tif) 1 0 2 ICALL } SCALL
```

This example uses the stretch in box option:

```
(picture1.jpg) CACHE [ 500 1000 ] 0 222 SCALL
```

In the following example...

```
(XObject) CACHE 1 90 0 SCALL
```

The advantage is efficiency and optimized productivity because **SCALL** no longer has to image the object.

Note: The JPEG/Exif marker contains information about the resolution that was ignored until FreeFlow version 4.0. In previous versions VIPP® may have defaulted to 300 dpi and possibly rendered the image at an incorrect size. Since the image will now be rendered at the correct size existing VIPP® jobs referencing JPEG files with Exif markers will experience a backward incompatibility issue. To fix this the scale option of the **SCALL** commands must be changed. When the change impacts production, the old behavior can be re-instantiated by temporarily adding the following statement in the JDT or in `/usr/xgf/src/xgf.def`: `[ /ProcessExif false ] SETPARAMS`.

Modes

This command is applicable in all modes.

Related commands

- ICALL
- CACHE
- FCALL
- MOVETO
- MOVEH
- SETFPATH
- SETIPATH
- SETMPATH

SETBAT

SETBAT defines a Background Attribute key. When activated by **SETTXB** or **INDEXBAT** a **BATkey** paints a background under all subsequent text imaged using the **SHx** commands. A text background may be made of any combination of lines and boxes (outlined or filled) called items. A collection of standard BATkeys is provided in /usr/xgf/src/xgf.bat and can be sampled by printing these files:

- `/usr/xgf/demo/sambat.ps` (for gray BATkeys)
- `/usr/xgf/demo/sambatr.ps` (for red BATkeys)
- `/usr/xgf/demo/sambatg.ps` (for green BATkeys)
- `/usr/xgf/demo/sambatb.ps` (for blue BATkeys)

Syntax

```
/BATkey
[Col_GEP LineW WVal WC HVal HC Xpos XC Ypos YC Corner CC UR
.......
] SETBAT
```

Where:

Col_GEP	is a Colorkey or a **GEPkey** as defined in xgf.gep.
LineW	is a percentage of the font height used to set the line width (thickness) for this item (for example, 0.06 with a font of 20 points will set a line width of 1.2 point). If zero, the line width set by the GEPkey will be used regardless of the font size.
WVal	is used to compute the width of the item in relation to the width code.
WC	is the width code: **0** WVal is a fixed value in current units **1** WVal is a percentage of the string height that will be added to the string width **2** WVal is a fixed value that will be added to the string width **3** WVal is a percentage of the maximum of the string width and string height
HVal	is used to compute the height of the item in relation to the height code.
HC	is the height code: **0** HVal is a fixed value in current units **1** HVal is a percentage of the string height **2** HVal is a fixed value that will be added to the string height **3** HVal is a percentage of the maximum of the string width and string height
Xpos	is used to compute the horizontal position of the item in relation to the X code.
XC	is the X code:

	0 Xpos is a fixed value in current units **1** Xpos is a percentage of the string height added to the horizontal string position **2** Xpos is a fixed value that will be added to the horizontal string position **3** Item is automatically centered horizontally on the text. In addition, Xpos is a percentage of the maximum of the string width and string height added to the horizontal item position
YC	is the Y code: **0** Ypos is a fixed value in current units **1** Ypos is a percentage of the string height added to the vertical string position **2** Ypos is a fixed value that will be added to the vertical string position **3** Item is automatically centered vertically on the text. In addition, Ypos is a percentage of the maximum of the string width and string height added to the vertical item position
Corner	is used to compute the corner radius of the item in relation to the corner code.
CC	is the corner code: **0** Corner is a fixed value in current units **1** Corner is a percentage of the string height **2** Corner is a fixed value that will be added to the string height **3** Corner is a percentage of the maximum of the string width and string height
UR	sets the string height: **0** string height is the height of the uppercase X **1** string height is the real height of the string (positions will adjust on ascenders and descenders)

Examples

This example defines a **BATkey** for underlining. The line will be black, have a thickness equal to 6 % of the font height, a width equal to the width of the text, and be placed at 25 % of the font height under the baseline.

```
/UNDL [ BLACK .06 0 2 0 0 0 2 -.25 1 0 0 0 ] SETBAT
```

Note: When the code is 0, WVal, HVal, Xpos, Ypos, and Corner can be specified as a variable (ex: */VAR* width). An internal variable (*/COLW)* is provided to represent the column width defined by **SETCOLWIDTH**.

Modes

This command is applicable in all modes.

Related commands

- SETCOLWIDTH
- SETPAT
- SETTXB
- SETTXC
- SHX

SETBFORM

SETBFORM selects a form for printing on the back sides of sheets without variable data for the current and all subsequent sheets. This command must be specified before any printing command on the sheet. Otherwise, this command produces the same results as **SETFORM**. Refer to **SETFORM** for syntax description and examples.

Note: **SETBFORM** triggers a back form on each page. The null **SETBFORM** command does not cancel the back form, instead, it triggers a blank back form on each page. If you want to cancel the back form, use `0 SETMAXBFORM`.

Syntax

```
Form_ref SETBFORM

Form_ref planenumber SETBFORM

Form_ref [ c1 c2 ... cn ] SETBFORM

Form_ref planenumber [ c1 c2 ... cn ] SETBFORM
```

An EPS, a JPEG, or a TIFF file can be used as a form by using the CACHE command after the resource name. Refer to **CACHE** for further information.

Modes

This command is applicable in all modes.

Related commands

- SETFORM
- SETFPATH
- SETMAXBFORM
- CACHE

SETBIDI

SETBIDI is used to configure the BIDI transform function. This command replaces the depreciated command, **SETARB**.

Syntax

```
ctx_table mrg_table (bi-directional index) (left-to-right index) SETBIDI
```

Where:

ctx_table	is a context analysis definition defined by **BEGINARBT/ENDARBT**
mrg_table	is a merge definition defined by **BEGINARBM/ENDARBM**
(bi-directional index)	is a font index, defined by **INDEXFONT**, to be used by BIDI processing with option 1
(left-to-right index)	is a font index, defined by **INDEXFONT**, to be used by BIDI processing with option 1

Examples

```
ARB1256_2 ARB1256_M1 (A) (E) SETBIDI
```

A default configuration is coded at the end of the bi-directional configuration file located at `xgf/src/arb.def`

Modes

This command is applicable in all modes.

Related commands

- BIDI
- BEGINARBT
- ENDARBT
- FCALL
- MOVETO
- SCALL
- ENDARBM

SETBUFSIZE

The **SETBUFSIZE** command sets the line or record buffer size for data files being processed by **STARTLM** or **STARTDBM**.

Syntax

```
size SETBUFSIZE

size (F) SETBUFSIZE
```

Where:

size	defines the maximum line length. Use this syntax to process files made of lines ending with LF, **CR**, or **CRLF** line delimiters.
size (F)	defines a fixed record length. Use this syntax to process files made of fixed length records without any line delimiters such as **LF**, **CR**, or **CRLF**.

The default value is defined in the `/usr/xgf/src/xgf.def` file.

Note: When the buffer size is too small, the job aborts with a range check error on read line.

Tip: Set the default value in the `/usr/xgf/src/xgf.def` file to the largest size required by the applications, for example, 3000 **SETBUFSIZE**. The default is 2000.

Modes

This command is applicable in line mode and database mode.

Related commands

None

SETCJKENCMAP

SETCJKENCMAP defines the table used to guess the encoding of a font from the font name when the font is selected with the **SETFONT** or **INDEXFONT** command. The **SETCJKENCMAP** command is used with multiple-byte fonts.

A default table is defined in the configuration file `xgf/src/cjk.def`.

Syntax

```
[ (guess string1) /encoding-name1 /wrapping-rules1 /v2h_cvname1
(guess string2) /encoding-name2 /wrapping-rules2 /v2h_cvname2
....
] switch SETCJKENCMAP
```

Where:

(guess stringX)

/encoding-nameX

The encoding to be associated with that font. Choose the encoding name from the following list:

- /ASCII
- /SJIS
- /EUC-JP
- /EUC-TW
- /GB18030
- /EUC-CN
- /EUC-KR
- /Johab
- /UHC
- /BIG5
- /GBK
- /EUC-KRpc
- /EUC-CNpc
- /BIG5pc
- /UTF8
- /UTF16
- /UTF32
- /UCS2
- /ISO-2022

wrapping-rulesX

The name of a wrapping table that is defined using the command **SETCJKRULES**. The name of the wrapping table is not restricted. A single table can be referenced by several encodings.

/v2h_cvnameX

The name of a conversion table that is defined using the command **SETV2HCONV**. The name of the conversion table is not restricted. A single table can be referenced by several encodings.

The table is used only with CJK fonts: /FontType=0. The table is not used for Roman fonts.

switch

Optional. If equal to /INI, a new table is created. When omitted, the entries are added on top of the current table.

Note:

- The **SETCJKENCMAP** command is intended for use in the CJK configuration file `xgf/src/cjk.def`. Although **SETCJKENCMAP** can be used directly in a VIPP® job, it is not recommended.
- The conversion table is searched top-down. The first matching entry is retained. For example, **GB-EUC** must be placed before **EUC**. The last entry in the table is the default, so the guess string for this entry must be empty.

Modes

This command is applicable in all modes.

Related commands

- SETFONT
- INDEXFONT
- SHP and SHp
- SHMF, SHMf, and SHmf

SETCJKRULES

SETCJKRULES defines the lists of characters that influences text wrapping with the **SHP** command for a given encoding family.

A default table is defined in the configuration file `xgf/src/cjk.def`.

Syntax

```
[ /wrapping-rules1
    [ opening_char_list1 ]
    [ closing_char_list1 ]
    [ punct_char_list1 ]
  /wrapping-rules2
    [ opening_char_list2 ]
    [ closing_char_list2 ]
    [ punct_char_list2 ]
  ....
] SETCJKRULES
```

Where:

/wrapping-rulesX	is an unrestricted name that can be used as the third parameter in an entry of the command **SETCJKENCMAP**.
opening_char_listX	is a list of strings representing the set of opening characters to be associated with a specific encoding.
closing_char_listX	is a list of strings representing the set of contiguous or closing characters to be associated with a specific encoding.
punct_char_listX	is a list of strings representing the set of punctuation characters to be associated with a specific encoding. Xerox recommends that the hexadecimal notation for these characters, a value enclosed in angled brackets be used in each list.

Note: This command is mainly intended for use in the CJK configuration file, `xgf/src/cjk.def`. Although it can be used directly in a VIPP® job, this is not recommended. For an example of the command, refer to the file `xgf/src/cjk.def`.

Examples

```
/SJIS
  % Opening
  [
    <816f> <8165> <8167> <8171> <8173> <a2> <8175> <8177> <8179>
    <816b> <8197> <40> <8198> <81a7> <8194> <23>
  ]
  % Contiguous/Closing
  [
    <21> <8149> <3f> <8148> <3a> <8146> <3b> <8147> <29> <816a>
    <5d> <816e> <7d> <8170> <818c> <8166> <8168> <818d> <8172>
    <8174> <a3> <8176> <8178> <817a> <816c>
   ]
   % Punctuation
   [ <8141> <8142> <2c> <8143> <2e> <8144> ]
/UTF8
  % Opening
  [
     <7b> <efbd9b> <e28098> <e2809c> <e38088> <e3808a> <efbda2>
     <e3808c> <e3808e> <e38090> <e38094> <efbca0> <40> <c2a7>
  ]
  % Contiguous/Closing
  [
      <21> <efbc81> <3f> <efbc9f> <3a> <efbc9a> <3b> <efbc9b> <29>
      <efbc89> <5d> <efbcbd> <7d> <efbd9d> <e280b2> <e28099> <e2809d>
      <e280b3> <e38089> <e3808b> <efbda3> <e3808d> <e3808f> <e38091>
  ]
  % Punctuation
  [ <e38081> <e38082> <2c> <efbc8c> <2e> <efbc8e> ]
] SETCJKRULES
```

Modes

This command is applicable in all modes.

Related commands

SETFONT, INDEXFONT, SHP and SHp

SETCOL

The **SETCOL** command assigns a name referred to as a **Colorkey**, to a color definition. A color definition can be one of the following types:

- Grayscale
- RGB
- CMYK
- Gradient a set of up to three colors that are applied gradually to a graphic element such as a box, circle, polygon, or test
- CMYK separation color space, a subset of CMYK components

Syntax

```
/Colorkey [c m y k] SETCOL      % for CMYK color definition

/Colorkey [r g b] SETCOL        % for RGB color definition

/Colorkey g SETCOL              % for grayscale color definition

/Colorkey [ color1 color2 color3 dispcode tint direction ] SETCOL
                                  % for gradient color definition

/Colorkey [ c /SCS_C ] SETCOL

/Colorkey [ m /SCS_M ] SETCOL

/Colorkey [ y /SCS_Y ] SETCOL

/Colorkey [ k /SCS_K ] SETCOL

/Colorkey [ c m /SCS_CM ] SETCOL

/Colorkey [ c y /SCS_CY ] SETCOL

/Colorkey [ c k /SCS_CK ] SETCOL

/Colorkey [ m y /SCS_MY ] SETCOL

/Colorkey [ m k /SCS_MK ] SETCOL

/Colorkey [ y k /SCS_YK ] SETCOL

/Colorkey [ c m y /SCS_CMY ] SETCOL

/Colorkey [ c m k /SCS_CMK ] SETCOL

/Colorkey [ c y k /SCS_CYK ] SETCOL

/Colorkey [ m y k /SCS_MYK ] SETCOL

(spotcolorname) [ c m y k ] SETCOL              % spot color definition

/Colorkey [ v /SCS_V ] SETCOL                   % standalone Clear Dry Ink

/Colorkey [ c m y k v /SCS_CMYKV ] SETCOL       % Clear Dry Ink combined
                                                % with a CMYK color
```

```
/Colorkey [ (spotcolorname) v /SCS_SV ] SETCOL  % Clear Dry Ink combined
                                                % with a spot color

/Colorkey [ c m y k o /SCS_CMYKO ] SETCOL       % color space key to support Orange 5th color
                                                %Extended Gamut combined with CMYK
/Colorkey [ c m y k b /SCS_CMYKB ] SETCOL       % color space key to support Blue 5th color
                                                % Extended Gamut combined with CMYK

/Colorkey [ c m y k g /SCS_CMYKG ] SETCOL       % color space key to support Green 5th color
                                                % Extended Gamut combined with CMYK

/Colorkey [ v  /SCS_MICR ] SETCOL               % color space key to support the "Xerox MICR"
                                                % spot color.
```

Where:

/Colorkey	is the name of the **Colorkey**. The name is user definable, and must not be a reserved word.
[c m y k]	is a CMYK cyan, magenta, yellow, black, color definition. Four real values are required within the array Square braces. Each value must be in the range of 0–1.
[r g b]	is a RGB red, green, blue color definition. Three real values are required within the array Square braces. Each value must be in the range of 0–1.
g	is a grey level color definition. It is a real value in the range 0–1. 0 is black, 1 is white.
v	is an integer between 0 and 1 representing the Xerox Specialty Inks coverage from 0–100 %.
color1/color2/color3	are 3 distinct Color keys that are used for painting by going gradually from one color to the other.
dispcode	is the dispatch code, which is a 3 digit code defining how the graduation is spread in the fill. Each digit is a weight ranging from 0 or 1–9. The **first digit** (1–9) defines the weight of the graduation from color 1 to color 2. The **second digit** (0–9) defines how long color 2 is maintained unchanged in the middle, useful for border effects. The **last digit** (0–9) defines the weight of the graduation from color 2 to color 3. When it is equal to zero only colors 1 and 2 are used.
tint	is a number ranging from 0–2 used to apply a common tint to the entire graduation: **0–1** light tints from white to plain colors **1** plain color. **1–2** dark tints from plain colors to black

direction	is one of: **/H** the graduation is applied horizontally **/V** the graduation is applied vertically
spotcolorname	is the name of a spot color defined at the printer DFE. The associated CMYK values in the **SETCOL** statement are only used when no spot color with that name has been defined at the device. Such a color is not impacted by the color management process possibly active on the target RIP. When spotcolorname is combined with Xerox Specialty Inks it must have been previously defined with a spot color **SETCOL** statement.

A gradient filled color can be used as **FillColorkey** to define a **GEPkey** or as a /ColorTable parameter for **DRAWBAR** or **DRAWPIE**.

Also a Gradient Fill can replace a **GEPkey** as a parameter for a **DRAWBx**, **DRAWPOL** or **SHx** command.

Note: The following pre-defined color 100 % standalone Xerox Specialty Inks has been added to `xgf/src/ xgf.gep`:

`/CLEAR [ 1 /SCS_V ] SETCOL`. A new parameter **/ClearSubst** has been added to set the rendering behavior on platforms that do not have Xerox Specialty Inks toner. Refer to the Parameters section for details.

Limitations

Gradient Fill definitions do not apply everywhere a color or **GEPkey** can be specified. In particular, they cannot be applied to **SETTXC**, INDEXCOLOR, **SETPIF/INDEXPIF**, **SETFRAME**, **SETZEBRA**, **MOVEH** and **RPE**. They cannot be combined with patterns.

Examples

```
/RB_BLUE1 [ DBLUE MBLUE LBLUE 252 1 /V ] SETCOL

/RB_GREEN1 [ DGREEN LGREEN DGREEN 505 1 /V ] SETCOL

/RB_RED1 [ DRED RED LRED 282 .7 /V ] SETCOL

/RB_BLUE_RED1 [ MBLUE LBLUE MRED 904 1 /V ] SETCOL

/GEP1 0 WHITE 0 RB_BLUE1 SETGEP
228 3124 2061 150 GEP1 50 DRAWBR

/GEP1 0 WHITE 0 RB_GREEN1 SETGEP
228 2836 2061 150 GEP1 DRAWB

228 2561 2061 150 RB_RED1 DRAWB

/GEP1 5 GREEN 0 RB_BLUE_RED1 SETGEP
228 2252 1039 600 GEP1 DRAWC

/NAGD 85 120 SETFONT
(Sample Text) 0 GEP1 0 SHX

/CLEAR50  [ .5 /SCS_V ] SETCOL

/CYANCLEAR [ 1 0 0 0 1 /SCS_V ] SETCOL

/PAN200CLEAR [ (PANTONE 200 CS) 1 /SCS_SV ] SETCOL
```

Modes

This command is applicable in all modes.

Related commands

- SETTXC
- INDEXCOLOR
- SETGEP
- SETTRAN

SETCOLWIDTH

The **SETCOLWIDTH** command sets the column width for use by subsequent commands with a justify option.

Syntax

```
colwidth SETCOLWIDTH
```

Where:

colwidth	is the column width in current units.

The default value is defined in the `/usr/xgf/src/xgf.def` file.

Modes

This command is applicable in all modes.

Related commands

- COLW
- SHJ and SHj
- SHMF, SHMf, and SHmf
- SHP and SHp
- SETLKF

SETCYCLECOPY

The **SETCYCLECOPY** command sets the number of cycle copies. Cycle copy is an alternative to the standard multiple copy mechanism that allows copy range specification for forms, media, and RPE items as described in the related commands.

Syntax

```
copynumber SETCYCLECOPY
```

Examples

This example prints three copies of each page, using FORM_A on copies 1 and 2 and FORM_B on copy 3.

```
COLLATE_off

3 SETCYCLECOPY

(FORM_A.frm) [1 2] SETFORM

(FORM_B.frm) [3] SETFORM
```

- Data file is spooled temporarily: On decentralized printers, when the collate mode **COLLATE_on** is used, and when no local disk is available, the data file is spooled temporarily on the local disk or in memory.
  - When the data file is spooled to disk, the **SPOOLNAME** command is used to define the path to the temporary spool file that is in the default `/usr/xgf/src/xgfunix.run` or `x:\xgf\src\xgfdos.run` location.
  - When the data file is spooled to memory, when the memory is exhausted, large jobs can abort. The memory limit is likely to be 1–3 Mbytes, depending on the amount of memory available.
- Collation command placement: Code any collation command before the **SETCYCLECOPY** command.

Place a **SETMAXCOPY** command at the beginning of the JDT or XJT to specify the maximum number of copies when **SETCYCLECOPY** is used to dynamically change the copy number in:

Line mode	either using NMPs or within a **BEGINPAGE** or **PROCESSDJDE** procedure
Database mode	inside a DBM
XML mode	inside a BTA

This initializes the memory structures and forces data file spooling in advance on desktop printers. In these situations, **SETCYCLECOPY** executes an implicit **CHKPOINT**, do not code **CHKPOINT**.

Note: Do not use a **REPEAT** or **RSAVE** command after the **SETCYCLECOPY** command, because unpredictable results can occur.

Tip: Combine this command with others like **DUPLEX_on** or **SETMULTIUP** to perform sophisticated multicopy functions.

When **SETCYCLECOPY** is used in native mode with **COLLATE_on**, the **CHKPOINT** command is mandatory at the end of the job.

If you ever use **SETCYCLECOPY** in a JDT in database mode it is mandatory to place it as the last command of the JDT. This is to ensure correct behavior on DFEs that do not have repositionnable

input stream.

This command does not restore variables before each copy pass. Because the values for each variable remain unchanged from one pass to the next, the user is responsible for re-initializing them as needed.

Modes

This command is applicable in all modes.

Related commands

- CHKPOINT
- COLLATE_dbm
- COLLATE_off
- COLLATE_on
- COPYRANGE
- SETBFORM
- SETDLFILE
- SETFORM
- SETMEDIA
- SPOOLNAME
- REPEAT

SETDATE

SETDATE temporarily sets the date variables to the supplied date. It is used in conjunction with **GETDATE**, **SHIFTDATE** and DAYS to compute various date offsets.

Syntax

```
[ YYYY MO DD ] SETDATE

[ YYYY MO DD HH MM SS ] SETDATE

days SETDATE
```

Where:

YYYY	is the year (>1970). It must be an integer.
MO	is the month (1–12). It must be an integer.
DD	is the day (1–31). It must be an integer.
HH	is the hours (0–23). It must be an integer.
MM	is the minutes (0–59). It must be an integer.
SS	is the seconds (0–59). It must be an integer.
days	represents a date as the number of days since 1.1.1970. It is an integer.

The date variables are reset to the day date by a page initialization or an explicit **GETDATE**. For this reason it must be placed as close as possible to the command using the date variables or the variables must be captured in a string using **VSUB** and **SETVAR** just after **SETDATE**.

With the first and 3rd syntax the current time is left unchanged.

Examples

This example assumes Year, Month, Day are fields from a DBF file.

```
[ Year Month Day ] SETDATE
/VARdate1 ($$D_DWL.  $$D_MO./$$D_DD./$$D_YYYY.) VSUB SETVAR
/VARstart [ Year Month Day ] DAYS SETVAR
VARstart'+'55 SETDATE
/VARdate2 ($$D_DWL. $$D_MO./$$D_DD./$$D_YYYY.) VSUB SETVAR
GETDATE
(From $$VARdate1. to $$VARdate2. you will be given the opportunity to visit our new shopping
center and purchase any article with a 50% discount.) VSUB 0 SHP
```

Modes

This command is applicable in all modes.

Related commands

GETDATE, SHIFTDATE, DAYS

SETDBSEP

SETDBSEP sets the field separator for use in database mode, or with RPE entries using the /FN option. Subsequent **STARTDBM** commands use this field separator to scan fields in a database file. For more information refer to VIPP® data streams, in the *FreeFlow VI Compose User Guide*. The default value : is defined in the `/usr/xgf/src/xgf.def` file.

To change the separator value either change this file, which applies the new separator to all applications, or use the **SETDBSEP** command in the JDT or before the **STARTDBM** command.

Syntax

```
(field separator) SETDBSEP
```

Where:

Field separator	is an alphanumeric string of one or more characters.

Examples

This example defines the : as the field separator.

```
(:) SETDBSEP
```

Hex or octal values can be used to define the field separator. Use this example to set the tab character as the field separator.

```
<09> SETDBSEP % Hex value
```

```
(\011) SETDBSEP % Octal value
```

Important: Never place this command in the Data Base Master file.

Modes

This command is applicable in line mode and database mode.

Related commands

- GETINTV
- STARTDBM
- FROMLINE
- RPEKEY

SETDLFILE

SETDLFILE activates multiple copy mode with a distribution list for the current job.

As an alternative to **SETCYCLECOPY**, multiple copies of a document can be obtained with the Distribution list feature. This is known as Set labeling.

SETDLFILE associates a distribution list and a JDT to produce cover pages in front of each document set. These pages are created by processing the Distribution List (DL) file with the associated JDT in the same manner as **STARTLM** processes them. Thus, the number of copies produced is equal to the number of pages of the DL file. In this case, **SETCYCLECOPY** cannot be used.

SETDLFILE is used in a native mode file, in a JDT, or in a submission file.

The DL file must be located in one of the libraries referenced by SETMPATH.

Syntax

```
(DL file name) (jdtname) SETDLFILE
```

Examples

This example prints the file `dist23.lst` using the JDT **dist.jdt**. The file `report57.dat` is printed between each cover page using the JDT report.jdt. The JDT `dist.jdt` contains a form call and page layout for the cover pages.

```
%!
(dist23.lst) (dist.jdt) SETDLFILE
(report57.dat) SETLMFILE
(report.jdt) STARTLM
```

Modes

This command is applicable in native mode and line mode.

Related commands

- SETCYCLECOPY
- SETJPATH
- SETMPATH
- STARTLM

SETENCODING

SETENCODING is used to re-encode a base PostScript font through the association of a base font and an encoding table.

A base PostScript font is essentially a collection of characters, the character set referenced by names, the character name or glyph name. In font terminology, a base font is any PostScript font with a type 1, for example, Times Roman, Courier, Helvetica, and Symbol, type 3 user-defined fonts or type 42 (TTF) format.

Textual data is a stream of character codes according to a specific encoding, for example, ASCII, ISO 8859-1, UTF8, and so on. A data stream can be single-byte for example, ISO 8859-1 or multi-byte, for example, UTF8. Base fonts, which are single-byte fonts, process single-byte data; multi-byte fonts, for example composite and CID-keyed process multi-byte data.

An encoding table is a file used to map a list of character codes to their corresponding character names for a given encoding. A base PostScript font contains a default encoding table, generally StandardEncoding. Re-encoding is a process to substitute the default encoding table with one that matches the data stream to be printed. When **SETENCODING** is used to re-encode a base font, a new font is created that is a copy of the base font but with a different encoding table.

Re-encoding a base font can be necessary when an invalid character is displayed or printed in place of the character you were expecting.

Note: An invalid character is likely caused when one of the following occurs:

- A character in your document is not included in the character set of the current font. The current font is the font that is specified through the **SETFONT** or **INDEXFONT** command.
- The character code in your document maps to the wrong character name in the current font.
- The character code in your document maps to a character name that does not exist in the current font.

To re-encode a base font, place one of the following **SETENCODING** commands in your job submission or resource file. Then, refer to the new, re-encoded font instead of the base font in a subsequent **SETFONT** or **INDEXFONT** command.

Note:

- For re-encoding that is performed at the first **SETFONT** or **INDEXFONT** command, font re-encoding does not occur for all fonts at once at the first occurrence of **SETENCODING**. Instead, font re-encoding is performed only at the first occurrence of the **SETFONT** or **INDEXFONT** command. This action saves VI Compose start-up time and memory consumption.
- For multiple-byte fonts and the **SETENCODING** base font list, **SETENCODING** applies only to base, single-byte fonts. Multiple-byte fonts cannot appear in a **SETENCODING** base font list because multiple—byte fonts use their own encoding mechanism, for example, PostScript CMap resources. To use a multiple-byte font, specify the PostScript font name with the **SETFONT** or **INDEXFONT** command.

Syntax

```
(font list file) (encoding table) SETENCODING

[font_entry_1, font_entry_2, …, font_entry_N ] (encoding table) SETENCODING
```

Where:

font list file	is a file that contains a list of font entries as described below.
font_entry_X	is a statement in any of the following four formats: /new_font_name /PS_font_name /new_font_name (font_file_name) /new_font_name [/PS_font_name (AFM_file_name)] /new_font_name [(font_file_name) (AFM_file_name)] Where: **new_font_name** is the new name for the re-encoded font. **PS_font_name** is the name of the original resident PostScript font to be re-encoded. **font_file_name** is the name of the file containing the font to be re-encoded. When the syntax with font_file_name is used, the font is processed as a VIPP® resource by VI Compose and does not need to be installed as a printer resident font. As such, the font are embedded in any container, VPC, ProofPrint container, and so on produced by the various Variable Information Suite tools. The font file must be stored in one of the libraries referenced by SETMPATH or SETPPATH in project mode. Only pfa and pfb font formats are supported. The original PostScript font name does not need to be specified; VI Compose extracts it from the font file. **AFM_file_name** is an Adobe Font Metrics (AFM) file that provides kerning information. See more on kerning below.
encoding table	is the name of a file that contains a list of character code and character name pairs. An encoding table can contain only characters, name/code pairs that require re-encoding. Characters not present in the encoding table, are copied from the PostScript ASCII encoding table (StandardEncoding). Encoding table can be replaced by null. In that case, the default encoding table is preserved and the purpose of re-encoding is only to provide a shorter font name for convenience. The character names in an encoding table must exist in the base font that is being re-encoded. If a matching name cannot be found, select a different base font. The character codes, however, can be either single- or multi-byte values. So although SETENCODING only applies to single-byte, base fonts, the re-encoded font can be a base font if the encoding table only contains single-byte character codes or a composite font if the encoding table contains multi-bytes character codes.

In addition /STARTFF and /ENDFF can be used to encapsulate up to 4 font entries to make up a font family:

```
/STARTFF
   font_entry_reguar            (for regular font)
   font_entry_bold              (for bold font)
   font_entry_italic            (for italic font)
   font_entry_boldItalic        (for boldItalic font)
/ENDFF
```

Where:

STARTFF	marks the beginning of a font family
ENDFF	marks the end of a font family

Kerning refers to the ability to adjust the amount of space between characters when imaging a block of text with a given font. When kerning is required, create a font entry using the AFM_file_name format, to establish a link between a given font and an AFM file and use the **SETKERN** or **INDEXKERN** commands.

VI Compose is delivered with a collection of generic kerning files (.afk) which are a subset of AFM files. These files contain generic kerning information that can be used when an AFM file for a given font is not available. The AFK files only exist in the `xgf/encoding` directory and are referenced in `xgf/encoding/fontlist` in case a user activates kerning with one of the VIPP® fonts listed there. There are two sets of default files, one for serif fonts (genericsr*) and one for sans serif fonts (genericss*). AFK files can be used with any font. Choose one of them depending on the type, serif or sans serif and attribute of the font, regular, bold, oblique (italic), bold-oblique. However, because they are generic, the result cannot be 100 % accurate. For accurate results the original AFM file provided by Adobe must be used.

For additional information on kerning refer to Kerning.

Below is an example of using the AFK files with the Helvetica sans serif font family:

```
/STARTFF
   /NHE[/Helvetica (genericss.afk)]
   /NHEB[/Helvetica-Bold (genericss_bo.afk)]
   /NHEO[/Helvetica-Oblique (genericss_it.afk)]
   /NHEBO[/Helvetica-BoldOblique (genericss_bi.afk)]
/ENDFF
```

Note: Multi-byte fonts and SETENCODING base font list

SETENCODING only applies to base, single-byte fonts. Multi-byte fonts must never appear in a **SETENCODING** base font list because they use their own encoding mechanism, for example, PostScript CMap resources. To use a multi-byte font, specify its PostScript font name directly with the **SETFONT** or **INDEXFONT** command.

Placement of consecutive character names

Consecutive character names can be placed after the initial character code.

For example

```
16#80  /Adieresis  /Aring  /Ccedilla
is equivalent to:
16#80  /Adieresis
16#81  /Aring
16#82  /Ccedilla
```

For re-encoding single byte data in a base font, use character codes that range from 0 to 255 (16#00 - 16#FF).

For example:

```
16#80  /Adieresis      /Aring          /Ccedilla       /Eacute
16#84  /Ntilde         /Odieresis      /Udieresis      /aacute
16#88  /agrave         /acircumflex    /adieresis      /atilde
```

For adding multi-byte UTF-8 data to a re-encoded font, use character codes that fall into one of the UTF-8 multi-byte ranges:

1 byte	16#00 - 16#7F
2 bytes	16#C080 - 16#DFBF
3 bytes	16#E08080 - 16#EFBFBF
4 bytes	<F0808080> - <F7BFBFBF>

For example:

```
16#27       /quotesingle
16#C2A1     /exclamdown /cent /sterling /currency /yen /brokenbar
16#E28098   /quoteleft /quoteright /quotesinglbase /quotereversed
```

Note that character codes in the 4-bytes range must be expressed as hexadecimal strings. For example:

```
<F090A080> /cypriot_syllable_A
```

It should be noted that the UTF-8 multi-byte ranges contain thousands of characters and only those character codes that can be mapped to a character name in the base font, generally using the Standard Roman Character Set l be imaged. Other character codes are default to the question mark (?).

A predefined UTF-8 encoding table is provided in xgf/src/encoding/utf8. Below is an example of how to use it:

```
[ /CR-UTF8  /Courier
  /HE-UTF8  /Helvetica
  /TM-UTF8  /Times-Roman ] (utf8) SETENCODING
```

Usage of -UTF8 in the new font name is not mandatory but highly recommended in order to benefit from character boundary recognition in VIPP® commands related to strings, **RPE/GETFIELD/ SETRCD/SETPCD/GETINTV**. This is because -UTF8 is registered as a guess string in cjk.def.

For additional information on encoding tables and font lists refer to Standard lists, tables, keys, and attributes and VIPP® resources in the *FreeFlow VI Compose User Guide*.

The `/usr/xgf/src/xgf.def` file contains two **SETENCODING** statements that provide a predefined list of re-encoded fonts.

Examples

```
(fontlist) (sun8) SETENCODING

[ /EHE /Helvetica ] (ebcdic) SETENCODING

[ /CD128 (mb034.pfb) ] null SETENCODING

[ /FN115F (fn115f.pfa) ] null SETENCODING
```

Modes

This command is applicable in all modes.

Related commands

- INDEXFONT
- SETEPATH
- SETFONT
- SETMPATH
- SETPPATH

SETEPATH

SETEPATH defines a library or a list of libraries for font lists and encoding tables. The specified libraries are used by **SETENCODING** to locate font lists and encoding tables. The default is defined in the file `/usr/xgf/src/xgfunix.run` or `x:xgf\src\xgfdos.run`.

Use **SETEPATH** commands only in the xgfunix.run or xgfdos.run files. Adding any **SETEPATH** to a VIPP® job compromises portability. Adding any **SETEPATH** to a VI Project compromises both portability and project organization.

Syntax

```
(path to enc. library) SETEPATH
```

```
[ (path to enc. library 1) (path to enc. library 2) ... ] SETEPATH
```

When a list of libraries is specified, as in the second syntax above, they are searched in the order in which they appear in the list.

Modes

This command is applicable in all modes.

Related commands

SETENCODING

SETFINISHING

Use the **SETFINISHING** command to select a finishing feature and an associated finishing option for the entire job, or for subsets of the job on those devices supporting subset finishing.

The **SETFINISHING** command consolidates the old and new finishing-related commands supported by Xerox printers into one unified command.

SETFINISHING commands are supported on Xerox production devices only. Limited support may be available when using non-Xerox devices. Validate that the device can support the **SETFINISHING** commands. **SETFINISHING** supersedes the following legacy commands, which are now deprecated, although still supported for backward compatibility:

- **STAPLE_on/off**
- **STAPLEDETAILS**
- **OFFSET_on/off**
- **BIND_on/off**
- **BINDDETAILS**

Syntax

```
/Feature /ON SETFINISHING
/Feature /OFF SETFINISHING
/Feature (option-spec) SETFINISHING
```

Note: The **SETFINISHING** command must be inserted at the beginning of the page before any marking command.

Feature	is the name of a finishing feature applicable to the finishing device of the printer. The finishing features that can be enabled by **SETFINISHING** are **/Staple** **/Offset** **/Bind** **/Fold** **/MakeBooklet** **/Punch**
/ON	enables the finishing feature on the finishing device using the default or last selected option.
/OFF	disables the finishing feature on the finishing device.

(option-spec) enables the finishing feature on the finishing device using the specified option. Use of this syntax implicitly turns on the feature. You do not have to code the /ON option.

(option-spec) can be coded to include several sub-actions separated by a colon, as shown here:

```
/Punch (TopPortrait:Two) SETFINISHING.
```

Any omitted sub-option will default to either the system default or last specified sub-option.

Note: When the feature option-spec is enabled, you can disable the feature. Use the /OFF or /ON syntax as shown in the example.

The table that follows consists of examples of available features and options per device for several printers, use them as a reference for inclusion as (option-spec) values. Also refer to the appropriate device manual for specific options not listed in this table.

Feature		Option-spec	61xx	DP 65/75/90 2045/2060 12/5252/6060/8000	Nu-vera	FFPS 4110, 4112, 700 and 770	All FFPS (Finishing options dependant on Print device)	EFI EXP4110
Bind		LeftPortrait	X					
		RightPortrait	X					
		Longedge						
		Shortedge						
Offset		Offset	X	X	X			
Fold	**FoldEdge**	CFoldOutside				X	X	X
		CFoldInside			X	X	X	X
		ZFoldOutside				X	X	X
		ZFoldInside			X	X	X	X
		ZFoldRightHalf				X	X	X
		HalfFoldOutside				X		X
		HalfFoldInside				X		X
		BookletSquare-FoldAndTrim*				X		
		BookletSaddleSquareFoldAndTrim*				X		
		CFold			X	X	X	X

Feature		Option-spec	61xx	DP 65/75/90 2045/2060 12/5252/6060/8000	Nu-vera	FFPS 4110, 4112, 700 and 770	All FFPS (Finish-ing options depend-ant on Print device)	EFI EXP4110
		ZFold			X	X	X	X
		BiFoldInsi-deAndTrim*				X		
	Trim-Posi-tion	a real value in points				X		
	Squar-eFol-dLevel	Weak2				X		
		Weak1				X		
		Normal				X		
		Strong1				X		
		Strong2				X		
		Off				X		
		* TrimPosition and SquareFoldLevel features are only applicable to 3 FoldEdge Option-spec choices: BookletSquareFoldAndTrim, BookletSaddleSquareFoldAndTrim, BiFoldInsideAndTrim.						
MakeBooklet		BookletFold			X	X	X	X
		BookletSad-dleStitch			X	X	X	X
Staple		SinglePortrait	X	X	X	X	X	X
		RightPortrait	X	X	X	X	X	X
		BottomLeftPor-trait				X	X	X
		BottomRight-Portrait				X	X	X
		SingleLand-scape	X	X	X	X		X
		RightLandscape	X	X	X	X	X	X
		BottomLeft-Landscape				X	X	X
		BottomRight-Landscape				X	X	X
		DualLeftPortrait	X	X	X	X	X	X

Feature		Option-spec	61xx	DP 65/75/90 2045/2060 12/5252/6060/8000	Nu-vera	FFPS 4110, 4112, 700 and 770	All FFPS (Finishing options dependant on Print device)	EFI EXP4110
		DualRightPortrait	X	X	X	X	X	X
		DualTopPortrait			X	X	X	X
		DualBottomPortrait			X	X	X	X
		DualLeftLandscape			X	X	X	X
		DualRightLandscape			X	X	X	X
		DualLandscape	X	X	X	X	X	X
		DualBottomLandscape					X	X
		CenterLeftPortrait				X		X
		CenterRightPortrait				X		X
		CenterTopPortrait				X		X
		CenterBottomPortrait				X		X
		CenterLeftLandscape				X		X
		CenterRightLandscape				X		X
		CenterTopLandscape				X		X
		CenterBottomLandscape				X		X
Punch	**Location**	TopPortrait				X	X	X
		BottomPortrait				X	X	X
		RightPortrait				X	X	X
		LeftPortrait				X	X	X
		TopLandscape				X	X	X

Feature		Option-spec	61xx	DP 65/75/ 90 2045/ 2060 12/ 5252/ 6060/8000	Nu-vera	FFPS 4110, 4112, 700 and 770	All FFPS (Finish-ing options depend-ant on Print device)	EFI EXP4110
		BottomLand-scape				X	X	X
		RightLandscape				X	X	X
		LeftLandscape				X	X	X
	Num-ber of Holes	Two				X	X	X
		Three*				X	X	X
		Four*				X	X	X
						X	X	X
		* Only one pair can be configured: Two and Three, or Two and Four.						

Note: **SETFINISHING** activates finishing that is available on the device. For example, when the device does not have an in-line staple option, the staple command will be ignored. External finishing devices cannot be controlled using VIPP® code. The only option available is to job the tray at the end of the set. When the finisher can detect the tray job, then this is an available option to use. When the job is of a constant number of pages, the external finisher can often be set to fire at a given page count.

Check your PostScript Addendum Guide for your device, or with your local Xerox analyst, to confirm which finishing options your device supports based on its configuration.

Examples

The code in this example turns stapling on for the first and last sets, and off for all other sets in a multi-set job.

```
/Staple  (SinglePortrait)  SETFINISHING
/Staple /ON SETFINISHING
STARTOFSET
. . . Pages of first set
PAGEBRK
/Staple  /OFF  SETFINISHING
STARTOFSET
. . . Pages of second set
PAGEBRK
STARTOFSET
. . . Pages of third and subsequent sets up to last set
PAGEBRK
/Staple /ON SETFINISHING
STARTOFSET
. . . Pages of last set
PAGEBRK
%%EOF
```

On FreeFlow Print Server this command only addresses the internal stapling station. To set an external finisher as a subset finishing destination use the **SETOBIN** command, for example, (SBM) **SETOBIN**.

Examples using **FoldEdge** with **TrimPosition** and **SquareFoldEdge**:

- `/Fold (BookletSquareFoldAndTrim:13.2:Strong1) SETFINISHING`
- `/Fold (BiFoldInsideAndTrim:13.2) SETFINISHING`

Modes

This command is applicable in all modes.

Related commands

- STARTBOOKLET
- ENDBOOKLET
- STARTOFSET
- ENDOFSET

SETFONT

SETFONT selects and scales a font. All data following this command is printed with the selected font until a new **SETFONT** command is encountered or a font index defined by **INDEXFONT** is invoked.

For more information, refer to Kerning

Syntax

```
/Fontname size SETFONT
/Fontname sizeX sizeY SETFONT
/glossfont_name GLT SETFONT
/microfont_name MPR SETFONT
```

Where:

Fontname	is one of the following: The name of a font chosen from the VIPP® font lists enabled by **SETENCODING** in the `/usr/xgf/src/xgf.def` file. For more information, refer to Standard lists, tables, keys, and attributes in the *FreeFlow VI Compose User Guide*. A special Fontname defined to facilitate changing the font face to a different member of the same font family without having to re specify the Fontname: **/~REG** for regular **/~BLD** for bold **/~ITL** for italic **/~BDI** for bold-italic **/~CUR** for current font For example, when the current font is Helvetica 12 point as set by /NHE 12 SETFONT, a **SETFONT** specifying the special Fontname /~BLD changes the face to Helvetica-Bold. When the current font is Times Italic, the same **SETFONT** definition changes the face to Times Bold. The special Fontname /~CUR can be used to change the size of a font but keep the current Fontname unchanged. Specifying a font size as null keeps the point size unchanged. For more information, refer to Applying Attributes to Fonts.
size	An **integer or real number** specifying the font size in units of 1/72 inches points. When a size of 0 is given, the font is scaled automatically according to the margins and grid defined by **SETMARGIN** and **SETGRID**. In this case, use a fixed font, for example, Courier. **null**, that specifies that the current point size is kept unchanged.
sizex	allows specification of the font size in the X direction in points. The values are the same as defined in size, however, a value of 0 is not allowed.
sizey	allows specification of the font size in the Y direction in points. The values are the same as defined in size, however, a value of 0 is not allowed.
glossfont_name	is the name of the gloss font used in a Specialty Imaging application. Also refer to Specialty Imaging with VIC in the *FreeFlow VI Compose User Guide*.

microfont_name	is the name of the micro font used in a Specialty Imaging application. Also refer to Specialty Imaging with VIC in the *FreeFlow VI Compose User Guide*.
GLT/MPR	are built-in variables that deliver the font size and set the appropriate line spacing for a given Specialty Imaging font.

Modes

This command is applicable in all modes.

Related commands

- INDEXFONT
- SETENCODING
- GLT
- MPR

SETFORM

SETFORM selects a form to be printed on the current and all subsequent pages. This command must be specified before any marking command on the page.

Forms must be coded in PostScript or VIPP® native mode and stored as procedures, in one of the libraries referenced by **SETFPATH**. Use of the .frm extension is recommended. Procedures are encapsulated within braces "{ }".

Syntax

```
Form_ref SETFORM
Form_ref planenumber SETFORM
Form_ref [ c1 c2 ... cn ] SETFORM
Form_ref planenumber [ c1 c2 ... cn ] SETFORM
```

Where:

Form_ref

Any one of these:

- **(Formname)** the name of a form enclosed in parenthesis
- **null** to disable the form (default)
- **{ form contents }** VIPP® code enclosed in braces to build a small in-line form instead of coding an external file.
- **[FormRef1 FormRef2 ... FormRefn]** a list of **FormRef** enclosed in square brackets to be used in sequence in a cyclical manner, cycle forms.

planenumber

Is the planenumber index (default is 0). It refers to the capability of imaging several forms on top of each other. planenumber ranges from 0 to maxplanenumber -1. The planenumber default is 0. The default for maxplanenumber, set by **SETMAXFORM**, is 1. This also determines the order in which forms are imaged.
A form with a planenumber of 0 is imaged before a form with a planenumber of 1. This order must be handled carefully because PostScript elements are opaque.

[c1, c2, ... cn]

Defines a copy range selection. It must be used with **SETCYCLECOPY**. Copies selection specifies on which copies the form is imaged for example, [1 5] indicates that the form is imaged only on copies one and five. When not specified, the form is imaged on all copies.

An EPS, JPEG, or a TIFF file can be used as a form by using the **CACHE** command after the resource name. For further information, refer to *CACHE*.

Form origin for forms coded using the VIPP® language:

- The bottom-left corner of the page by default or when **ORIBL** is coded
- The top-left corner of the page when **ORITL** is coded

Bounding Box:

If SETFORM is used in conjunction with **CACHE** the bounding box statement, **%%BoundingBox**: llx lly urx ury, if any, is used to determine the size and clipping of the form image. llx lly is considered as the origin of the image related to the bottom-left corner of the page.

Examples

```
(form1.frm) SETFORM
(form1.frm) 0 SETFORM
(form2.frm) 1 SETFORM
(copy.frm) 3 [2 3] SETFORM
null SETFORM
null 3 SETFORM
{ PORT 100 3200 MOVETO (image.tif) 1 0 ICALL } SETFORM
[ (form1.frm) (form2.frm) (form3.frm) ] SETFORM
```

These examples use **CACHE SETFORM**:

```
(form1.ps) CACHE SETFORM

(logo.eps) CACHE SETFORM

(image3.tif) CACHE SETFORM
```

Combine SETFORM, plane numbers, and condition statements to perform unique functions. In this example an in-line form is used to position a different image on the page based on a value in a record.

```
{ /VAR_FIRST8 1 0 8 GETFIELD } BEGINPAGE

{ x y MOVETO
  CASE VAR_FIRST8 {}
   (xxxxxxxx) { (image1.tif) 1 0 ICALL }
   (yyyyyyyy) { (image2.tif) 1 0 ICALL }
   ....
  ENDCASE
} SETFORM
```

Note: Always use the **SETMAXFORM** command when you use multiple forms. If you do not use the **SETMAXFORM** with multiple forms, a range check error occurs.

Note: For specific syntax that is related to Decomposition Services on DocuPrint, NPS, and FreeFlow Print Server systems, refer to *Decomposition Services Hints and Tips* in the *FreeFlow VI Compose User Guide*.

Modes

This command is applicable in all modes.

Related commands

- SETBFORM
- SETCYCLECOPY
- SETFPATH
- SETMAXFORM
- SLIPSHEET
- CACHE

SETFPATH

SETFPATH defines a library or a list of libraries for forms. The default is defined in the file `/usr/xgf/src/xgfunix.run` or `x:\xgf\src\xgfdos.run`.

Use **SETFPATH** commands only in the xgfunix.run or xgfdos.run files. Adding any **SETFPATH** to a VIPP® job compromises portability. Adding any **SETFPATH** to a VI Project compromises both portability and project organization.

Syntax

```
(path to form library) SETFPATH
```

```
[ (path to form library 1) (path to form library 2) ... ] SETFPATH
```

The libraries specified with **SETFPATH** are used by **SETFORM**, **SETBFORM**, **SCALL**, **FCALL**, and **STARTDBM** to locate forms, segments, and Data Base Masters (DBM).

When a list of libraries is specified, they are searched in the order in which they appear in the list.

Modes

This command is applicable in all modes.

Related commands

- FCALL
- SCALL
- SETBFORM
- SETFORM
- STARTDBM
- CACHE

SETFRAME

SETFRAME draws an overall frame around the current page and all following pages.

Syntax

`linewidth offset Colorkey cradius SETFRAME`

Where:

linewidth	is the width of the line in current units. 0 indicates no frame.
offset	is the distance in current units from the margins.
Colorkey	is the color.
cradius	is the corner radius of the frame in current units. 0 indicates square.

The default setting is specified in the `/usr/xgf/src/xgf.def` file.

Modes

This command is applicable in all modes.

Related commands

SETMARGIN, SETPAT, SETTXC

SETFTSP

SETFTSP selects and scales a font. In addition, it sets the line spacing to point size * 1.2, which is the most common spacing value. The syntax is otherwise identical to SETFONT.

Syntax

```
/Fontname size SETFTSP
```

```
/Fontname sizeX sizeY SETFTSP
```

For a description of the operands, refer to SETFONT.

Example

```
/NHE 12 SETFTSP
```

Assuming **POINT SETUNIT**, the example above is equivalent to:

```
/NHE 12 SETFONT 14.4 SETLSP
```

Modes

This command is applicable in all modes.

Related commands

SETFONT SETLSP

SETFTSW

The **SETFTSW** command sets the font/color switch for subsequent **SHMF** or **SHP and SHp** commands.

Syntax

```
(ftsw) SETFTSW
(ftsw) index_length SETFTSW
```

Where:

ftsw	is an alphanumeric string that will trigger a font or color switch. Choosing a character sequence that is not likely to appear in the data to be printed is recommended.
index_length	indicates the number of bytes following ftsw in the data that represents the font/color index. When no index_length is indicated the operand defaults to 1.

The default value (//) is defined in the `/usr/xgf/src/xgf.def` file.

Examples

This example shows how to use a three-character font index with an **SHMF** command.

```
(//) 3 SETFTSW
/H10 /NHE 10 INDEXFONT
/H12 /NHE 12 INDEXFONT
(//H10 use Helvetica 10 //H12 use Helvetica 12) 0 SHMF
```

Modes

This command is applicable in all modes.

Related commands

- INDEXCOLOR
- INDEXFONT
- INDEXSST
- SETTXS
- SHMF, SHMf, and SHmf
- INDEXBAT
- SHP and SHp
- INDEXALIGN
- INDEXLSP

SETGEP

SETGEP defines a Graphic Element Property key (**GEPkey**) used in subsequent draw commands.

Syntax

```
/GEPkey LineWidth LineColorkey LineDash FillColorkey SETGEP
```

Where:

GEPkey	is any alphanumeric string starting with an alphabetic character.
LineWidth	defines the width of the outline border. 0 indicates no outline.
LineColorkey	defines a Colorkey for the outline border.
LineDash	defines the dash pattern for the outline border. 0 indicates a solid line. It may be either a single number defining the equal width of filled and unfilled portions or a list specifying the initial offset and a width sequence [offset gap1 gap2 ... gapn].
FillColorkey	defines a Colorkey used to fill the inside.

Units for LineWidth and **LineDash** are defined by **SETGUNIT**.

Examples

This example specifies an evenly spaced dashed black outline border defined as follows:

- The width of the outline border is four units wide
- The pattern consists of six black units alternating with six white units
- A light medium **LMEDIUM** inside fill pattern is used

```
/GEPK1 4 BLACK 6 LMEDIUM SETGEP
```

This example specifies an oddly spaced dashed black outline border.

```
/GEPK2 4 BLACK [ 0 10 5 5 5 ] LMEDIUM SETGEP
```

Modes

This command is applicable in all modes.

Related commands

- DRAWB and DRAWBR
- DRAWBM and DRAWBRM
- DRAWPATH and DRAWPATHR
- DRAWPOL
- SETGUNIT
- SETPAT
- SETTXC
- SHX
- SETCOL
- SETTRAN

SETGRID

SETGRID sets the number of characters per line (cpl) and the number of lines per page (lpp).

Syntax

```
cpl lpp SETGRID
```

Where:

cpl and lpp	can be plain numbers or numeric string values.

Modes

This command is applicable in all modes.

Related commands

LSP, SETFONT

SETGUNIT

SETGUNIT sets the unit of measure for all subsequent **SETGEP** commands.

Syntax

`unit SETGUNIT`

Where:

unit	can include:
	DOT3 (1/300 inch) default value
	PELS (1/240 inch)
	POINT (1/72 inch)
	CM (centimeter)
	MM (millimeter)
	INCH (inch)

Using **PELS** may help when converting AFP resources to VIPP® resources.

Modes

This command is applicable in all modes.

Related commands

SETGEP, SETUNIT

SETINDENT

SETINDENT sets the indention for subsequent **SHP and SHp** commands. Once set, an indentation causes the first line of each paragraph printed with **SHP** and **SHp** to be offset horizontally.

Syntax

`indent SETINDENT`

Where:

indent	is the indentation value in current units. The default is 0.

When SETINDENT is used in conjunction with SHP and SHp, option +20 is applied to each sub-paragraph.

Modes

This command is applicable in all modes.

Related commands

SHP and SHp

SETIPATH

SETIPATH defines a library or a list of libraries for images. The default is defined in either the `/usr/xgf/src/xgfunix.run` or `x:\xgf\src\xgfdos.run` file. The specified libraries are used by **ICALL** to locate images.

Use **SETIPATH** commands only in the xgfunix.run or xgfdos.run files. Adding any **SETIPATH** to a VIPP® job compromises portability. Adding any **SETIPATH** to a VI Project compromises both portability and project organization.

Syntax

```
(path to image library) SETIPATH
```

```
[ (path to image library 1) (path to image library 2) ... ] SETIPATH
```

When a list of libraries is specified, they are searched in the order in which they appear in the list.

Modes

This command is applicable in all modes.

Related commands

ICALL, RUNTIF, CACHE

SETJDT

SETJDT activates the settings in the JDT for subsequent pages. Unspecified settings are inherited from the previous settings, not from the base defaults.

JDT must be coded with VIPP® native mode commands and stored in one of the libraries referenced by **SETJPATH**. Use of the `.jdt` extension is recommended. Refer to VIPP® data streams in the *FreeFlow VI Compose User Guide*, and the **STARTLM** command for further information on JDT processing.

Syntax

```
jdt_ref SETJDT

jdt_ref count SETJDT
```

Where:

jdt_ref	can be one of these: **(JDTname)** the name of a JDT enclosed in parenthesis **[(jdtname1) {JDT contents}... (jdtnameN)]** a list enclosed in square brackets that consists of jdtnames or VIPP® code (inside braces) that will be used in sequence in a cyclical manner (cycle JDT). VIPP® code inside braces can be used to build a small in-line JDT instead of coding an external file.
count	expresses the number of pages after which the JDT is applied (this is similar to the count operand of STARTLM). This operand delays the application of a JDT on subsequent pages. Refer to the GETFIELD example for further information.

Note: When inside a JDT, marking commands are allowed only in an in-line form or an **ENDPAGE** procedure.

TIP

Use this command in a Native Mode Prefix (**NMP**) record (**%%XGF**) to allow the JDT to change on a page-by-page basis. The JDT referenced by **SETJDT** is called a slave JDT, which contains only settings that change page-by-page. The master JDT referenced by **STARTLM** at the beginning of the job contains all of the global settings.

The **BEGINPAGE** command has been designed to facilitate the use of banner, master, and slave JDTs in the same application. Refer to BEGINPAGE for further information.

Important: The maximum number of JDTs that can be included in a single job is 65535. The maximum number of JDTs that can be included in a **SETJDT** command is also 65535. However, this many JDTs in a single **SETJDT** command can degrade system performance and deplete virtual memory. Therefore, Xerox recommends that fewer than 1000 JDTs are included in a **SETJDT** command.

Modes

This command is applicable in all modes.

Related commands

- SETJPATH
- STARTLM
- STARTDBM

- SETPAGEDEF

SETJPATH

SETJPATH defines a library or a list of libraries for JDTs. The default is defined in the file `/usr/xgf/src/xgfunix.run` or `x:\xgf\src\xgfdos.run`.

Use **SETJPATH** commands only in the xgfunix.run or xgfdos.run files. Adding any **SETJPATH** to a VIPP® job compromises portability. Adding any **SETJPATH** to a VI Project compromises both portability and project organization.

Syntax

```
(path to JDT library) SETJPATH
```

```
[ (path to JDT library 1) (path to JDT library 2) ... ] SETJPATH
```

The specified libraries are used by **STARTLM**, **SETJDT**, and **SETDLFILE** to locate JDTs.

When a list of libraries is specified, as in the second syntax listed above, they are searched in the order in which they appear in the list.

Modes

This command is applicable in all modes.

Related commands

SETDLFILE, SETJDT, STARTLM

SETKERN

The **SETKERN** command sets the kerning options for all subsequent text imaged using one of the **SHx** commands.

Syntax

```
[ PW_opt TG_opt TK_deg ] SETKERN
```

Where:

PW_opt	defines the pair-wise kerning option. PW_opt can take one of these values: **0** disable pair-wise kerning. **not 0** enable pair-wise kerning by multiplying the pair-wise kerning values provided by the AFM file (KP, KPX or KPY entries). The recommended value is 1 and can be a real number. **null** keep the current pair-wise kerning option in effect.
TG_opt	defines the generic track kerning option. TG_opt can take one of these values: **0** disable generic track kerning. **not 0** enable track kerning by multiplying generic track kerning values defined by VIPP®. The recommended value range is -3 to +3, and can be a real number. **null** keep the current generic track kerning option in effect.
TK_deg	defines the track kerning degree. TK_deg can take one of these values: **0** disable track kerning degree. **not 0** enable track kerning by selecting values from the closest track kern degree defined in the TrackKern entries of the AFM file. Track kerning degrees generally range from -3 to +3, they must be an integer. **null** keep the current track kerning degree in effect. When TK_deg is not zero and TrackKern entries exist in the AFM file, TG_opt is ignored. When TK_deg is not zero and no TrackKern entries exist in the AFM file, TG_opt is used instead. PW_opt and TK_deg are ignored when no AFM file is associated with the current font or no kerning information is present in it. TG_opt does not require any information from the AFM file so it can be used even when there is no AFM file associated with the current font. Use [0 0 0] SETKERN to disable all kerning options (default).

Examples

Use this example to enable pair-wise kerning:

```
[1 0 0] SETKERN
```

Use this example to enable AFM medium track kerning:

```
[0 0 -2] SETKERN
```

Use this example to enable pair-wise and AFM medium track kerning:

```
[1 0 -2] SETKERN
```

Use this example to enable medium generic or AFM medium track kerning:

```
[0 -2 -2] SETKERN
```

Use this example to enable pair-wise and light generic track kerning:

```
[1 1 0] SETKERN
```

Modes

This command is applicable in all modes.

Related commands

SHx, INDEXKERN

SETLAYOUT

SETLAYOUT is an alternative to **SETMULTIUP** to establish an even Multi-Up layout.

Syntax

```
/layoutname SETLAYOUT
[ /PageWidth    pagewidth
  /PageHeight   pageheight
  /TopBleed     topbleed
  /LeftBleed    leftbleed
  /RightBleed   rightbleed
  /BottomBleed  bottombleed
  /HGutter      hgutter
  /VGutter      vgutter
  /Across       across
  /Down         down
  /Rotate rotate | [ rotate1 rotate2 ... rotateN ]
  /FillOrder    fillorder
  /LayoutMarks  markoption
  /MarkLength   marklength
  /MarkWidth    markwidth
  /MarkOffset   markoffset
] SETLAYOUT
```

Where:

Use current units when appropriate.

layoutname	is the name of a layout defined by **DEFINELAYOUT**
pagewidth	is the width of logical pages
pageheight	is the height logical pages
topbleed	is the top bleed (default: 0)
leftbleed	is the left bleed (default: 0)
rightbleed	is the right bleed (default: 0)
bottombleed	is the bottom bleed (default: 0)
hgutter	is the horizontal gutter (default: 0)
vgutter	is the vertical gutter (default: 0)
across	is the number of logical pages across the sheet (default: 1)
down	is the number of logical pages down the sheet (default: 1)
rotate	is the rotation of logical pages; possible values: 0 (default), 90, 180, 270 (or -90). When an array is used the rotate values are applied in cyclical sequence in the order of the pages.
fillorder	is the order for filling the logical pages (default: /RD) **/RD** (right-down) left to right then top to bottom **/LD** (left-down) right to left then top to bottom **/RU** (right-up) left to right then bottom to top

	/LU (left-up) right to left then bottom to top
	/DR (down-right) top to bottom then left to right
	/UR (up-right) bottom to top then left to right
	/DL (down-left) top to bottom then right to left
	/UL (up-left) bottom to top then right to left
markoption	is the option for layout marks on logical pages:
	0 no marks (default)
	1 crop marks
	2 bleed marks
	3 crop and bleed marks
	+ option to print the marks on front, back or both
	+0 print marks on front page only
	+10 print marks on back page only
	+20 print marks on front and back pages
	+100 disable inner crop marks
	+200 disable inner bleed marks
	+300 disable inner crop and bleed marks
marklength	is the length of marks in current units (default 0)
markwidth	is the width of marks in current units (default 0)
markoffset	is the mark offset from the corner in current units (default 0)

Any parameter with a default can be omitted. **SETPARAMS** is used to change default values or assigned default values to parameters that do not have one.

A layout simulator `/xgf/demo/layoutsimulator.nm` is available to help implement the **SETLAYOUT** command.

Examples

```
POINT SETUNIT
612 792 SETPAGESIZE
[ /PageWidth  216
  /PageHeight  270
  /HGutter  18
  /VGutter  18
  /Across  2
  /Down  3
  /Rotate  90
] SETLAYOUT


[...
  /Rotate  [  0 180 0 180  ]
  ..
  ]  SETLAYOUT
```

Modes

This command is applicable in all modes.

Related commands

DEFINELAYOUT, SETMULTIUP

SETLFI

The **SETLFI** command sets the line feed increment.

Syntax

```
lfi SETLFI
```

Where:

lfi	is the number of lines skipped after each record when positive, or before each record when negative. The value must be an integer. The default is 1.

The line spacing value is defined by **SETGRID** or **SETLSP**.

Modes

This command is applicable in line mode.

Related commands

SETGRID, SETLSP

SETLKF

The **SETLKF** command enables the Linked Frames mode (text re-flow). This mode allows you to define a collection of rectangular frames on the page into which text or graphical elements will be placed by VIPP® commands such as **SHx**, **ICALL**, or **SCALL**. Frames will fill in the order defined by **SETLKF**. When the last frame is filled, an implicit **PAGEBRK** occurs and the next element (or remaining part of the text element in case of **SHP** and **SHp**) will be placed in the first frame of a new page.

Do not use the **MOVETO** command when **LKF** mode is active. Only relative placements using **MOVEH**, **MOVEHR**, **NL**, and **SETLSP** can be coded. Elements that must fall into fixed locations must be placed using a form definition (**SETFORM**) either in-line or external.

All alignments will be computed according to the current frame (center means center of frame, right means right edge of frame, column width is adjusted to the width of the frame, etc. Secondary print position as defined by **MOVEH**, or **MOVEHR** will apply inside the frame when it differs from the main print position.

SETLKF, FOREACH and SHROW

SETLKF, **FOREACH**, and **SHROW** can interact with each other when **SHROW** is called in the **FOREACH** loop. This behavior is enabled by the following **FOREACH** syntax:

```
{ sequence of VIPP commands } variable_array /MF FOREACH
```

When the /MF option is used, the total size of all **SHROW** called in the **FOREACH** procedure is evaluated. If it does not fit in the frame, the rows are not imaged, a **NEWFRAME** is called, and the procedure is executed a second time with the same table entry values. This option is intended to address multiple frames with different layouts, so that on the second execution a different **SHROW** is possibly called.

Syntax

```
[ [ Hor1 Ver1 Width1 Height1 rotate1 ]
  [ Hor2 Ver2 Width2 Height2 rotate2 ]
  ....
  [ HorN VerN WidthN HeightN rotateN ]
] SETLKF
```

This syntax is used to turn off linked frame mode:

```
[] SETLKF
```

Where:

HorX and VerX	define the horizontal and vertical coordinates (in current units) of the top left corner of the frame related to the current origin (TL or BL) of the logical page.
WidthX and HeightX	define the size of the frame (in current units).
rotateX	defines the rotation values. The supported value is 0, VI Compose ignores any other value.

Note: The vertical placement (VPOS) in a frame starts at the top of the frame. In a text block in a frame, the VPOS is not the baseline of the font, but is the top of the character cell (assuming similar font size and LSP). The baseline of the font is positioned downward 70 % of LSP. This allows text ascenders (and descenders at the bottom of the frame) to fit within the frame. For example, the T character in the first line will not get truncated by the top of the frame and the "p" character will not get truncated by the bottom of the frame.

Frame transition is triggered when the remaining space is less than the line spacing. LSP should represent the minimal space to fit one character (70 % for ascenders and 30 % for descenders).

This rule does not apply to charts, images, and segments. The origin must be placed in relation to the width, height, and alignment of the item, possibly by using the **NL** and **MOVEHR** commands when the current VPOS does not fit.

Tip: Select an LSP close to the font size, then you can do minor vertical adjustments using NL. To make the adjustment proportional to LSP you can use an arithmetic expression.

Example:

```
LSP'*'.2 NL % move down 20% of the current LSP
```

Examples

LSP'*'.2 NL % move down 20 % of the current LSP

```
[ [  200 200 1025 2900 0 ]     % Left column
  [ 1325 200 1025 2900 0 ]     % right column
] SETLKF
```

Note: To allow automatic flow over page boundaries, code **SETLKF** before any marking command on the page. When not the case (for example when **SETLKF** is coded in a form definition) no page transition will occur. Instead the last frame will overflow beyond the bottom edge of the frame and possibly beyond the page edge. It is possible to define different sets of frames and forms for consecutive pages using the **SETPAGEDEF** command.

Modes

This command is applicable in native and database modes.

Related commands

- FRCOUNT
- FRLEFT
- GOTOFRAME
- NEWFRAME
- SETGRID
- SETLSP
- SETMARGIN

SETLMFILE

SETLMFILE causes the following **STARTLM** or **STARTDBM** command to read data from a specified file, which must be located in one of the libraries referenced by **SETMPATH**.

Syntax

```
(file name) SETLMFILE
```

```
(file name) skiplines SETLMFILE
```

The second syntax causes the first skiplines lines in the file to be ignored.

Examples

Submitting the syntax contained in this example causes the data file `invoice.dat` to print using the layout described in `invoice.jdt`.

```
%!
(invoice.dat) SETLMFILE
(invoice.jdt) STARTLM
```

Modes

This command is applicable in line mode and database mode.

Related commands

SETMPATH, STARTDBM, STARTLM

SETLSP

The **SETLSP** command sets line spacing. Line spacing is computed from the margins and grid specifications by default.

Syntax

```
LSPval SETLSP
```

Where:

LSPval	is the line spacing value.

Note: Specifying a negative line spacing value will result in the print position moving up the page. This is often used to print an address block where the postal code or zip code needs to be placed at a specific X, Y position and the rest of the address lines spaced from that position.

It is important to reset the line spacing to a positive value for the rest of the application.

Modes

This command is applicable in all modes.

Related commands

- SETGRID
- SETMARGIN
- SHC and SHc
- SHJ and SHj
- SHL and SH
- SHR and SHr
- SHX
- SHMF, SHMf, and SHmf
- SHP and SHp
- INDEXALIGN
- INDEXLSP

SETMARGIN

SETMARGIN sets the top, bottom, left, and right margins using the current units. The default is specified in the `/usr/xgf/src/xgf.def` file.

Syntax

`top bottom left right SETMARGIN`

Modes

This command is applicable in line mode.

Related commands

SETFONT, SETGRID

SETMAXBFORM

SETMAXBFORM sets the maximum number of planes of backforms on one page. Plane numbers range from 0, to maxplanenumber-1, and associated backforms are imaged in that order. PostScript elements are opaque.

SETMAXBFORM must be used when the application requires more than one back form.

Syntax

```
maxplanenumber SETMAXBFORM
```

Modes

This command is applicable in all modes.

Related commands

SETBFORM

SETMAXCOPY

SETMAXCOPY sets the maximum number of copies for a job to the specified value. Use this command when **SETCYCLECOPY** is set dynamically using **PROCESSDJDE** or **BEGINPAGE**. **SETMAXCOPY** does not set the copy count until a **SETCYCLECOPY** command is executed. The value for **SETCYCLECOPY** cannot be greater than the value specified for **SETMAXCOPY**.

Syntax

```
number SETMAXCOPY
```

Where:

number	is a user-defined value.

Examples

```
10 SETMAXCOPY
{CASE DJDECMD {} (COPIES) {DJDEPAR SETCYCLECOPY} ENDCASE } 0 ($DJDE) 3
PROCESSDJDE
```

Modes

This command is applicable in all modes.

Related commands

SETCYCLECOPY, PROCESSDJDE

SETMAXFORM

SETMAXFORM sets the maximum number of planes of forms on one page. Plane numbers range from 0, to maxplanenumber-1, and associated forms are imaged in that order. PostScript elements are opaque. The default is specified in the `/usr/xgf/src/xgf.def` file.

Syntax

```
maxplanenumber SETMAXFORM
```

Modes

This command is applicable in all modes.

Related commands

SETFORM

SETMEDIA

The **SETMEDIA** command sets the media requirement.

In the syntax examples below, **SETMEDIA** sets **MediaType**, **MediaColor**, and **MediaWeight** as the current media type requirements for subsequent pages.

Syntax

```
(MediaType:MediaColor:MediaWeight) SETMEDIA
(MediaType:MediaColor:MediaWeight) [c1 c2 ... cn] SETMEDIA
[ (Med req1) (Med req2) ...(Med reqN) ] SETMEDIA
```

This media requirement is compared to the related values set by the **SetTray** command on the DocuPrint NPS user interface, or by the VI Compose utility, trayload. When one or more trays match this requirement, the sheets for the current and subsequent pages, up to the next **SETMEDIA** command, are fed from these trays.

The second syntax sets the media requirement for the specified selection of copies [c1, c2, ... cn].

The third syntax defines a list of media requirements to be used in sequence in a cyclical manner (cycle media).

Keep this in mind when using **SETMEDIA**:

- When any of the media attributes are specified as null, those attributes are ignored in the following media selections. This example ignores MediaColor.

  ```
  (Drilled:null:100) SETMEDIA
  ```

- When any of the media attributes such as type, color, or weight are omitted, the last specification or the default value for that attribute remains in effect.
- The trailing ":" can be omitted as shown in this example.

  ```
  (Plain::) SETMEDIA
  (Plain:) SETMEDIA
  (Plain) SETMEDIA
  ```

The **PageSize** media attribute is not set by this command, it is set using the **SETPAGESIZE** command.

Note: For more information about devices that do not support standard PostScript media selection, refer to Media support.

Tip: Coding a %%DocumentMedia record at the beginning of the data file is required on DocuPrint NPS to force the related trays to be available for the job. Refer to %% for more information.

Also refer to VIC considerations for iGen in the *FreeFlow VI Compose User Guide* for more information.

For FreeFlow Print Server only

The syntax of **SETMEDIA** has been extended to support selection of coating for front and back.

You can use the media option **postnoimage** when pulling media from a tray, sometimes referred to as the Bypass or Imposer tray, that bypasses the imaging path. This action is used to pull pre-printed forms that bypass the imaging path. The following example inserts a single insert: `(postnoimage:pink) SETMEDIA`

```
0 NL PAGEBRK
```

The following example inserts five ordered inserts:

```
(postnoimage/5:white) SETMEDIA
{ 0 NL PAGEBRK } 5 REPEAT
```

Syntax

```
(MediaType:MediaColor:MediaWeight:MediaFrontCoating:MediaBackCoating)
SETMEDIA
```

Where:

MediaFrontCoating and MediaBackCoating	refer to the front and back coating attributes of a given media and can have the following values: **None (uncoated)** **Glossy** **HighGloss** **SemiGloss** **Satin** **Matte**

These new attributes follow the same rules for media attributes as described in the **SETMEDIA** command and can be applied to all forms of the **SETMEDIA** syntax.

Examples

```
(Cover1:::Glossy:None) SETMEDIA
(Cover2::120:Matte:Matte) SETMEDIA
```

Modes

This command is applicable in all modes.

Related commands

- % %
- SETCYCLECOPY
- SETPAGESIZE
- SETMEDIAT
- SLIPSHEET

SETMEDIAT

SETMEDIAT temporarily sets the media requirement for the current page. The next page reverts to the previous media selected, or to the media required to satisfy the cycle media.

Syntax

```
(MediaType:mediaColor:MediaWeight) SETMEDIAT
```

As with the **SETMEDIA** command, the media values can be unchanged by leaving the corresponding fields empty.

Examples

This example selects a media with *Type=drilled*, *Weight=120*, and Color equal to the current setting for the current page. However, the subsequent page reverts to the previous settings.

```
(drilled::120) SETMEDIAT
```

Modes

This command is applicable in all modes.

Related commands

% %, SETMEDIA, SETPAGESIZE

SETMPATH

SETMPATH defines a library or a list of libraries for miscellaneous files. The default is defined in the file `/usr/xgf/src/xgfunix.run` or `x:\xgf\src\xgfdos.run`.

Use **SETMPATH** commands only in the `xgfunix.run` or `xgfdos.run` files. Adding any **SETMPATH** to a VIPP® job compromises portability. Adding any **SETMPATH** to a VI Project compromises both portability and project organization.

Syntax

```
(path to misc. library) SETMPATH

[ (path to misc. library 1) (path to misc. library 2) ... ] SETMPATH
```

The specified libraries are used by **SETLMFILE**, **SETDLFILE**, **CACHE**, and **RUN** to locate the files referred to by these commands.

When a list of libraries is specified, as in the second syntax, they are searched in the order in which they appear in the list.

Modes

This command is applicable in all modes.

Related commands

- RUN
- SETDLFILE
- SETLMFILE
- STOREVAR
- CACHE

SETMULTIUP

SETMULTIUP enables Multi-Up printing. Multi-Up printing is the printing of several logical pages on one physical page.

Syntax

In this syntax each logical page is defined by an entry.

```
[ Hor1 Ver1 rotate1 Hscale1 Vscale1 % setting for logical   page 1
  Hor2 Ver2 rotate2 Hscale2 Vscale2 % setting for logical   page 2
.....
]  SETMULTIUP

[ [ Hor1 Width1 ] [ Ver1 Height1 ] rotate1 Hscale1 Vscale1
  [ Hor2 Width2 ] [ Ver2 Height2 ] rotate2 Hscale2 Vscale2
   ....
]
```

Where:

Hor and Ver	define the horizontal and vertical origins of the logical pages. These are always measured from the bottom left corner of the physical page.
Width and Height	define the size of the original page. The original page size is the size for which the job has been originally designed. When not specified, the size of the original page is equal to the size of the current physical page (as per printer default or set by **SETPAGESIZE**). The size of the logical page will be computed by applying the scaling factors (Hscale and Vscale) to the original page size. The original page size specification is useful when the size of the original pages differs from the physical page (for example, print 2 A4 pages on A3 paper).
rotate	defines the rotation angles.
Hscale and Vscale	define horizontal and vertical scaling.

When Multi-Up is enabled, each page skip that occurs after a **PAGEBRK**, Form Feed, Skip to channel one, or last line reached causes a skip to the next logical page. The physical page is printed only when the last logical page is filled or when the end of the job is reached.

In Multi-Up mode, consider that when no proper reduction is applied (reduction = scaling<1), only the part of the logical page overlapping the physical page is printed, adjustments to placement values is required.

This command must be used as the first command following the %! line in the print file or as the first command in a JDT.

Use **NEWFRONT**, **NEWBACK**, and **NEWSIDE** to force a new physical page.

Examples

This example defines two-up printing on A4 paper with DOT3 units.

```
[ 0 3500 -90 .7 .7

0 1750 -90 .7 .7 ] SETMULTIUP
```

This example defines two-up A4 printing on an A3 sheet for a document that has been ripped by the Decomposition Service.

```
4960 3500 SETPAGESIZE                % size of the sheet
[ [   0 2480][0 3500] 0 1 1          % original A4 page
  [2480 2480][0 3500] 0 1 1          % original A4 page
] SETMULTIUP
(doc1.ps) RUNDD
```

Note: Using Multi-Up mode and reduction can affect printer performance, especially when images are called.

Where you place the **SETMULTIUP** command dictates the result. For example, when a four-up statement is coded inside the DBM file, four copies of the currently processing record will print. Four copies of the same physical page. When the Multi-Up is coded in the data file, or to create and call a JDT with a four-up **SETMULTIUP** statement, the resulting page contains records 1, 2, 3, and 4.

Tip: Use this command with **SETCYCLECOPY** and **COLLATE_off** to produce several copies of a smaller document on one sheet of paper. The coordinates of each logical page are related to the bottom left corner of the physical page, and define the bottom left corner of the logical page.

Globally pre-defined Multi-Up settings for a site should be placed in the file `/usr/xgf/src/xgf.mup`. Refer to Standard lists, tables, keys, and attributes in the *FreeFlow VI Compose User Guide* for further information. To invoke them, use this syntax:

```
VAR.4UP SETMULTIUP
```

When you create a job that does not use the complete page, with the intent to use **SETMULTIUP**, for example, a check application that will be printed four-up on a page, create a master physical page with the check image and variable data placed on the bottom of the page and not on the top. This makes it easier to set the vertical and horizontal positions used in the **SETMULTIUP** command, as the bottom left hand corner of the master page is the 0,0 coordinate used to position the second, third and fourth check in the Multi-Up definition.

Modes

This command is applicable in all modes.

Related commands

- NEWBACK
- NEWFRONT
- NEWSIDE
- ONEUP
- TWOUP
- SETLAYOUT

SETNMP

SETNMP defines the Native Mode Prefix (NMP) string. Refer to VIPP® data streams in the *FreeFlow VI Compose User Guide*, and to % %XGF for further information on NMP.

Syntax

```
(NMP string) SETNMP
```

% %XGF is the default NMP value defined in the `/usr/xgf/src/xgf.def` file.

Modes

This command is applicable in line mode

Related commands

% %XGF, NMP_off, STARTLM

SETOBIN

SETOBIN sets the output bin destination.

Syntax

```
(OutputType) SETOBIN
(OutputType) [ c1 c2 ... cn ] SETOBIN
[ (OType1) (OType2) ... (OTypeN) ] SETOBIN
```

The first syntax allows you to select a specific output tray. The second and third syntax allow you to select a different output tray for each copy of a document or to define a cycle for output of each page to a different tray, similar to the **SETMEDIA** cycle syntax.

The OutputType is device-dependent and may be mapped to standard keys using **SETVAR**.

Examples

```
/VARTT (TopTray) SETVAR
VARTT SETOBIN
```

Each printer will vary the description of output bins. Check for legal OutputType values in the printer documentation.

SETOBIN relies on the OutputAttributes dictionary, which is a PostScript Level 2 feature not implemented on all Level 2 devices.

Note: On a FreeFlow Print Server, you can use this command to set an external finisher as a subset finishing destination. For example: `(SBM) SETOBIN`. In this situation, do not use **/ON SETFINISHING**.

Modes

This command is applicable in all modes.

Related commands

SETOBINT

SETOBINT

SETOBINT temporarily sets the output bin destination for the current page. The next page reverts to previous output bin destination or next in cycle output bin.

Syntax

```
(OutputType) SETOBINT
```

Modes

This command is applicable in all modes.

Related commands

SETOBIN

SETOTL

The **SETOTL** command sets text outline for all subsequent printed text. By default, text outline is disabled.

Syntax

```
[ LineWidth Colorkey ] SETOTL
```

Where:

LineWidth	is the width of the line in current units.
Colorkey	defines the color of the line.

Examples

```
[ 2 RED ] SETOTL
```

Modes

This command is applicable in all modes.

Related commands

INDEXOTL

SETPAGEDEF

Use the **SETPAGEDEF** command to define different sets of frames, forms, media and other layout settings for consecutive pages. The layouts can be applied in the order they are defined in a cyclical manner unless the last layout is followed by the /R key. When using the /R key, the last layout will be applied indefinitely.

Note: When you use the **SETPAGEDEF** command, you cannot use the cycle syntax of commands such as **SETFORM** or **SETJDT**, because they use the same index. To cycle through forms, use a **BEGINPAGE** routine and **CASE** statements. The first layout definition is activated at the **SETPAGEDEF** execution. Subsequent layout definitions are activated at the end of **PAGEBRK** for the next page. To cancel **SETPAGEDEF**, use the following statement: `[{}] SETPAGEDEF`

Syntax

```
[ { layout_definition1 } { layout_definition1 } ... ] SETPAGEDEF
[ { layout_definition1 } { layout_definition1 } ... /R ] SETPAGEDEF
```

Where:

layout_definitionX	defines the layout for a particular page. It can contain any VIPP® command related to layout setting, for example, **SETLKF**, **SETFORM**, **SETMEDIA**, and **SETJDT**.

Examples

```
[ { [ [  200  500   990  1330 0 ]     % layout for page 1
      [  200 2400   990   900 0 ]
      [ 1290  500   990   430 0 ]
      [ 1290 2300   990  1000 0 ]
    ] SETLKF

    (form1.frm) SETFORM
  }
  { [ [  200  200   990  3100 0 ]     % layout for page 2
      [ 1290  200   990  3100 0 ]
    ] SETLKF

    (form2.frm) SETFORM
  }
  /R                                  % repeat layout 2
] SETPAGEDEF
```

Modes

This command is applicable in all modes.

Related commands

SETLKF, SETJDT

SETPAGENUMBER

The **SETPAGENUMBER** command sets the page numbering for all subsequent pages.

Syntax

```
(format) start pos SETPAGENUMBER
(format) start rotate pos SETPAGENUMBER
(format) start hpos vpos align SETPAGENUMBER
(format) start hpos vpos rotate align SETPAGENUMBER
```

Where:

format	is any string where # represents the page number place holder. Use multiple # characters to print leading zeros.
start	specifies the first page number. This value can be zero or negative. When using zero or a negative value, page numbers less than one do not print. The maximum value is 999999.
rotate	specifies the angle of rotation. Positive values rotate in a counterclockwise direction.
pos	provides the position as follows: **0** do not print page number (temporary disablement) **1** bottom center **2** bottom right **3** top center **4** top right
hpos and vpos	provide the absolute horizontal and vertical coordinates.
align	provides the alignment as follows: **5** left **6** right **7** center

The default page numbering, the font, reserved /PNFT font index, and color, reserved /PNCL color index, used to print the page numbers are specified in the `/usr/xgf/src/xgf.def` file.

To retain the current values while changing other page numbering options for the current and subsequent pages, use a null value in the format and start operands. The rotate operand is mandatory in this case. **SETPAGENUMBER** can be placed before any page marking commands.

Examples

This example prints Page 001 at the bottom right corner of the first page.

```
(Page ###) 1 2 SETPAGENUMBER
```

This example prints at the top center with the same format and page continuation.

```
null null 0 3 SETPAGENUMBER
```

This example prints at the bottom center with the same format, and restarts page numbering at 1.

```
null 1 0 1 SETPAGENUMBER
```

Modes

This command is applicable in all modes.

Related commands

SETMARGIN

SETPAGESIZE

The **SETPAGESIZE** command defines the page size requirement for all subsequent pages.

Syntax

```
pagewidth pageheight SETPAGESIZE
```

Where:

pagewidth	is the width of the physical page in current units. By convention pagewidth refers to the short dimension of the sheet, horizontal axis in portrait orientation.
pageheight	is the height of the physical page in current units. By convention pageheight refers to the long dimension of the sheet, vertical axis in portrait orientation.

The VIPP® commands **PORT**, **LAND**, **IPORT** and **ILAND** rely on this convention to place elements on the page. Failing to follow it can cause orientation mismatch.

All page layout settings such as orientation, grid, and margins are based on these values. When not specified, the default value is the default page size of the imaging device. An error occurs when this command is placed incorrectly, for example, in the middle of a page.

On level 2 printers, this command causes subsequent pages to print on the proper media.

Examples

This example sets the page size to portrait USLetter (8.5 x 11 in.) using current units of DOT3.

```
DOT3 SETUNIT
2550 3300 SETPAGESIZE
```

This example sets the page size to portrait A3, also using current units of MM.

```
MM SETUNIT
297 420 SETPAGESIZE
```

Modes

This command is applicable in all modes.

Related commands

% % XGF, NMP_off

SETPARAMS

SETPARAMS sets persistent parameters throughout the job. Once a parameter has been set with **SETPARAMS** that parameter applies to all of the subsequent commands to which it relates. For more information, refer to Parameter Descriptions.

Syntax

```
[parameters] SETPARAMS

[ /Name1 val1 /Name2 val2 ... /NameN valN ] SETPARAMS

[ /code /Name1 val1 /Name2 val2 ... /NameN valN ] SETPARAMS
```

Where:

NameX	is a parameter name taken from the table in Parameter Descriptions.
valX	is the value to be assigned to this parameter according to the information provided in the table.
code	is an integer that allows setting several parameters at once. It is computed by adding all codes related to the parameters required. Refer to the Parameter Descriptions. Not all parameters have an associated code.

Examples

```
[/3D true /SliceBurst .1] SETPARAMS
```

This example shows the use of /Pagerange at the beginning of a data file:

```
%!
XGF
[ /PageRange 1 ] SETPARAMS
1 10 PAGERANGE
...
```

Note: **SETPARAMS** replaces the **SETDDGPARAMS** command, but **SETDDGPARAMS** is supported for backward compatibility.

Modes

This command is applicable in all modes.

Related commands

- DRAWPIE
- DRAWBAR
- DRAWCRV
- FORMAT
- Parameters

SETPAT

Use **SETPAT** to instantiate a pattern from a prototype pattern defined using **SETPPAT**. A pattern is a small drawing that can be used repeatedly to fill an area. Once defined it can be used instead of, or in conjunction with, a color.

Syntax

```
/PATkey /PPATkey Htrans Vtrans rotate Hscale Yscale SETPAT

/PATkey (imagename) Htrans Vtrans rotate Hscale Yscale SETPAT

/PATkey [ /VPGL lwidth color1 color2 ] Htrans Vtrans rotate Hscale Yscale
SETPAT

/PATkey [ /VPCR lwidth color1 color2 ] Htrans Vtrans rotate Hscale Yscale
SETPAT

/PATkey [ /VPCR lwidth color1 color2 color3 ] Htrans Vtrans rotate Hscale
Yscale SETPAT

/PATkey [ /VPPG1 textPixelFrequency backgroundPixelFrequency color1
color2] Htrans Vtrans rotate Hscale Yscale SETPAT
```

Where:

PATkey	is the name of the pattern (Pattern key) to define any alphanumeric string starting with an alphabetic character.
PPATkey	is the name of the prototype pattern on which to build the pattern.
/VPGL	is a built-in **PPATkey** used to define a vector pattern Gloss effect.
/VPCR	is a built-in **PPATkey** used to define a vector pattern Correlation effect (1 or 2 layers).
/VPPG1	is a built-in **PPATkey** used to define a void pantograph effect.
textPixelFrequency	this defines the size (frequency) of the text pixel that needs to be hidden.
backgroundPixelFrequency	this defines the size (frequency) of the background pixel that hides the text. This supposed to be greater than double the value of textPixelFrequency.
lwidth	is the width of the hatch line in pixel units for a Gloss or Correlation effect. Most appropriate values range between 1 and 3 (real). 1 is the most common.
color1	is the background color for a Gloss or Correlation effect. This is the pixel background color of the background pixels for void pantograph effect
color2	is the hatch color for a Gloss or 1 layer Correlation effect. This is the color of background and text pixels for void pantograph effect
color3	is the hatch color for the 2nd layer of a 2 layers Correlation effect.
Htrans	is the initial horizontal translation of 1st cell in points.
Vtrans	is the initial vertical translation of 1st cell in points.
rotate	is the cell rotation in degrees.

Hscale	is the horizontal scaling factor of the cell.
Vscale	is the vertical scaling factor of the cell.
imagename	can be a JPEG, TIF, or EPS file.

Once defined by **SETPAT** a pattern key can be used in place of a **Colorkey** with any VIPP® commands that require one. Namely:

- **SETTXC**
- **INDEXCOLOR**
- **FROMLINE/RPEKEY**
- **SETPARAMS**: ColorTable, BGColor
- **SETGEP**: FillColorkey

When the pattern type is uncolored (PaintType=2) it will be painted using the last plain color defined in the VIPP® job. An alternative is to use the pattern key in conjunction with a Colorkey using this syntax:

```
[ Colorkey PATkey ]
```

Examples

```
/Star /PStar 5 –5 0 .8 .8 SETPAT % pattern definition

Star SETTXC % invoke pattern with current color

[ BLUE STAR ] SETTXC % invoke pattern with color BLUE

/VP01 [ /VPGL 1 OWHITE GL_Black ] 0 0 0 1 1 SETPAT % define a Gloss effect

/VP02 [ /VPCR 1 OWHITE CR_Red50 ] 0 0 0 1 1 SETPAT % define a Correlation effect
(1 layer)

/VP03 [ /VPCR 1 OWHITE BLUE CYAN ] 0 0 0 1 1 SETPAT % define a Correlation effect
(2 layers)

/VP04 [ /VPPG1 12 36 OWHITE BLACK ] 0 0 0 1 1 SETPAT % defines voidpantograph
effect
```

A collection of pre-defined patterns are provided in the `src/xgf.gep` file, to sample them, print `xgf/demo/sampat.nm`.

GL/CR Vector Pattern usage

Gloss and Correlation vector pattern effects are intended as a replacement for Gloss and Correlation legacy effects using dedicated fonts. With vector pattern effects any regular font can be used (except Type 3 fonts). Once a vector pattern effect has been defined using **SETPAT** the usage is similar to a Fluorescent or Infrared effect. A GL/CR vector pattern can be used to fill a box and any following text or bi-level TIFF image printed on top of the box will activate the effect.

Example

```
500 3000 1750 300 VP02 DRAWB
/NHEB 50 SETFONT
575 2770 MOVETO (Correlation Text) SHL
```

Example of GL effect on bi-level image (truk1.tif)

```
226 2400 1000 530 VP01 DRAWB
309 2380 MOVETO (truk.tif) 1.5 0 ICALL
```

Example of CR effect on bi-level image (truk.tif)

```
1347 2400 1000 530 VP02 DRAWB
1468 2380 MOVETO (truk1.tif) 1.5 0 ICALL
```

For a 2 layers correlation a second text can be placed on top of the first.

Example

```
500 3000 1750 300 VP03 DRAWB
/NHEB 50 SETFONT
575 2770 MOVETO (Correletion Text) SHL
654 2770 MOVETO (2nd layer Text) SHL
```

GL or CR colors are recommended for vector pattern effects but not mandatory. Testing is highly advised to determine colors that render the best effects on a given printer. 3 VPCs are provided in the `xgf/demo` folder to demonstrate the effects and help determining the best ones:

- `SI_VP_GlossMark.vpc`
- `SI_VP_Correlation_1L.vpc`
- `SI_VP_Correlation_2L.vpc`

Correlation key transparency overlay

To create a Vector Pattern Correlation key transparency you can simply make a simple VIPP® native mode file with the following contents:

```
/VPKey [ /VPCR 1 WHITE BLACK ] 0 0 0 1 1 SETPAT
100 100 2000 3000 VPKey DRAWB
PAGEBRK
```

and print it on a transparency. For the second layer the key must be rotated 90 degrees (rotate=90). Such a file is provided in the `xgf/demo` folder:

`SI_VP_Correlation_key.nm`

Custom Correlation variants

Note that by varying the **SETPAT** parameters Htrans, Vtrans, rotate, Hscale and Yscale it is possible to create custom correlation variants that can only work with the associated key transparency using the same parameters.

Void Pantograph usage

Void pantograph effect is intended to secure and protect the original documents from copying or tampering. Void pantograph effect can use any regular font except Type 3 fonts. Void pantograph effect has been defined using **SETPAT** command. A void pantograph can be used to fill a box and any following text with pixels.

Example

```
4 3499 2479 172 VP04 DRAWB
8 3350 MOVETO
/NHEB 50 SETFONT % (Helvetica Bold)
```

```
(Void Void Void Void Void) SHL
```

SI_VP_VoidPanto.vpc is provided in the xgf/demo folder to demonstrate the effect and help determining the best one.

Modes

This command is applicable in all modes.

Related commands

- SETPPAT
- SETTXC
- INDEXCOLOR
- SETTPAT

SETPBRK

Use **SETPBRK** to specify any string as a page delimiter. When this string is detected, the line on which it appears can be one of these:

- Last line of the current page
- First line of a new page
- Nonprintable page delimiter
- Split into a left part and a right part

In the last case, the process can be recursively repeated on the right part.

Syntax

```
(delimiter string) option SETPBRK
```

Where:

delimiter string	is the string identifying a page delimitation.
option	is a three-digit number with these possible values: First digit: **0** matching line is the last line **1** matching line is the first line Second digit: **0** print left part when not empty **1** print left part **2** print right part **3** print the complete line **4** print none Third digit: **0** loop on right part. Checks whether there is more than one instance of the test string in the record **1** do not loop on right part. Does not check whether there is more than one instance of the test string in the record.

Examples

This is the default setting for backward compatibility:

```
<0C> 000 SETPBRK
```

This example sets the Form Feed as the default page delimiter, allows multiple Form Feeds to be processed on a single line, and avoids empty pages in case of adjacent multiple Form Feeds.

```
<0C> 010 SETPBRK       % full ASCII printer compatibility

<0C> 001 SETPBRK       % FF at the end of the last line
                           % suppresses extra blank line on the next page

(Page) 131 SETPBRK     % assumes that a line with Page is the first line

($$INDEX) 141 SETPBRK % detects a nonprintable delimiter
                      % this line may contain additional information such as
    % archive indexing
```

Modes

This command is applicable in line mode.

Related commands

PAGEBRK, XGFDEBUG

SETPCC

SETPCC enables PCC processing for the current job. When PCC is enabled, vertical spacing is controlled by the first byte of each record (PCC byte) whose action is defined in the PCC definition. Refer to **BEGINPCC** for more information.

Note: When PCC bytes are used in **FROMLINE/RPEKEY** entries, where recpos=0 refers to the first data byte, skipping the PCC, column one of the original data file is not be treated as user data. It is used as the PCC index.

However, this is not true for **GETFIELD**, **SETRCD** and **SETPCD** where recpos=0 always refers to the first byte of the record regardless of whether **SETPCC** is coded or not.

Syntax

```
/pccname SETPCC
```

Where:

pccname	refers to a PCC definition previously defined by **BEGINPCC**.

Modes

This command is applicable in all modes.

Related commands

BEGINPCC, ENDPCC, SETVFU

SETPCD

The **SETPCD** command sets a page criteria definition. Although this command is similar to **SETRCD**, the condition is evaluated at the page level rather than at the record level. Tests using **SETPCD** result in an outcome of true or false.

All **PCDs** are evaluated for each page before the page composition. Therefore, **PCDkeys** can be used in **BEGINPAGE** or **ENDPAGE** procedures, in forms using native mode tests **IF**, **ELSE**, and **ENDIF**, or in an **RPE** definition in the same way RCD tests are used.

Note: When **PCC** bytes are used in **FROMLINE/RPEKEY** entries, where recpos=0 refers to the first data byte, skipping the **PCC**, column one of the original data file is not be treated as user data. It is used as the PCC index. However, this is not true for **GETFIELD**, **SETRCD** and **SETPCD** where recpos=0 always refers to the first byte of the record regardless of whether SETPCC is coded or not.

Syntax

- `/PCDkey line_nr line_ct recpos length /cnd (comp.str) SETPCD`
- `/PCDkey line_nr line_ct field_nr /FN /cnd (comp.str) SETPCD`

Where

- **PCDkey** is the **PCD** name.
- **line_nr** is the line number from which the condition can be evaluated starting from 1. If line_nr is out of range, no error occurs, but the test can be false.
- **line_ct** is the number of lines, starting at line_nr, on which the condition can be evaluated. When line_nr + line_ct exceeds the number of lines of the current page, no error occurs and line_ct defaults to its maximum number of lines.
- **recpos/length** selects the record portion of the line to compare using the compare string. recpos starts with 0.
- **field_nr** specifies the field number to compare using the compare string. field_nr starts with 0 and applies to records with a field delimited structure. The field delimiter is defined by **SETDBSEP**. The default is a semi-colon (:).
- **/cnd** specifies the test operator, for example, /eq, /ne, /ge, /gt, /le, /lt, /CIEQ, /CINE, and /HOLD. **CIEQ** and **CINE** compare the selected record portion with (comp.str) regardless of the letter case, /HOLD searches for the comparison string anywhere in the selected record portion.

  For more information, refer to Test Operators and Conditional Expressions.
- **comp.str** is the reference string for the test. For long strings repeating the same character use a count value. For example `100(*)`.

Use /PREV to refer to the equivalent string on the previous page. /ne/PREV is always true on the first page. When /PREV is used, the line_ct operand can be 1.

comp.str can also be a variable in the form of /VARxxx. This allows you to change the compare string during the job.

Note: **SETPCD** also applies on NMP records. This allows you to test non-printable data embedded in NMP comments, for example: `%%XGF% marker 1`.

Examples

- In this example, the condition is true when the string `FORM=` appears in the first 132 positions on any line from 10 to 18.

  ```
  /BANNER 10 9 0 132 /HOLD (FORM=) SETPCD
  ```

- In this example, the condition is true when the first six positions of line 10 are different from those of the previous page. On the first page the condition is always true.

  ```
  /NEWDEPT 10 1 0 6 /ne /PREV SETPCD
  ```

- In this example, the condition is true when the string `START OF JOB:` appears in position 5 for a total of 13 character positions on any line from 3 to line 5.

  ```
  /STARTBANNER 3 3 5 13 /eq (START OF JOB:) SETPCD
  ```

Modes

This command is applicable in all modes.

Related Commands

- BEGINPAGE
- ENDPAGE
- GETFIELD
- IF/ELSE/ELIF/ENDIF
- SETRCD

SETPIF

SETPIF defines a **PIF** destination or note that can be associated with the next and only element imaged on the page using **SHP**, **SHMF**, **SHX**, **ICALL**, **SCALL**, **DRAWB**, or **BOOKMARK**.

Syntax

```
[ /PIFtype param1 param2 .... ] SETPIF
```

```
null SETPIF
```

Where:

PIFtype	can be one of these: **PAGE** a page in the document **DEST** a named destination defined by **PDFDEST** **XPAGE** a page in another PDF document **XDEST** a named destination in another PDF document **FILE** a non-PDF document **URI** an Internet/Intranet site or document **NOTE** a note
null	cancels any active PIF definition.

Note: Because active PIFs are canceled automatically when they are associated with an element, use of the null syntax is seldom necessary.

Depending on PIFtype, these parameters must be supplied:

- [/PAGE pagenum view]
- [/PAGE {/Prev} view]
- [/PAGE {/Next} view]
- [/DEST /destname]
- [/XPAGE (fileref) pagenum view]
- [/XDEST (fileref) /destname]
- [/FILE (fileref)]
- [/URI]
- [/URI (URIstring)]
- [/NOTE (title) (contents)]
- [/NOTE (title) (contents) notetype color option]

Where:

pagenum	is the page number of the destination starting with 1. It can be a variable.
{/Prev}	is the destination is the previous page
{/Next}	is the destination is the next page

view defines how to adjust the view for a page destination. It can be one of these:

null use the current view.

[/Fit] fit the page to the window.

[/FitB] fit the bounding box of the page contents to the window.

[/FitR left bottom right top] the page rectangle specified by the bounding box left bottom right top is magnified and centered in the viewer window.

[/FitH top] fit the width of the page to the window and place its top origin at top current units of the page origin.

[/FitV left] fit the height of the page to the window and place its left origin at left current units of the page origin.

[/XYZ left top zoom] place the origin of the window at left top current units of the page origin.

destname is a destination name. It can be a variable.

fileref is the full or relative path and name of an external file. Refer to the /ResolvePath parameter to set the path resolution mode.

For more information, refer to Test Operators and Conditional Expressions.

URIstring is a link to an Internet or Intranet site or document name. When omitted, the text of the associated element (printed by **SHP** or **SHMF**) is used instead.

title is the title of the note.

contents is the contents of the note. It can be a variable.

Single byte data can be encoded using ISO-8859-1. Multi-byte data can be encoded using UTF8. It can be converted automatically into UTF16 by VI Compose for insertion in the PDF because this is the only multi-byte encoding supported by the PDF format. To trigger the conversion of UTF8 data to UTF16 the current font selected by **SETFONT** or **INDEXFONT** can have an UTF8 encoding.

notetype is one of these keys listed by category:

icons

- /Note
- /Comment
- /Help
- /Insert
- /Key
- /NewParagraph
- /Paragraph

stamps

- /Draft
- /Approved
- /Experimental
- /NotApproved
- /AsIs
- /Expired
- /NotForPublicRelease
- /Confidential
- /Final

	• /Sold • /Departmental • /ForComment • /TopSecret • /ForPublicRelease
	free text /FT
	markup /MK
color	is one of the following:
	A **Colorkey** that defines the color of the note
	null if no color is required. Only plain RGB and grey scale color are supported.
option	is one of these:
	+1 the note is presented open (applies to icons only). The default is closed.
	+2 notetype is a stamp.
	+4 center note icon on anchor element.

When notetype, color, and option are omitted, they default to /Note, YELLOW, and 0.

Modes

This command is applicable in all modes.

Related commands

- PDFDEST
- INDEXPIF
- BOOKMARK
- PDFINFO
- PDFOPEN

SETPPAT

The **SETPPAT** command defines a prototype pattern for use with the **SETPAT** command. A pattern is a small drawing that can be used repeatedly to fill a area. Once defined it can be used instead of, or in conjunction with, a color.

Syntax

```
/PPATkey PaintType [BBox] Xstep Ystep {PaintProc} SETPPAT
```

Where:

PPATkey	is the name of the prototype pattern to define any alphanumeric string starting with an alphabetic character.
PaintType	is an integer that specifies the paint type as one of these: **1** Colored pattern. Color is specified inside the PaintProc. **2** Uncolored pattern. Color can be defined prior to pattern usage.
BBox	is the bounding box; an array of 4 numbers in points, giving the lower-left X, lower-left Y, upper-right X, and upper-right Y of the pattern cell.
Xstep	is the horizontal spacing between cells in points.
Ystep	is the vertical spacing between cells in points.
PaintProc	is a PostScript language procedure for painting the cell.

Examples

```
/PStar 2 [-5 -5 5 5] 10 10{pop 0 5 moveto 4 {144 rotate 0 5 lineto} repeat
closepath fill} SETPPAT
```

Modes

This command is applicable in all modes.

Related commands

SETPAT, SETTPAT

SETPPATH

The **SETPPATH** command defines a list of libraries for projects. The library paths can contain the key sequences $$FOLDER. and $$PROJECT as place holders for project folders and project names. The defaults are defined in the file `/usr/xgf/src/xgfunix.run` or `x:\xgf\src\xgfdos.run`.

Use **SETPPATH** commands only in the xgfunix.run or xgfdos.run files. Adding any **SETPPATH** to a VIPP® job compromises portability. Adding any **SETPPATH** to a VI Project compromises both portability and project organization.

Syntax

`[ (path to project library 1) (path to project library 1) ... ] SETPPATH`

The parameters defined by **SETPPATH** take effect when a single **SETPROJECT** command is placed at the beginning of the VIPP® job. There can be only one **SETPROJECT** command per job and it can be placed at the beginning of the job prior to any other command referencing a resource.

When using **SETPPATH** and **SETPROJECT**, all resource libraries are redefined to those defined by **SETPPATH** for all resources types, all previous **SETEPATH**, **SETFPATH**, **SETIPATH**, **SETJPATH**, and **SETMPATH** definitions are replaced by the **SETPPATH** definition.

Project paths are divided into these three categories (scopes):

Local scope	paths that contain both $$FOLDER and $$PROJECT. These libraries can hold resources that pertain only to the project.
Folder scope	paths that contain only $$FOLDER. These libraries can hold project libraries and resources shared by projects belonging to the same folder.
Global scope	paths that contain neither $$FOLDER nor $$PROJECT. These libraries can hold resources shared by all projects.

Limitations and rules:

- In the Local scope category, $$PROJECT immediately follows $$FOLDER.
- A path containing $$PROJECT without $$FOLDER is not allowed.
- When present, $$FOLDER and $$PROJECT appears only once in each path.
- No additional path components are allowed after $$PROJECT.
- A path ending by $$FOLDER is invalid.
- There can be at least one path for each category.
- There are several paths in each category but they can be defined and grouped by category (local, folder, global) in the SETPPATH list.
- A folder or project name appears only once in the trees of directories covered by SETPPATH.
- When a resource is present with the same name in more than one scope, the order of precedence is: local, folder, global.
- To improve cross-platform portability, Xerox recommends that FOLDER and PROJECT names do not contain more than 32 characters, and only use the characters `a to z`, `0 to 9`, `dot (.)`, `dash (-)`, and `underscore (_)`.

Examples

```
[ (c:\\xgfc\\$$FOLDER.\\$$PROJECT.\\)
  (c:\\xgfc\\$$FOLDER.\\fshared\\)
  (c:\\xgfc\\gshared\\)
] SETPPATH
```

Only those resolutions supported by the current device and selectable through PostScript are allowed. Check for legal resolution values in the printer documentation. This command is supported only in level 2 devices.

Modes

This command is applicable in all modes.

Related commands

SETPROJECT

SETPROJECT

The **SETPROJECT** command activates the **SETPPATH** definition with the related project and folder names for all subsequent resource accesses. The **SETPPATH** command is defined in `xgfunix.run` or `xgfdos.run`.

When **SETPROJECT** is used it can be defined only once and placed at the beginning of the job before any other command referencing a resource.

Note: To improve cross-platform portability, it is recommended that folder and project names do not contain more than 32 characters, and use only the characters `a–z`, `0–9`, `.` period, – hyphen, and _ underscore.

Syntax

```
[ (folder_name) (project_name) ] SETPROJECT
```

Where:

folder_name	is the replacement value for $$FOLDER.
project_name	is the replacement value for $$PROJECT.

Examples

```
%!
[(projects) (holidays)] SETPROJECT
(holi1.dbm) STARTDBM
.......
```

Only those resolutions supported by the current device and selectable through PostScript are allowed. Check for legal resolution values in the printer documentation. This command is supported only in level 2 devices.

When associated with the example presented in **SETPPATH**, the **SETPROJECT** command can assign these libraries to be searched for resources during the rest of the job:

- `c:\xgfc\projects\holidays`
- `c:\xgfc\projects\fshared`
- `c:\xgfc\gshared`

Modes

This command is applicable in all modes.

Related commands

SETPPAT

SETRCD

SETRCD sets a Record Criteria Definition (RCD) for use in subsequent RPE definitions. For more information, refer to RPE Command Information and to other related RPE commands.

Note: When PCC bytes are used in **FROMLINE/RPEKEY** entries, where recpos=0 refers to the first data byte, skipping the PCC, column one of the original data file is not be treated as user data. It is used as the PCC index. However, this is not true for **GETFIELD**, **SETRCD** and **SETPCD** where recpos=0 always refers to the first byte of the record regardless of wether **SETPCC** is coded or not.

Syntax

- `/RCDkey recpos length /cond (compare string) SETRCD`
- `/RCDkey field_nr /FN /cond (compare string) SETRCD`
- `/RCDkey [ RCDkey1 RCDkey2 /bool_op ] SETRCD`
- `/RCDkey { condition statement } SETRCD`

Where

- **RCDkey** is the RCD name used in the RPE definition.
- **recpos/length** selects the record portion of the line to compare using the compare string. recpos starts with 0.
- **field_nr** specifies the field number to compare using the compare string. field_nr starts with 0 and applies to records with a field delimited structure. The field delimiter is defined by **SETDBSEP**. The default is a semi-colon `(:)`.
- **cond** specifies the test operator, for example, /eq, /ne, /ge, /gt, /le, /lt, /CIEQ, /CINE, and /HOLD. **CIEQ** and **CINE** compare the selected record portion with (comp.str) regardless of the letter case, /HOLD searches for the comparison string anywhere in the selected record portion.

  For more information, refer to Test Operators and Conditional Expressions.
- **compare string** is the reference string for the test. It can be expressed using a count value. For example, `100(*)`.
- **/bool op** is a boolean operator, such as /or, /and, and /not, used to combine several RCD definitions.
- **{ condition statement }**
- is a sequence of VIPP® commands expected to deliver a boolean. Built-in or custom variables are likely to be used in this statement.

Once defined, an RCDkey can be placed before any RPE entry causing this entry to be processed only when the resulting condition is true. When the next entry is preceded by /ELSE, it is processed only when the previous RCD is false.

When using /ENDIF, the condition and the following /ELSE can apply to several RPE entries and can be nested. Refer to the examples in the **FROMLINE** extensions for further information.

Examples

This example shows how to print a line beginning with the word TOTAL using a different font.

```
/IF_TOTAL 0 5 /eq (TOTAL) SETRCD
5 BEGINRPE
% align  rotate  init  displ  Yinit  Ydispl  recpos  length  font color
1 FROMLINE
[ 2  0  835  0  300  0  00  99  /F4  BLACK ]
10 FROMLINE
/IF_TOTAL [0  0  230  0  560  75  00  33  /F2T  BLACK ]
/ELSE [ 0  0  230  0  560  75  00  33  ./F2T  BLACK]
[ 1  0  1345  0  560  75  33  112  /F2  BLACK ]
ENDRPE
```

This example defines a new RCD that is true when IF_CND1 RCD or IF_CND2 RCD is true.

```
/IF_CND3 [ IF_CND1 IF_CND2 /or ] SETRCD
```

This example defines a new RCD that is true when IF_CND1 RCD or IF_CND2 RCD is true and IF_CND3 RCD is also true. Complex combinations can affect performance.

```
/IF_CND4 [ IF_CND1 IF_CND2 /or IF_CND3 /and ] SETRCD
```

These two examples are equivalent.

- /IF_CND1 0 10 /eq 10(*) SETRCD
- /IF_CND1 0 10 /eq (**********) SETRCD

This example is true when the word DIVISION appears anywhere in the first 100 positions of the record. Searching every record and every byte in the record for a string may affect performance.

```
/IF_CND1 0 100 /HOLD (DIVISION) SETRCD
```

This is an example using the built-in variable *CURLINE*.

```
/IF_1STLINE { CURLINE  1 eq } SETRCD
2 BEGINRPE
/HEAD RPEKEY
/IF_1STLINE
  [ ..... ]  % entry processed only if a record prefixed with "HEAD"
             % is on the first line
/ENDIF
  [ ..... ]  % entry processed for all records prefixed with "HEAD"
/BODY RPEKEY
  [ ..... ]
ENDRPE
```

Modes

This command is applicable in all modes.

Related Commands

- CURLINE
- FROMLINE
- LNCOUNT
- RPEKEY
- SETPCD

SETRES

The **SETRES** command sets the device resolution for current and subsequent pages.

Syntax

```
Res SETRES
```

Examples

```
600 SETRES
```

Only those resolutions supported by the current device and selectable through PostScript are allowed. Check for legal resolution values in the printer documentation. This command is supported only in level 2 devices.

Modes

This command is applicable in all modes.

Related commands

None

SETRPE

The **SETRPE** command invokes an RPE definition set by **RPEKEY** for the lines following the command. This command is embedded in the data stream with a Native Mode Prefix (NMP). For more information, refer to RPE Command Information and to other related RPE commands.

Syntax

```
/rpekeyname SETRPE
```

SETRPE and **SETRPEPREFIX** are mutually exclusive.

Examples

This is an example that invokes ADR0 for the following lines until the next **SETRPE** is encountered.

```
%%XGF /ADR0 SETRPE
```

Modes

This command is applicable in all modes.

Related commands

RPEKEY, SETRPEPREFIX

SETRPEPREFIX

The **SETRPEPREFIX** command enables RPE prefix mode and sets the prefix length. For more information, refer to RPE Command Information and to other related RPE commands.

Syntax

```
prefixlength SETRPEPREFIX

[prefixlength position] SETRPEPREFIX
```

Where:

prefixlength	is the length of the prefix.
position	is the position of the prefix in the record starting with 0 (default is zero).

The second syntax provides support to a prefix starting at any position in the record. Use this option to migrate **LCDS** applications that use a font index by mapping the font index byte to an **RPEKEY** definition. Refer to RPEDEF.

When RPE prefix mode is turned on, all lines in the data file begins with a prefix that invokes the matching RPE definition defined by **RPEKEY** in the JDT for this line.

When the prefix for a given line is not defined:

- The line is skipped when the prefixlength is positive
- The job aborts with an `VIPP_RPE_invalid_prefix` error message instead of /rpekeyname.

Note: Whenever an RPE prefix contains spaces it can be coded between parenthesis instead of preceded by a /. For example:

```
(REC1 45 C) RPEKEY

[ ... RPE entry ... ]
```

An RPE group is made of consecutive lines that use the same prefix but have a different last digit. This indicates that when only the last digit of the prefix changes from the previous line, the line belongs to the same RPE group.

An RPE prefix can be at least two bytes in length unless the RPE prefix is used to process an LCDS data stream using fontindex. For further information, refer to RPEDEF. This is due to the possibility of an RPE group usage. When the prefix is only one byte in length, all RPE definitions belongs to the same unique group. When this is the case, all records are processed in the same manner.

Use different prefixes in a group to specify various field attributes such as font, color, alignment, and position when different types of records appears in random order. For example, in a bank statement, credit lines can be prefixed by **LINEC** and debit lines by **LINED**. This allows the printing of the amount with a different color or in a different column.

When a data stream uses prefixes that do not follow the 'last digit' rule use **NEWGROUP**.

For an explanation on RPE processing, refer to RPEKEY and FROMLINE

Examples

This is an example of a data stream using prefixed records.

```
%!
4 SETRPEPREFIX                  % should normally be coded in the JDT
(bill.jdt) STARTLM
PER0 01.11.1989  -  31.12.1989
REF0 14153 01764960

ADR0 M. MARCEL DUPONT
ADR0 RESIDENCE "LES MIMOSAS"
ADR0 PLACE DE LA GARE, 44
ADR0 2323 SAGEX

DFA0 23.01.1990
DPY0 23.02.1990
DRE0  7.12.1989
CRN0 068.025.000 COMMUNS IMMEUBLE

LFA0 14 COMMUNS         81978   30248   30144    104            44,40
LFA0 41 EAU TARIF I       491   12869   12714    155  0,2800    43,40
LFA0    TAXE DE BASE                             2,0   7,692    15,35
LFA0    LOCATION COMPTEUR                        2,0    2,00     4,00
LFA0 82 GAZ CHAUFFAGE    3500   40972   40126   8798  0,0330   290,35
LFA0    TAXE DE BASE                             2,0  16,666    33,35
LFA1 TOTAL S.I.                                               430,85
LFA0 41 ASSAINISSEMENT    491                    155    0,25    38,75

MNT0 MONTANT A PAYER       469,60
BRE0 00 00014 15301 76496 00120 01019
OLI0 0100000469609>000001415301764960012001019+
%%EOF
```

This example of `[prefixlength position]` **SETRPEPREFIX**, defines the prefix as 1 byte long and located at position 132 on the record. The default position is zero.

```
[ 1 131 ] SETRPEPREFIX
```

Modes

This command is applicable in all modes.

Related commands

RPEKEY, SETRPE

SETSKIP

The **SETSKIP** command is an alternative to **SETPCC**. In addition, it provides a mechanism to handle widow and orphan printing situations. **SETSKIP** and **SETPCC** are exclusive.

Syntax

```
[ skip_cond1 [ pre-skip  print-action  post-skip  BOF  TOF ]
  skip_cond2 [ pre-skip  print-action  post-skip  BOF  TOF ]
  ......
  skip_condn [ pre-skip  print-action  post-skip  BOF  TOF ]
  ()          [ pre-skip  print-action  post-skip  BOF  TOF ]   %default
] SETSKIP
```

Where:

skip_condx	is one of the following: A **string** of any length enclosed in parenthesis. When the beginning of a line matches the string then the associated skip action occurs. The string remains as part of the printable data. An **RCDkey**. When the RCD condition is true for that line, then the associated skip action occurs.
pre_skip	is one of the following: A **number of lines** A **key** to be assigned later by **SETVFU**
print_action	is one of the following: **True**, print the record **False**, do not print the record
post_skip	is one of the following: A **number of lines** A **key** to be assigned later by **SETVFU**
BOF	defines the bottom threshold. A skip to next page, before print, occurs when the number of remaining lines on the page after pre-skip is less than this number. The number of lines per page is defined by the **SETGRID** command.
TOF	defines the top threshold. If not zero, **TOF** forces that line to be the target line (regardless of the pre-skip) when a page transition occurs during pre-skip process.

Examples

In this example any line starting with 0001 can cause a skip to channel one.

```
[ (0001) [ /SK1 true 0 0 0 ]
() [ 1 true 0 0 0 ]
] SETSKIP
```

Modes

This command is applicable in all modes.

Related commands

- SETGRID
- SETPBRK
- SETPCC
- SETVFU

SETTAB

The **SETTAB** command enables tab processing and defines tab spacing in character units.

Note: **SETTABS** is used to define the tab width in current units. Tab width is the space allocated each time that the tab character is processed. **SETTABS** is used to set a series of tab stops that can be used in a text block, for example, to align columns. With **SETTABS**, the print position skips to the next tab stop.

Syntax

```
charcount SETTAB
```

Where:

charcount	is the number of characters used to position tabs.

Examples

This example defines a tab stop every eight characters.

```
8 SETTAB
```

Note: Use SETTAB only when required, as it can affect performance.

Note: Use the **SETUTAB** command with proportional fonts.

Modes

This command is applicable in basic line mode (no RPE).

Related commands

SETUTAB, SETTABS

SETTABS

SETTABS enables tab processing in text blocks that are printed with **SHP** or **SHMF**. When the **SETTABS** command has been coded, **SHP** or **SHMF** can honor tab characters inside a string.

- When tabs are enabled, only left alignment is supported for **SHP** and **SHMF**. With **SHP**, line wrapping does not take into account any tabs, so each line with tabs can fit in the defined column width.
- When you set option +20, several lines can be placed in one text block.

Note: **SETTABS** is used to define the tab width in current units. Tab width is the space allocated each time that the tab character is processed. **SETTABS** is used to set a series of tab stops that can be used in a text block, for example, to align columns. With **SETTABS**, the print position skips to the next tab stop.

Syntax

```
[ [ tab_stop1 ] [ tab_stop2 ] .... default_tab_space ] SETTABS
```

Where:

[tab_stopX]	Can be of the form: **[align position]**: No leader, no align char **[align position null null]**: No leader, no align char **[align position (leader) null]**: Leader, no align char **[1 position null (align_char)]**: No leader, align char **[1 position (leader) (align_char)]**: Leader, align char
align	is one of: **0**: Left tab **1**: Right tab **2**: Center tab
position	Tab position in current units relative to the current main print position.
default_tab_space	Applied when all tab stops have been exhausted.

Examples

```
[ [ 0   20 ]
  [ 1   50 ]
      [ 1   100 null (.) ]
  [ 2   150 (*-*) null ]
  [ 1   100 (*-*) (.) ]
] SETTABS
```

Modes

This command is applicable in all modes.

Related commands

SETTAB, SETUTAB

SETTRAN

SETTRAN sets the transparency for all subsequent marking commands including images. This command is only effective in VI eCompose and APPE RIP. On a PS RIP it is emulated by an opaque tint.

Syntax

```
TRlevel SETTRAN
```

Where:

TRlevel	is a real number between 0 and 1 specifying the transparency level. Only effective in VI eCompose and APPE RIP. On a PS RIP it is emulated by an opaque tint.

Examples

This example sets the transparency to 50 %:

```
.5 SETTRAN
```

Modes

This command is applicable in all modes.

Related commands

SETCOL, SETTXC, SETGEP

SETTPAT

Use **SETTPAT** to create a patterned ink pattern that includes a repeating text string. The text string can contain variable text. A text pattern fills an area by repeating the text. Once defined, it can be used instead of a color to paint an object.

Syntax

To define a patterned ink without a background color:

```
/tpatname (text) /fname fsize htr vtr rot Tcolor SETTPAT
```

To define a patterned ink with a background color:

```
/tpatname (text) /fname fsize htr vtr rot [Tcolor Bcolor] SETTPAT
```

Where:

/tpatname	is the name of the text pattern
(text)	is the text to be spread
/fname	is the name of the font to use
fsize	is the font size in points
htr	is an horizontal adjustment in points
vtr	is a vertical adjustment in points
rot	is the rotation angle in degrees of the text in the pattern
Tcolor	is the color of the text
Bcolor	is the color of the background

When the pattern is invoked to fill a shape, the shape is first painted with the Bcolor and the text pattern using Tcolor is painted over it.

When using the first syntax, Bcolor is not specified, so no background is painted, except in the following cases:

- When the font is a GL or CR font, a white background is painted
- When Tcolor is a UV/IR ink, the background is painted using the Alternate ink
- When Tcolor is `ARTBLACK_A` or `ARTBLACK_B`, the background is painted with a contrasting black

Examples

Regular text pattern:

```
/TVipp (VIPP) /NHEB 20 15 15 45 [ RED YELLOW ] SETTPAT
```

Fluorescent text pattern:

```
/TVipp (VIPP) /NHEB 28 5 2 45 UV_OLIVE1 SETTPAT
```

GlossText text pattern:

```
/TVipp (VIPP) /NeueClassic-GL-24 GLT 0 0 45 GL_Magenta SETTPAT
```

Art black text pattern:

```
/TVipp (VIPP) /NHEB 20 5 5 45 ARTBLACK_A SETTPAT
```

The following example fills the box with the text VIPP® rotated by 45 degrees:

```
100 200 400 150 TVipp DRAWB
```

Modes

This command is applicable in all modes.

Related commands

SETPPAT, SETPAT

SETTXB

The **SETTXB** command activates a background for all subsequent text imaged using an **SHx** command.

Syntax

```
/BATkey SETTXB
null SETTXB
```

Where:

BATkey	represents any key previously defined with **SETBAT**.
null	disables a currently active **BATkey**.

Examples

This example illustrates how to activate underlining and light gray.

```
UNDL SETTXB
PB_LT SETTXB
```

Modes

This command is applicable in all modes.

Related commands

SETBAT, SHX, INDEXBAT

SETTXC

The **SETTXC** command sets the color and pattern for all subsequent printed text and **ICALL** commands.

Syntax

```
Colorkey SETTXC
PATkey SETTXC
[ Colorkey PATkey ] SETTXC
null SETTXC
(ColorKey~Tintlevel) SETTXC
(ColorKey#TRlevel) SETTXC
```

Where:

Colorkey	is the name of the color used for subsequent text and **ICALL** commands.
PATkey	is the name of the pattern used for subsequent text and **ICALL** commands. The color used is the last plain color is defined in the VIPP® job.
[Colorkey PATkey]	sets the color and pattern used for subsequent text and **ICALL** commands. Colorkeys and patterns are defined in the `/usr/xgf/src/xgf.gep` file listed in Standard lists, tables, keys, and attributes in the *FreeFlow VI Compose User Guide*.
null	sets the area being painted to transparent, rather than filling with a color.
Tintlevel	is a real number, between 0 and 1, specifying the tint level.
TRlevel	is a real number, between 0 and 1, specifying the transparency level.

Note: TRlevel is effective only in VI eCompose and APPE RIP. On a PS RIP, TRlevel is emulated by an opaque tint.

The default color is BLACK. The predefined Colorkeys include:

- BLACK
- XLIGHT
- LMEDIUM
- DARK
- RED
- WHITE
- LIGHT
- MEDIUM
- XDARK
- BLUE
- DMEDIUM
- GREEN

To define new Colorkeys, edit the `/usr/xgf/src/xgf.gep` file and add lines using this syntax:

`/Colorkey color_definition SETCOL`

Where:

color_definition	can be a number between 0 (black) and 1 (white) to define a gray level: `/LIGHT .9 SETCOL` Refer to the SETCOL description for possible syntaxes regarding color_definition.

The default pattern is no pattern. Pattern keys are defined using the **SETPAT** and **SETPPAT** commands. Predefined Patterns include:

- Diamond
- Diamond2
- Cross
- Grid
- Honey
- VIPP®
- Strike
- Spot
- Spot2
- Star
- Star1
- Wave

For more information about color tints, refer to Color Tints.

Examples

These examples illustrate how to apply transparency to a color.

```
(RED#.5) SETTXC
20 20 300 400 (BLUE#.7) DRAWB
```

Modes

This command is applicable in all modes.

Related commands

- INDEXCOLOR
- SETPAT
- SETCOL
- SETTRAN

SETTXS

Use the **SETTXS** command to activate or deactivate superscript and subscript on a case-by-case basis between **SH** commands. Subscript and superscript are handled as text attributes, as are font, color, and background.

Syntax

```
sst_param SETTXS
[ Ydispl1 Fstring ] SETTXS
```

Where:

sst_param	can be one of these: **A numeric value** representing a vertical displacement in current units from the current vertical position, positive for superscript and negative for subscript. In this case, font size is handled by the font commands (**SETFONT**, **INDEXFONT**). **/SUP** for automatic superscript activation. Smaller font and offset computed from the current font settings. **/SUB** for automatic subscript activation. Smaller font and offset computed from the current font settings. **null** indicates end of sub/superscript.
Ydispl	defines the vertical offset as a factor of the current font size. It can be positive (superscript) or negative (subscript). The expected value range is [-1 +1].
Fshrink	defines the shrink factor to apply to the current font. The expected value range is [>0 +1].

Examples

These examples print March, 17th.

```
100 3000 MOVETO
/NHE 16 SETFONT
(March, 17) SH
/NHE 8 SETFONT
20 SETTXS (th) SH null SETTXS

100 3000 MOVETO
/NHE 16 SETFONT
(March, 17) SH
/NHE 8 SETFONT
/SUP SETTXS (th) SH null SETTXS   %uses VIPP computed placement
```

This example sets a superscript with a vertical offset equal to 40 % of the current font size, and current font shrink at 60 %.

```
[.4.6] SETTXS
```

Modes

This command is applicable in all modes.

Related commands

INDEXSST, SETFTSW

SETUNIT

SETUNIT sets the unit of measure for all subsequent commands except **SETGEP**.

Syntax

`unit SETUNIT`

Where:

unit	Can be one of these values: **DOT3**: 1/300 in. is the default value **PELS**: 1/240 in. **POINT**: 1/72 in. **CM**: (centimeter) **MM**: Millimeter **INCH**: in.

PELS can help when converting AFP resources to VIPP® resources.

Important: **SETUNIT** is not a VIPP® start-up command. When the job begins with a **SETUNIT** statement, place an **XGF** command before the **SETUNIT** statement.

Modes

This command is applicable in all modes.

Related commands

SETGUNIT

SETUTAB

The **SETUTAB** command enables tab processing and defines the tab length in current units.

Syntax

```
unitcount SETUTAB
```

Where:

unitcount	is the number of units used to position tabs.

Examples

This example defines a tab stop every 50 current units.

```
50 SETUTAB
```

Note: Use SETUTAB only when necessary as it can affect performance.

Note: Use the **SETUTAB** command with proportional fonts.

Modes

This command is applicable in basic line mode (no RPE).

Related commands

SETTAB, SETTABS

SETV2HCONV

SETV2HCONV defines a set of conversion tables used when the appropriate **SHP** or **SHMF** option (+4000) is set. This command is used with multi-byte fonts.

Syntax

```
[ /v2h_cvname1 [ (char1) (conv1) ... ]
/v2h_cvname2 [ (char1) (conv1) ... ]
....
] SETV2HCONV
```

Where:

/v2h_cvname1	is an unrestricted name that can be used as the fourth parameter in an entry of the command, **SETCJKENCMAP**.
(charX) (convX)	defines the conversion pairs. When charX is encountered in the string argument of **SHP/SHMF** it is replaced by convX.

Examples

```
[ /SJIS [ (1) <88EA> (2) <93F1> (3) <8E4F> (-) <815C> ] ] SETV2HCONV
```

Modes

This command is applicable in all modes.

Related commands

- SETFONT
- INDEXFONT
- SHP and SHp
- SHMF, SHMf, and SHmf

SETV2HTABLE

SETV2HTABLE defines a list of characters to be processed horizontally when a vertical font is active and the appropriate **SHP** or **SHMF** options, (+1000/+2000), are set.

Syntax

```
[ def_scale (char1) [(char2) sca2] ... ] SETV2HTABLE
```

Where:

def_scale	is a default scale factor to be applied to the horizontal width of each character processed horizontally with method 0.
charX	is a character to be processed horizontally.
scaX	is an optional individual scale factor to be substituted to the default factor only for that character.

Examples

```
[ 1.0 [(0) 1.2] (1) (2) (3) (4) (5) (6) (7) (8) (9) ] SETV2HTABLE
```

Modes

This command is applicable in all modes.

Related commands

SHP, SHMF, SHMf, and SHmf

SETVAR

The **SETVAR** command sets a variable. When defined, a variable reference is used in any command in place of a string, number, or array operand.

Syntax

```
/VARname value SETVAR
/VARname value /INI SETVAR
/^name value SETVAR
value /VARname /SWP SETVAR
```

Where:

VARname	can be any ASCII alphanumeric string starting with the prefix VAR. The maximum length is 20 characters. The dash – and underscore _ characters are allowed. However, dot . is allowed only when the variable is not used with **VSUB**.
value	is a string, a number, or an array.
/INI	when this option is used, the variable will be set only if it does not already exist. A variable that already exists will not be re-initialized when the /INI option is used. Use this option in a DBM to initialize a variable only during the first call.
^name	is a variable name generally associated with an XML job ticket.
/SWP	/SWP is a switch used to obtain the value from the stack. This is mainly intended to be used in a **FROMLINE/RPEKEY** align proc to capture and use the RPE string on the stack, as in this example:

```
...
[ { /VARqrc /SWP SETVAR [ /AC VARqrc ] QRCODE }
rot Xi Xd Yi Yd pos length /Font Color ]
```

In this example of an RPE entry, **SETVAR** captures the data string extracted by pos length and uses the variable to produce a **QRCODE** symbol.

Numeric strings accommodate large numbers up to 40 digits, 25 digits for the integer part and 15 digits for the decimal part. In a numeric string the negative sign and the decimal delimiter are defined by the parameters /DecimalPoint and /NSign, and can occur anywhere in the string.

It is mandatory to set these parameters with appropriate values to ensure accurate results. Defaults are defined in the file `/usr/xgf/src/xgf.def`. When the variable is processed by **ADD**, SUB, MUL, or DIV characters in the numeric string other than these two plus the digits 0–9 are discarded.

The initial length of the string defined by **SETVAR** is automatically extended up to 40 digits when needed.

Reals and integers are used only for small values <= 99999 for instance the implementation of a counter. The decimal delimiter, when present, is always the point `(.)`. The negative sign, when present, is always the minus `(-)` and is the first character.

Real and integers can be converted to strings using the **VSUB** command. Then they can be formatted for printing using the **FORMAT** command.

In an XML job, XML variables are automatically assigned a full name depending on their name and the current XML path. In addition they are automatically reset when parent nodes are crossed.

XML variables start with ^ and were designed for use in an XML job ticket (`.xjt`). They do not require

explicit initialization. When not initialized they are automatically pre-set to an empty string or zero when they appear to be updated using ++, –, ADD or SUB.

Assuming the current XML path starts with: `^invoices^invoice^customer`, the variable: ***^customer^area*** will be assigned a full variable name of:

`^invoices^invoice^customer^area` and will be automatically reset whenever a new `<customer>` or `<invoice>` tag is crossed.

As to XML tree variables, XML variables can be accessed through their full name or any unambiguous sub-name at the time they are referenced.

Examples

```
/VAR.date (December, 12th. 1993) SETVAR

/VARX1 200 SETVAR

/VARY1 300 SETVAR

VARX1 VARY1 MOVETO

VAR.date SHL

($$VARX1.) VSUB SHL

($$VARX1.) VSUB (:#######:) FORMAT SHL
```

In this example, **SETVAR** initializes a variable in a DBM. Because /INI was used to define the variable, it will not be re-initialized for each record, allowing the **/VAR.COUNT ++** to increment the count. When /INI is not used the count is set to zero for each new record.

```
/VAR.COUNT 0 /INI SETVAR
IF VAR.COUNT 50 eq
   { (slipsheet) SETMEDIAT
     0 NL PAGEBRK
     /VAR.COUNT 0 SETVAR
   }
ENDIF
/VAR.COUNT ++
.....
```

In this example **SETVAR** is used to define a procedure. This procedure prints an address block. It is executed each time the variable name is coded in the VIPP® code.

This example also demonstrates the usage of **SHP** to print the address lines, `Addr1` and `Addr2`. When one of the address lines is empty, VI Compose will not move the print position down. This allows the address block to adjust for empty lines.

```
/VARadd_block
{
  ($$Fname. $$Lname.) VSUB SHL
   Addr1 0 SHP
   Addr2 0 SHP
  ($$City. $$State. $$Zip.) VSUB SHL
}  SETVAR

% When needed
300 3000 MOVETO
VARadd_block
```

The VIPP® language allows for incrementing and decrementing of variables. There are differences in applying this to integers and string values.

Integers:

```
/VARNumber 0 SETVAR       % Value equals 0
/VARNumber ++             % Value now equals 1
/VARNumber 3 ADD          % Value now equals 4
/VARNumber --             % Value now equals 3
```

String values:

- When you want the number to be zero filled, then declare a number with the appropriate number of zeros.
- When you do not want zero filled numbers, use FORMAT to format the number.

```
/VARNumber  (000) SETVAR          % value is 000
/VARNumber ++                     % Value is now 001
/VARNumber 7 ADD                  % Value is now 008
/VARNumber --                     % Value is now 007
/VARNumber -2 ADD                 % Value is now 005
/VARNumber (7) ADD                % Value is now 012
```

Modes

This command is applicable in all modes.

Related commands

SETMPATH, STOREVAR, VSUB

SETVFU

The **SETVFU** command defines a Vertical Format Unit (VFU) table, channel-skip to line-number assignments for the current PCC definition.

Syntax

```
[ /skip-key1 line-number1 /skip-key2 line-number2 ... ] SETVFU
[ ... /skip-keyx [line-numberx1 line-numberx2] ... ] SETVFU
```

Where:

/skip-key	the skip keys as defined in the PCC definition, for further information, refer to Standard lists, tables, keys, and attributes in the *FreeFlow VI Compose User Guide*.
line-number	the matching line numbers for the current job. A negative line number forces a skip to the specified line on a new page. Several line numbers are associated with a single skip-key. The lines will be skipped to in order, starting with the first one greater than or equal to the current line.

Refer to **BEGINPCC** for further information on PCC definitions.

SETVFU is coded in a JDT.

Examples

```
[ /SK1 -1 /SK2 35 /SK9 66 ] SETVFU
```

Modes

This command is applicable in line mode.

Related commands

BEGINPCC, ENDPCC, SETPCC

SETZEBRA

The **SETZEBRA** command enables zebra, or greenbar, printing for all subsequent pages. Zebra printing is the printing of background shaded boxes.

Syntax

```
Colorkey lines-with lines-without SETZEBRA
Colorkey lines-with lines-without /V SETZEBRA
```

Where:

Colorkey	defines the color. WHITE indicates no zebra printing.
lines-with and lines-without	define the sequence of lines with and without zebra printing.
/V	varies the number of zebra lines depending on the number of lines printed on the page. Without /V the number of zebra lines is constant and controlled by the number of lines per page set by the **SETGRID** command.

Zebra boxes are computed from the current grid and margins. The default zebra, or greenbar, pattern is set in the `/usr/xgf/src/xgf.def` file. Colorkeys are defined in the `/usr/xgf/src/xgf.gep` file listed in Standard lists, tables, keys, and attributes in the *FreeFlow VI Compose User Guide*. The predefined keys include:

- BLACK
- XLIGHT
- LMEDIUM
- DARK
- RED
- WHITE
- LIGHT
- MEDIUM
- XDARK
- BLUE
- DMEDIUM
- GREEN

Edit the `/usr/xgf/src/xgf.gep` file to add new Colorkeys

Examples

```
LIGHT 3 3 SETZEBRA
```

Modes

This command is applicable in all modes.

Related commands

- SETGRID
- SETMARGIN

- SETPAT
- SETTXC

SHx Commands

The generic term SHx, is used throughout this, and other Variable Information Suite Documentation, to indicate that one or more of the commands listed here can be used:

- SHC and SHc
- SHJ and SHj
- SHL and SH
- SHMF, SHMf, and SHmf
- SHP and SHp
- SHR and SHr
- SHT and SHt
- SHX

SHC and SHc

SHC and **SHc** prints data centered on the main (**SHC**) or secondary (**SHc**) print position. **SHC** resets the main horizontal print position to the last value specified by **MOVETO** and forwards the vertical print position by the **SETLSP** value. **SHc** sets the secondary print position at the point reached after printing the data.

Syntax

```
(printable data) SHC
(printable data) SHc
```

Modes

These commands are applicable in native mode, line mode, and database mode.

Related commands

- MOVETO
- MOVEH
- MOVEHR
- SETLSP
- SHJ and SHj
- SHL and SH
- SHMF, SHMf, and SHmf
- SHR and SHr

SHIFT

The **SHIFT** command defines horizontal (X) and vertical (Y) shift values in current units for front and back pages. All of the page content is shifted from the bottom left origin on the horizontal and vertical axis according to the specified values, either positive or negative. These values always refer to the bottom left origin of the portrait sheet regardless of the current orientation.

Syntax

```
Xfront Yfront Xback Yback SHIFT
```

Modes

This command is applicable in all modes.

Related commands

SETPARAMS,SETVAR, SHX

SHIFTDATE

SHIFTDATE sets the current date variables to the current date increased by the specified value. Thus, with a value of zero, **SHIFTDATE** is equivalent to **GETDATE**. The current date is restored by a new **GETDATE** or by the next **PAGEBRK**. Refer to **GETDATE** for more details.

Syntax

```
shift /unit SHIFTDATE
```

Where:

shift	is an integer indicating the value in units specified by /unit
/unit	is one of these values: **/S** seconds **/M** minutes **/H** hours **/D** days **/W** weeks

When using **SHIFTDATE**:

- Negative values are acceptable.
- Multiple **SHIFTDATE** commands can be placed on a single page, all computing from the current date.
- **SHIFTDATE** does comprehend leap years.
- Use **PAGEBRK** or **GETDATE** to reset the date variables to the current system date.

Examples

```
15 /D SHIFTDATE                          % set current date plus 15 days
($$DD_MO./$$D_DD./$$D_YYYY.) VSUB SH     % print date plus 15 days
```

The following example will print, This will expire in 30 days, on 07/28/2005.

```
30 /D SHIFTDATE
(This will expire in 30 days, on $$D_MOL./$$D_DD./$$D_YYYY.) VSUB 0 SHP
PAGEBRK
```

Modes

This command is applicable in all modes.

Related commands

GETDATE,SPOOLNAME,SETPARAMS

SHJ and SHj

SHJ and **SHj** print data justified from the main (**SHJ**) or secondary (**SHj**) print position to the width specified by the last **SETCOLWIDTH**.

Inter-word justification is applied when the string contains spaces, if not, inter-character justification is applied.

`SHJ` resets the main horizontal print position to the last value specified by **MOVETO** and forwards the vertical print position by the **SETLSP** value.

SHj sets the secondary print position at the point reached after printing the data.

Syntax

```
(printable data) SHJ
(printable data) SHj
```

Note: When the width that is set by the **SETCOLWIDTH** command does not fit the length of the printable data, the printed text can overlap.

Modes

These commands are applicable in all modes.

Related commands

- MOVETO
- MOVEH
- MOVEHR
- SETCOLWIDTH
- SETLSP
- SHL and SH
- SHC and SHc
- SHMF, SHMf, and SHmf
- SHR and SHr

SHL and SH

SHL and **SH** print data left aligned at the main (**SHL**) or secondary (**SH**) print position.

SHL resets the main horizontal print position to the last value specified by **MOVETO** and forwards the vertical print position by the **SETLSP** value.

SH sets the secondary print position to the point reached after printing the data.

Syntax

```
(printable data) SHL
(printable data) SH
```

Examples

This example prints: Hello World it's me.

```
(Hello World) SH ( it's me) SH
```

Modes

These commands are applicable in all modes.

Related commands

- CASETI
- GETINTV
- MOVETO
- MOVEH
- MOVEHR
- SETLSP
- SHC and SHc
- SHJ and SHj
- SHMF, SHMf, and SHmf
- SHR and SHr

SHMF, SHMf, and SHmf

SHMF, **SHMf** and **SHmf** print data at the main (**SHMF/SHMf**) or secondary (**SHmf**) print position. Use these commands to switch between fonts and colors in the printable data by using a font/color switch sequence. The font/color switch prefix is defined in the `/usr/xgf/src/xgf.def` file using **SETFTSW**. // is the default.

SHMF resets the main horizontal print position to the last value specified by **MOVETO**, and forwards the vertical print position by the **SETLSP** value.

SHmf sets the secondary print position to the point reached after printing the data.

Using **SHMF** allows you to print line mode data streams with embedded font switch sequences. This can be useful when migrating **XES** data streams to VIPP®.

SHMf behaves like **SHMF**, however as with the **SHP** command, it does not forward the vertical print position when printable data is an empty string.

Syntax

```
(printable data) align SHMF
(printable data) colwidth align SHMF
```

Where:

align is the alignment option, and can be specified as one of these values, the plus (+) sign indicates that the value can be combined with other values where appropriate, it is not part of the syntax:

0 left

1 right

2 center

3 justify

4 alignment on decimal point (the decimal point is the character currently defined by the /DecimalPoint parameter)

+100 fit-in-width

+200 stretch-in-width

Note:

- **fit-in-width**: When this option is enabled, the current font is scaled on the horizontal axis so that the printed string fits into the current column width, as specified by the **SETCOLWIDTH** command or the colwidth operand. Scaling is performed only when the natural width exceeds the column width. If the natural width does not exceed the column width, no scaling is applied.
- **stretch-in-width**: When this option is enabled, the current font is scaled on the horizontal axis so that the printed string fills the current column, as specified by the **SETCOLWIDTH** command or the colwidth operand.

The following example prints the name in an address box that is left-aligned and constrained within a width of 500 dots:

```
Mrs. Mary-Eleonore de Bourbon l'Archambaud 500 100 SHMF
```

For Specialty Imaging Gloss and Correlation text, to control the width of the SI effect, use the **stretch-in-width** option. Text is padded to the defined column width. For more information, refer to GlossMark and Correlation fonts (GL/CR) in the *Programming Tips* topic.

- **+1000**: Set vertical-to-horizontal processing, method 0, horizontal in vertical
- **+2000**: Set vertical-to-horizontal processing, method 1, 90degree clockwise rotation
- **+4000** Set conversion
- **+5000**: Set conversion and vertical-to-horizontal, method 0, horizontal in vertical
- **+6000**: Set conversion and vertical-to-horizontal, method 1, 90degree clockwise rotation
- **+10000**: Apply bi-directional transform on each line
- **+20000**: Apply bi-directional transform on each line and switch to a different font for left-to-right characters
- **+30000**: Apply bi-directional transform on each line and substitute European digits for Hindi digits

Note: The list of characters to be changed from vertical to horizontal is previously set by the **SETV2HTABLE** command. The conversion table for the related encoding are previously set by the **SETV2HCONV** command. Default lists and tables are defined in the configuration file: `xgf/src/cjk.def`.

colwidth	is the column width for justification (align=3). When colwidth is not specified, the value defined by a previous **SETCOLWIDTH** is used. When colwidth is specified, it overrides and replaces the value defined by a previous **SETCOLWIDTH**.

Examples

This example prints Text using font 1 Switch to font 2 blue and Font 3.

```
/1 /NHE 18 INDEXFONT
/2 /NHE 24 INDEXFONT
/3 /NHEO 18 INDEXFONT
/A BLACK INDEXCOLOR
/B BLUE INDEXCOLOR
200 200 MOVETO
(//1Text using font 1 //2//BSwitch to font 2 blue //3and Font 3) 0 SHMF
```

Note: For more information, refer to **INDEXFONT** and **SETFTSW**.

TIP

Use the appropriate **SHx** commands when font or color switching is not required.

Modes

These commands are applicable in all modes.

Related commands

- INDEXBAT
- INDEXFONT
- MOVETO
- MOVEH
- MOVEHR

SHP and SHp

The **SHP** command prints data at the main print position and **SHp** prints data at the secondary print position with line wrapping.

These commands behave the same as **SHMF**. In addition, they perform word wrapping on spaces and dashes in printable data according to the width defined by **SETCOLWIDTH** and the line spacing defined by **SETLSP**. Printable data has a long string including up to 65,535 characters dynamically formatted in a paragraph by **SHP**.

SHP resets the main horizontal print position to the last value specified by **MOVETO** and forwards the vertical print position reached after the last line by the **SETLSP** value.

Syntax

```
(printable data) align SHP
(printable data) colwidth align SHP
[ (printable data1)  (printable data2) .... ] align SHP
[ (printable data1)  (printable data2) .... ] colwidth align     SHP
(printable data) [width height spacing] align SHP
[ count (printable data)) ] align SHP
```

Where:

align is the alignment option, and can be specified as one of these values, the plus (+) sign indicates that the value can be combined with other values where appropriate, it is not part of the syntax:

0 left

1 right

2 center

3 justify with last line aligned left (re-qualifying)

4 justify with last line aligned right

5 justify with last line centered

6 justify all lines

+00 treat new line characters (0x0A) as spaces

+10 Strip duplicate blanks between words at end of lines only

+20 treat new line characters (0x0A) as end of line

+30 Strip duplicate blanks between words and treat new line characters (0x0A) as end of line

+40 do not wrap on the dash character

+000 wrap on Roman words and between any Asian characters

+100 wrap-down according to Asian rules

+200 wrap-up according to Asian rules

+300 wrap-up + hanging punctuation according to Asian rules

+400 hanging punctuation according to Asian rules

+600 apply bi-directional transform on each line

+700 apply bi-directional transform on each line and switch to a different font for left-to-right characters

+800 apply bi-directional transform on each line and substitute European digits for Hindi digits

+1000 set vertical-to-horizontal processing, method 0, horizontal in vertical

+2000 set vertical-to-horizontal processing, method 1, 90 degree clockwise rotation

+4000 set conversion

+5000 set conversion and vertical-to-horizontal, method 0, horizontal in vertical

+6000 set conversion and vertical-to-horizontal, method 1, 90 degree clockwise rotation

Note: Asian language and formatting rules for the related encoding is previously set by the **SETCJKENCMAP** and **SETCJKRULES** commands.

The list of characters to be changed from vertical to horizontal is previously set by the **SETV2HTABLE** command. The conversion table for the related encoding are previously set by the **SETV2HCONV** command.

Default lists and tables are defined in the configuration file `xgf/src/cjk.def`.

For vertical alignment of the paragraph on the current print position, use the following settings:

- +0 top alignment; default
- +10000 bottom alignment
- +20000 center alignment

To fit or stretch text within an area or object, use the following settings:

- +100000 fit line in width
- +200000 stretch line in width

The two preceding options are equivalent to the **SHMF** options +100 and +200 and are intended for use with the +20 option . Each line that is delimited by the new line characters 0x0A is compressed, according to the option, instead of being wrapped.

- +300000 fit line in width, preserve wrapping
- +400000 stretch line in width, preserve wrapping

The two preceding options are similar to options +100000 and +200000, except that word wrapping is applied before the compression.

For Specialty Imaging Gloss and Correlation text, to control the width of the SI effect, use the stretch line in the width option. To control vertical padding, use the text-in-box syntax with spacing=0. Text is padded to the defined box width and height. For more information, in the Programming Tips topic, refer to GlossMark and Correlation Fonts (GL and CR)

colwidth	is the column width used for word wrapping. When colwidth is not specified, the value defined by a previous **SETCOLWIDTH** is used. When colwidth is specified, it overrides and replaces the value defined by a previous **SETCOLWIDTH**.
width	is the horizontal constraint used as colwidth.
height	is the additional vertical constraint.
spacing	is the line spacing factor in relation to the font size, examples: **0** vertical justification. This option causes the line spacing to be adjusted so that the text fills the box entirely. **1** single line **1.5** one and one half **2** double spacing **.5** half line
count	is an integer used to indicate the number of times printable data is duplicated.

With the third and fourth syntax examples, **SHP** concatenates the specified list of strings and formats the resulting string, the sum of all lengths cannot exceed 65,535 characters.

The syntax, printable data [width height spacing] align **SHP**, forces a block of text to fit into a pre-defined box by automatically adjusting the font size.

When using alignment 3, any word that causes the line to exceed the column width, but is within a specific percentage, prints with that line. The threshold is by default set to 0.05 % of the column width. This value is stored in a variable named *VAR.SHP_OVER*. Use the **SETVAR** command to change this value.

Note: When the printable data area is empty, the vertical print position is not forwarded. To make the text flow from frame to frame, or from page to page, combine an **SHP** command with a **SETLKF** command.

Examples

This example illustrates the use of database records to print a name and address block, which can contain blank fields. These are the example database records:

Fname,Lname,addr1,addr2,city,state,zip

David,Kirk,101 Continental Blvd., Suite 1, El Segundo, CA, 90245

Mary, Smith, 5 Euclid Lane, Santa Monica, CA 90403

The example above prints this information:

David Kirk

101 Continental Blvd.

Suite 1

El Segundo, CA 90245

Mary Smith

5 Euclid Lane

Santa Monica, CA 90403

The following example uses the additional align parameters:

```
DOT3 SETUNIT
200 200 MOVETO
/VARtextexample ( The lazy brown dog jumped over the sleeping fox ) SETVAR
```

Inserting this syntax into the example.

```
VARtextexample 300 0 SHP
```

This creates a one inch wide paragraph that is similar to this:

```
   The
lazy brown
dog jumped
over the
sleeping
fox
```

Inserting this syntax into the example...

```
VARtextexample 300 10 SHP
```

This creates a one inch wide paragraph that is similar to this:

```
The lazy
brown dog
jumped over
the sleeping
fox
```

This example produces the name and address as shown below:

```
(John R. Doe\n1405 Ocean Drive\nEl Segundo, CA 90245) 20 SHP
```

John R. Doe

1405 Ocean Drive

El Segundo, CA 90245

This example repeats “Hello World” 50 times, wrapped in a paragraph:

```
[ 50 (Hello World ) ] 0 SHP
```

These examples can be combined with other align options:

```
(... text block ...) 20002 SHP % align vertically and horizontally
(... text block ...) 41 SHP % align left and do not wrap on dash
```

Modes

This command is applicable in all modes.

Related commands

- MOVETO
- MOVETO
- MOVEH
- SCALL
- SETCOLWIDTH
- SETLSP
- SETVAR
- SHMF, SHMf, and SHmf
- VSUB
- INDEXALIGN
- INDEXLSP

SHPATH

SHPATH prints a text string on a path previously defined by commands using the **TPATH GEPkey**. The current text attributes, font, color, background, and so on are used to print the text. Attribute switches are allowed inside the text string.

Syntax

```
(string) baseline_offset text_start align SHPATH
(string) [baseline_offset path_offset] text_start align SHPATH
(string) baseline_offset [text_start text_end] align SHPATH
(string) [baseline_offset path_offset] [text_start text_end] align SHPATH
```

Where:

(string)	is the text string to be printed
baseline_offset	is the offset in current units, positive or negative between the text baseline and the path
path_offset	is similar to baseline_offset but it offsets the path instead of the characters.
text_start	is the offset in current units from the beginning of the path to the point where the text starts. When equal to 'null' the text starts where the previous text ended.
text_end	is the offset in current units from the beginning of the path to the point where the text ends. If text_end is omitted then: When the align option +40 is set (typically for a closed path) the entire path is available for text, text_start only defines the start point. When the align option +40 is not set (typically for an open path) the path is available from text_start to the end of the path.
align	is one of these alignment codes: **0** align on start of path **1** align on end of path **2** center on path **3** justify text on the path **+20** reverse path direction **+40** allow text to continue at start of path when it goes beyond the end of the path

Examples

```
(text to print on a path) 0 0 0 SHPATH
```

Modes

This command is applicable in all modes.

Related commands

SHPIT

Use **SHPIT** to print distorted text into a quadrilateral shape. The distorted text will fill or take the shape of the quadrilateral frame. The text can or cannot have a drop shadow effect applied. The quadrilateral itself can also be drawn by coding an additional **DRAWPOL**. This command can be used for image personalization by printing text on top of an image like text on a wall or in the sand. For a better seamless effect you can replace the text and shadow colors with image patterns using a small pattern extracted from the image, see **SETPAT** for how to create an image pattern. The small image can be created using any image tool on the market.

Syntax

Without drop shadow:

```
(variable string) VSUB [ llx lly ulx uly urx ury lrx lry ] [text_color] XYZ
SHPIT
```

With drop shadow:

```
(variable string) VSUB [ llx lly ulx uly urx ury lrx lry ]
[text_color shadow_color Xoff Yoff] XYZ SHPIT
```

Where:

llx and lly	are the lower left point of the quadrilateral
ulx and uly	are the upper left point of the quadrilateral
urx and ury	are the upper right point of the quadrilateral
lrx and lry	are the lower right point of the quadrilateral
X	is one of these:
	0 no mirroring or rotation
	1 rotate 90 degrees
	2 rotate 180 degrees
	3 rotate 270 degrees
	4 mirror
	5 mirror and rotate 90 degrees
	6 mirror and rotate 180 degrees
	7 mirror and rotate 270 degrees
Y	is one of these:
	0 no fitting
	1 fit-in-width
	2 stretch-in-width
Z	is one of these:
	0 align left
	1 align right

	2 align center
	XYZ Examples:
	200 = rotate 180 degrees, no fitting and align left
	401 = mirror, no fitting and align right
	622 = mirror, rotate 180 degrees, stretch-in-width and align center
text color	can be a plain color, pattern, or **GEPKey**
shadow color	can be a plain color, pattern, or **GEPKey**
Xoff	drop shadow horizontal offset in current units (can be negative).
Yoff	drop shadow vertical offset in current units (can be negative).

Examples

```
([=FirstName=] [=LastName=] ) VSUB [ 78.0 346.8 78.0 182.966 384.0
43.2 384.0 103.2 ] [COLOR004 COLOR003 2.0 2.0] 22 SHPIT
```

Modes

This command is applicable in all modes.

Related commands

SETPAT,DRAWPOL

SHR and SHr

SHR and **SHr** print data right aligned on the main (**SHR**) or secondary (**SHr**) print position.

SHR resets the main horizontal print position to the last value specified by **MOVETO** and forwards the vertical print position by the **SETLSP** value.

SHr sets the secondary print position at the point reached after printing the data.

Syntax

```
(printable data) SHR
(printable data) SHr
```

Modes

These commands are applicable all modes.

Related commands

- MOVETO
- MOVEH
- MOVEHR
- SETLSP
- SHL and SH
- SHC and SHc
- SHJ and SHj
- SHMF, SHMf, and SHmf

SHROW

Use **SHROW** to print a row in a table.

Syntax

```
[ [ /param1 value1 ... /paramX ValueX ]     % cell 1
  ....
  [ /param1 value1 ... /paramX ValueX ]     % cell N
] SHROW

[ [ /param1 value1 ... /paramX ValueX ]     % (without /Width): default params applicable to all cells
  ....
  [ /param1 value1 ... /paramX ValueX ]     % cell 1
   ....
  [ /param1 value1 ... /paramX ValueX ]     % cell N
] SHROW
```

Where:

/param can be one of the following:

/Width width of the cell in current units

/Height minimum height of the cell in current units (optional)

/MaxHeight maximum height of the cell in current units (optional)

/Margins cell margins in current units

/CellFill color to fill the cell. `null` is the default value and means that the cell is not filled with any color.

/CellStroke GEPkeys to stroke the borders of the cell

/TextAtt VIPP® code to set text attributes

/Align align attribute (same as SHP)

/CellText text to be placed in the cell

/CellImage image to be placed in the cell

/FixHeight cell height is fixed and text is scaled to fit in the cell, recommended to improve performance

/IAlign align attribute for CellImage, same as **ICALL**. If not present the image is aligned according to /Align but restricted to the **ICALL** align values.

Note: **SETLKF**, **FOREACH** and **SHROW** commands can interact with each other when **SHROW** is called in the **FOREACH** loop. This behavior is enabled by the following **FOREACH** syntax: `{ sequence of VIPP commands } variable_array /MF FOREACH`. When the /MF option is used, the total size of all **SHROW** commands that are called in the **FOREACH** procedure is evaluated. If the total size of the **SHROW** commands do not fit in the frame, the rows are not imaged, a **NEWFRAME** is called, and the procedure is executed a second time with the same table entry values. This option is intended to address multiple frames with different layouts, so that possibly on the second , a different **SHROW** is called.

Example

```
[   % default cell parameters
        [ /FixHeight 18
          /Margins [ 4 4 4 4 ]
          /CellStroke [null LT null null]
          /CellFill null
          /TextAtt { F004 L003 C003 }
        ]
        % cells
        [ /Width 52   /Align 000020 /CellText Date ]
        [ /Width 53   /Align 000020 /CellText Time ]
        [ /Width 146  /Align 000020 /CellText Place ]
        [ /Width 126  /Align 000020 /CellText Number ]
        [ /Width 48   /Align 000020 /CellText Rate ]
        [ /Width 55   /Align 000021 /CellText Minutes ]
        [ /Width 60   /Align 000021 /CellText Amount ]
      ] SHROW
```

Modes

This command is applicable in all modes.

Related commands

- ENDRPE
- FROMLINE
- INDEXRPE
- RPEKEY
- STARTLM
- BEGINTABLE
- ENDTABLE

SHT and SHt

SHT and **SHt** print possibly truncated data on the main (**SHT**) or secondary (**SHt**) print position. The amount of data printed is limited to the length defined by the column width. These commands behave like **SHP** and **SHp** but print only the first line of the paragraph. When the data is truncated an ellipsis, . . ., is printed at the end of the line to indicate the truncation.

Syntax

```
(printable data) align SHT
(printable data) colwidth align SHT
[ (printable data1) (printable data2) .... ] align SHT
[ (printable data1) (printable data2) .... ] colwidth align SHT
[ count (printable data)) ] align SHT
```

Where:

align is the alignment option, and can be specified as one of these values:

0 left

1 right

2 center

3 XYcenter (uppercase)

+000 wrap on Latin words and between any Asian characters

+100 wrap-down according to Asian rules

+200 wrap-up according to Asian rules

+300 wrap-up + hanging punctuation according to Asian rules

+400 hanging punctuation according to Asian rules

Note: Asian language and formatting rules for the related encoding are previously set by the **SETCJKENCMAP** and **SETCJKRULES** commands.

colwidth is the column width used for word wrapping. When colwidth is not specified, the value defined by a previous **SETCOLWIDTH** is used. When colwidth is specified, it overrides and replaces the value defined by a previous **SETCOLWIDTH**.

With the last two syntax examples, **SHT** appends the specified list of strings and formats the resulting string, the sum of all lengths cannot exceed 65,535 characters.

When using alignment 3, any word that causes the line to exceed the column width, but is within a specific percentage, prints with that line. The threshold is by default set to 0.05 % of the column width. This value is stored in a variable named *VAR.SHT_OVER*. Use the **SETVAR** command to change this value.

Modes

These commands are applicable in all modes.

Related commands

- MOVETO
- MOVEH

- MOVEHR
- SCALL
- SETCOLWIDTH
- SETLSP
- SETVAR
- SHMF, SHMf, and SHmf
- VSUB

SHX

The **SHX** command prints data with a **GEPkey** and rotates data.

SHX resets the main horizontal print position to the last value specified by **MOVETO** and forwards the vertical print position by the **SETLSP** value.

Syntax

```
(printable data) rotate GEPkey align SHX
```

Where:

rotate	is the angle in degrees by which the text will be rotated. Positive is counterclockwise.
GEPkey	refers to a **GEPkey** that will be applied to each character glyph (outline and/or fill).
align	defines the number of records to read for processing the DBM. The default is 1. The record grouping option causes the records of the database file to be processed in packets, the first of which contains the field names. This is useful for long records containing numerous fields. **0** left **1** right **2** center **4** XYcenter (uppercase) **5** XYcenter (lowercase)
XYcenter	indicates that centering is performed not only horizontally, according to string length, but also vertically, according to character height.

Note: **SHX** can be associated with a background, refer to **SETTXB** and **INDEXBAT** only when a non-outlined **GEPkey** is used. A non-outlined **GEPkey** has a LineWidth equal to 0, refer to **SETGEP**.

TIP

This command can be used to place outlined text across the page using a large font.

Modes

This command is applicable in all modes.

Related commands

- INDEXBAT
- SETGEP
- MOVETO
- SETBAT
- SETLSP
- SETTXB

SKIPPAGE

SKIPPAGE causes the current page to be skipped, not printed. This command is used in a **BEGINPAGE** procedure under the control of an **IF/ENDIF** statement.

Syntax

```
SKIPPAGE
```

Examples

This example shows how to cause the banner pages, the pages with lines of asterisks to be skipped.

```
/BANNER 1 10 /eq 132(*) SETPCD
{ IF BANNER {SKIPPAGE} ENDIF } BEGINPAGE
```

Modes

This command is applicable in all modes.

Related commands

BEGINPAGE, IF/ELSE/ELIF/ENDIF,SETPCD

SLIPSHEET

SLIPSHEET inserts an additional page, intended to separate sets, in the middle of a job. Although available for all modes, it is mainly intended for line mode because in this mode it is the only way to generate extra pages in addition to the pages of data. In duplex or Multi-Up mode, a **NEWFRONT** is coded before and after the **SLIPSHEET** if the page is required on a separate sheet unless a media change is coded in the slipsheet procedure.

Syntax

```
{ slipsheet proc } SLIPSHEET
```

Where:

{ slipsheet proc }	can contain the commands necessary to specify the format of the slipsheet, for example, **SETMEDIA** or **SETFORM**.

The procedure enclosed in braces does not affect the remainder of the print job, for example, page numbering. The slipsheet prints using the current form by default. To cancel it or to select another one use **SETFORM** in the procedure.

Examples

This is an example of coding that selects a blue media and a specific form.

```
/SLIP 1 5 0 10 /eq (criteria string) SETPCD
{ IF SLIP
  {  { (:blue) SETMEDIA
       (slipform.frm) SETFORM
     } SLIPSHEET
  }ENDIF
} BEGINPAGE
```

Note: In native mode or database mode, always code **SLIPSHEET** immediately after a **PAGEBRK** or before the beginning of the first page.

TIP

When used in line mode, this command is called in either **BEGINPAGE** or in a /P **ENDPAGE** procedure under a PCD test. Use **GETFIELD** to capture data from the current page or from previous pages and print then on the slip sheet using an in-line form.

Modes

This command is applicable in all modes.

Related commands

- BEGINPAGE
- ENDPAGE
- GETFIELD
- SETFORM
- SETMEDIA

SOF_off

SOF_off disables end of line mode when %! is encountered.

Syntax

```
SOF_off
```

Modes

This command is applicable in line mode and database mode.

Related commands

%%EOF, %!

SORT

SORT sorts a table in either ascending or descending order based on a field key.

Syntax

`VARtable /VARkeyX /opt SORT`

Where:

VAR_table	is of the form: `[[/VARkey1 /VARkey2 ..] [val11 val12 ..] .. [valN1 valN2 ..]]`
/VARkeyX	is one of the keys in `[/VARkey1 /VARkey2 ...]` is one of: **/A** ascending **/D** descending

Modes

This command is applicable in all modes.

Related commands

- SETVAR
- ADD
- FOREACH
- UPDATE

SPOOLNAME

SPOOLNAME specifies a path and file name for a temporary file used to store the data in cyclecopy mode, refer to **SETCYCLECOPY**. Use this command in either the `/usr/xgf/src/xgfunix.run` or `x:\xgf\src\xgfdos.run` file, where the default is specified.

Syntax

```
SOF_off
```

Note: This file is not used on DocuPrint NPS and FreeFlow Print Server products, except in streaming mode.

Modes

This command is applicable in all modes.

Related commands

SETCYCLECOPY,SETMAXCOPY

STARTBOOKLET

Use the **STARTBOOKLET** command to start a booklet. This command is coded before any marking command on the first page of the booklet. When present, **STARTBOOKLET** executes the **VIPPStartBooklet** procedure, the implementation of which is device dependant and intended to execute the appropriate action on the device to initialize a booklet. If booklet is not supported on the device the procedure is empty, as that is the default. For more information, refer to Booklet Support.

Syntax

```
STARTBOOKLET
```

Modes

This command is applicable in all modes.

Related commands

ENDBOOKLET,SETPARAMS

STARTDBM

STARTDBM starts database mode. This command can precede the data in the database file. A DBM file name is provided as an operand. It is stored in one of the libraries defined by **SETFPATH** or in the Project directories used by the job. Use of the `.dbm` extension is recommended. The DBM contains all page layout and processing instructions required to print a document for each record in the database file.

When **SETCYCLECOPY** is used inside a Data Base Master, a **CHKPOINT** is implicitly executed. When combined with a conditional statement, **IF/ELSE/ENDIF** on specific records this allows a set of records to be processed as a document, set with a specific number of copies.

Commands in a DBM are any native mode command. In addition, **VSUB** is used to substitute field contents.

For further information on database mode refer to VIPP® data streams in the *FreeFlow VI Compose User Guide*.

- To set up a Multi-Up duplex job: Before a **STARTDBM** command, or in the associated JDT,

  place a **ZSORT** statement with `stacksize=1`. For more information, refer to ZSORT.

- Correct processing of data when the database file contains a multiple-byte data stream:

  Before a **STARTDBM** command, place a **SETFONT** command that calls the appropriate multiple-byte font.

Syntax

```
(Data Base Master name) STARTDBM<EOL>
(Data Base Master name) record-grouping STARTDBM<EOL>
```

Where:

(Data Base Master name)	is the DBM name to use. When no DBM name is supplied to **STARTDBM**, for example, () **STARTDBM**, a DBM name is supplied with each record in a field called DBM_NAME. This allows the usage of a different DBM for each record.
<EOL>	is the end-of-line marker, for example, LF, CRLF, or CR. It can immediately follow the keyword **STARTDBM**. Any additional space between **STARTDBM** and <EOL> causes the job to fail
record grouping	defines the number of records to read for processing the DBM. The default is 1. The record grouping option causes the records of the database file to be processed in packets, the first of which contains the field names. This is useful for long records containing numerous fields.

Because the DBM is executed for each record, for efficiency and to avoid side effects, you can place general setting commands such as **DUPLEX_on**, **SETMULTIUP**, **SETDBSEP** and others, before the **STARTDBM** command. These commands can be grouped in a JDT and called with **SETJDT** before **STARTDBM**. Last, set up an initialization section at the beginning of the DBM as shown in this example:

```
/VAR_INI true /INI SETVAR
% this section executes only on the first call to the DBM
IF VAR_INI
{ TWOUP
   DUPLEX_on
   2480 3500 SETPAGESIZE
   /VAR_INI false SETVAR
} ENDIF
```

Modes

This command is applicable in database mode.

Related commands

- SETFPATH
- SETJDT
- STARTLM
- VSUB

STARTLM

STARTLM starts line mode, for further information on line mode refer to VIPP® data streams in the *FreeFlow VI Compose User Guide*.

This command precede the line mode data. A JDT file name is provided as an operand. It is stored in one of the libraries defined by **SETJPATH** or in the Project directories used by the job. Use of the .jdt extension is recommended. This JDT file contains all page layout and processing instructions for the job.

In a JDT, use only those commands related to the page layout setting such as orientation, **SETGRID**, **SETMARGIN**, **SETFONT**, **SETFORM**, and RPE definitions. Commands that print items on the page such as **SHX**, **ICALL**, and **DRAWB**, cannot be used except in an in-line form definition or in an **ENDPAGE** procedure.

When a specific command is not specified in the JDT, the job uses the defaults set in the `/usr/xgf/src/xgf.def` file.

Syntax

```
(jdtname) STARTLM<EOL>
```

```
(jdtname2) (jdtname1) count STARTLM<EOL>
```

Where:

jdtname	is the JDT name to use. When no JDT reference is specified, for example,() **STARTLM**, no JDT is searched for and commands are embedded between the %! and **STARTLM** statements.
<EOL>	is the end-of-line marker; for example, LF, CRLF, or CR. It immediately follow the keyword **STARTLM**. Any additional space between **STARTLM** and <EOL> causes an extra blank line at the top of the first page.
jdtname1	will be applied for count pages before applying jdtname2. This is useful for printing banner pages with a specific JDT.

Modes

This command is applicable in line mode.

Related commands

- LMSKIP
- SETJDT
- SETJPATH
- STARTDBM

STARTOFSET

STARTOFSET acts as a set delimiter for FreeFlow Print Server print jobs only. Use this command with FreeFlow Print Server. You can place this command at the beginning of the first page of a set.

For further information on setting up and using the subset finishing features of the FreeFlow Print Server products, refer to the appropriate FreeFlow Print Server publication.

Note: This command can be used on FFPS without **SETFINISHING**. Finishing can be set at the FFPS using a queue with Subset Output Option set to Retrieved from PDL.

Syntax

```
STARTOFSET
```

Modes

This command is applicable in all modes.

Related commands

- ENDOFSET
- SETFINISHING
- SETOBIN

STARTXML

STARTXML initiates XML mode. This command precede the XML data or be used after a **SETLMFILE** command referencing an XML file. It takes as an operand a reference to an XML Job Ticket (XJT) file. The XJT file is stored in one of the libraries defined by **SETJPATH** or the Project directories used by this job. Use of the `.xjt` extension is recommended.

When the XJT operand is an empty string VI Compose creates the XJT name by appending .xjt to the root tag name of the XML file. If this XJT does not exist VI Compose uses an XJT called `xmldump.xjt` located in `xgf/jdtlib`.

An XJT file contains instructions on how to process and arrange the XML data into a document. It is similar to a line mode JDT and the same rules and limitations apply to its contents, refer toSTARTLM. It contains all global layout definitions for the document, orientation, forms, medias, frames, fonts, colors, plus an XML Processing Definition (XPD) table that describes specific actions to be performed on specific XML tags.

STARTXML parses and consumes the XML file populating the `VXVpath` directory with node contents and performing actions or substitutions as defined in the XPD table.

When **SETCYCLECOPY** is used in a BTA start process for a given tag a **CHKPOINT** is implicitly executed when the end tag is encountered. This allows the entire XML sub-tree to be processed as a document (set) with a specific number of copies.

STARTXML ends after the end root tag. At that point VI Compose returns to native mode.

XML files encoded using UTF-16 are supported providing the XML data begins with a UTF-16 BOM (0xFFFE or 0xFEFF). Note that UTF-16 XML data is converted to UTF-8 on the fly, so the node contents are printed with UTF-8 encoded fonts.

Syntax

```
(xjtname) STARTXML
```

```
() STARTXML
```

Modes

This command is applicable in XML mode.

Related commands

- BEGINXPD
- ENDXPD
- BTA
- ETA
- BTS
- ETS
- SETJPATH

STOREVAR

Use **STOREVAR** to store data in a file for later use by the same or another application. Everlasting counters or data shared by different applications are examples of data to store using **STOREVAR**.

For **STOREVAR** to be able to write to a file three conditions are met:

- The file and directory have write access at the file system level
- FFPS, if the target can allow a PS program to write to the directory where the file is written
- In project mode the **STOREVAR** file exist before the job is submitted

Use **STOREVAR** to define in which file to place data to be stored. The actual storage file is defined prior to using **STOREVAR** in project mode. This is because in project mode a resource needs to be assigned one of three levels of scope, local, folder, global, and is important because when several projects are using **STOREVAR** with the same file name the intent can be to share that file folder or global scope, so all projects write to the same file, or to keep one instance of the file for each project, local scope. By creating the file in the appropriate directory, for example, either in `/usr/xgfc/FolderX/ProjectY`, `/usr/xgfc/FolderX/fshared` or `/usr/xgfc/gshared` before submitting any job that writes to it, the user implicitly indicates the scope of that file.

When a VPC is created the **STOREVAR** file, even empty have part of the VPC so it is created when the project is deployed. Printing a VPC using **STOREVAR** in print and forget mode defeats the purpose of **STOREVAR**, sharing information between jobs because the `vpcpfd` project directory is destroyed after the job is completed.

Important: This command stores data in the file system of the printer controller, in an area that is usually accessible by other applications or users. Use the command only as intended. For internal variables in a given VIPP® job, use the **SETVAR** command. Do not use the **STOREVAR** command.

Note: **STOREVAR** has been enhanced on VI Explorer and VI Design Pro:

On VIE the effect of **STOREVAR** is disabled.

On VDP, **STOREVAR** is disabled when browsing pages, Next and Previous page buttons. When the Refresh button is pressed, the effect of **STOREVAR** is visible but not persistent. The update is performed on a temporary copy of the file and is discarded at exit.

Syntax

```
data (store_file) STOREVAR
data (store_file) /option STOREVAR
```

Where:

data can either a string enclosed in parenthesis, an integer, a real or a variable of one of these types.

store_file is the name of the file into which data is stored. It is located in one of libraries referenced by **SETMPATH**. When the file exists, its contents is overwritten by data. When it does not exists, it is created in the first library referenced by **SETMPATH**. The library and the file have the appropriate write permissions.

option can be any combination of:

M allow **STOREVAR** in VIeC multi-instance mode. When using this option users ensure that the name /store_file is unique for each **STOREVAR** execution and that two jobs running in parallel under different VIeC instances can never execute **STOREVAR** with the same file name.

P in project mode, allow store_file to be created in the first **SETPPATH** directory when it does not exist. When using this option users ensure that the target directory is coherent with their workflow.

A store_file is opened in append mode. When this option is used the contents of store_file is not overwritten. Instead data is appended to it.

T when data is a string it is written to the file without parenthesis, text mode versus PostScript mode.

Example:

```
(Hello\nMr Smith\n) (textXYZ.txt) /AT STOREVAR
```

appends the following two lines to the contents of `textXYZ.txt`:

```
Hello
Mr. Smith
```

Examples

This example shows how to store, retrieve, and increment a variable.

```
VAR.NUMBER (number.var) STOREVAR      (this stores the variable)
/VAR.NUMBER (number.var) RUN SETVAR   (this retrieves the variable)
/VAR.NUMBER ++                        (this increments the variable)
```

Modes

This command is applicable in all modes.

Related commands

SETMPATH, SETVAR

SUB

SUB subtracts a value from a numeric variable defined by **SETVAR** or an XML variable.

Syntax

```
/VARname number SUB
/^XMLname number SUB
```

Where:

/VARname	refers to a numeric variable previously initialized by **SETVAR**.
/^XMLname	refers to an XML variable that does not need not be explicitly initialized. VI Compose initializes all XML variables to an empty string, which is equivalent to a numeric string equal to zero.
number	is the number to be subtracted from the variable. It can be an integer, a real, or a numeric string. When large numbers are involved a numeric string is mandatory.

Numeric strings accommodate large numbers up to 40 digits, 25 digits for the integer part and 15 digits for the decimal part. In a numeric string the negative sign and the decimal delimiter are defined by the parameters /DecimalPoint and /NSign and can occur anywhere in the string.

It is mandatory to set these parameters with appropriate values to ensure accurate results. Defaults are defined in the file `/usr/xgf/src/xgf.def`. Characters in the numeric string other than these two plus the digits 0–9 are ignored.

The initial length of the string defined by **SETVAR** is automatically extended up to 40 digits when needed.

Reals and integers are used only for small values <= 99999, for instance the implementation of a counter. The decimal delimiter, if present, is always the point (.). The negative sign, if present, is always the minus (-) and are the first character.

Built in variables are available in forms or **ENDPAGE** procedures when **RUNTIF** is used:

TIFPAGE	page number of the current page
TIFLAST	page number of the last page

Examples

```
/VAR.CNT1 0 SETVAR
/VAR.CNT1 12 SUB
/VAR.CNT1 –3 SUB
```

```
/VAR.SUM (0) SETVAR
/VAR.SUM (1,234,890,566,00–) SUB
```

Modes

This command is applicable in all modes.

Related commands

- ADD
- SETVAR

TIFORI_off

TIFORI_off disables processing of tag 274 Orientation within **ICALL**. **TIFORI_off** is the default.

Syntax

```
TIFORI_off
```

Modes

This command is applicable in all modes.

Related commands

TIFORI_on,ICALL

TIFORI_on

TIFORI_on enables processing of tag 274 Orientation within **ICALL**. The **ICALL** default does not process the orientation tag, **TIFORI_off** and assumes orientation is 1, the default for TIFF files. Orientation processing is provided as an option to ensure compatibility with previous releases.

Syntax

```
TIFORI_on
```

Modes

This command is applicable in all modes.

Related commands

TIFORI_off,ICALL

TUMBLEDUPLEX_off

TUMBLEDUPLEX_off disables duplex printing. **TUMBLEDUPLEX_off** is the default. This command is equivalent to **DUPLEX_off** and exists only for compatibility with earlier versions.

Syntax

```
TUMBLEDUPLEX_off
```

Modes

This command is applicable in all modes.

Related commands

- BCALL
- DUPLEX_off
- DUPLEX_on
- ENDIMP
- TUMBLEDUPLEX_on

TUMBLEDUPLEX_on

TUMBLEDUPLEX_on enables tumble duplex printing, 180˚ rotation on the back side. The default is **TUMBLEDUPLEX_off**.

Syntax

```
TUMBLEDUPLEX_on
```

Modes

This command is applicable in all modes.

Related commands

- BCALL
- DUPLEX_off
- DUPLEX_on
- ENDIMP
- TUMBLEDUPLEX_off

TWOUP

TWOUP enables two-up mode. Two reduced logical pages are printed on top of each other, **LAND** and **ILAND** or side by side PORT and **IPORT** on the same physical page.

This command must be used as the first command following the %! line in the print file or as the first line in a JDT.

Syntax

```
TWOUP
```

Modes

This command is applicable in all modes.

Related commands

- BCALL
- ENDIMP
- ONEUP
- SETMULTIUP

UPDATE

UPDATE is used exclusively in a **FOREACH** loop to update the current entry of the table after updating one or more values in this entry.

Syntax

```
UPDATE
```

Examples

This example adds 1 to each *VAR1* value in the table:

```
{ /VAR1 ++ UPDATE } VARtable FOREACH
```

Modes

This command is applicable in all modes.

Related commands

- FOREACH
- SETVAR
- ADD
- SORT

USPS4CB

USPS4CB creates and images a USPS 4-state Customer Barcode, Intelligent Mail® Barcode from the Tracking Data and Routing Code strings specified. No special font is required.

The dimensions of the bars meet the specifications in Revisions D and E of the *USPS Intelligent Mail® Barcode, 4-State Customer Barcode, Specification USPS-B-3200* dated 07/24/07 and 10/30/07.

Syntax

```
(tracking data) (routing code) USPS4CB
(tracking data) (routing code) align USPS4CB
```

Where:

tracking data	is the 20 digit tracking data string comprised of the following sub-fields: 2 digit Barcode Identifier. The second digit must be 0 through 4. 3 digit Service Type Identifier. 6 digit Mailer Identifier. 9 digit Serial Number. This string may have spaces to match the human readable form that may be required by some uses of the 4-state barcode and to improve readability.
routing code	is the delivery point ZIP code. It can be in any of the 4 forms accepted by the USPS: **( )** An empty or null string for no ZIP code. **(12345)** A 5 digit ZIP code **(12345-6789)** A 9 digit ZIP+4 code **(12345-6789 01)** An eleven digit ZIP+4 + 2 digit DPC This string may have spaces to match the human readable form that may be required by some uses of the 4-state barcode or dashes (-) to improve readability.
align	indicates which point of the barcode will be aligned on the secondary print position using these values: **0** top left **1** top right **2** top center **10** bottom left (default) **11** bottom right **12** bottom center **20** center left **21** center right **22** center center

The tracking data and routing data strings may have spaces, dashes, or other printable characters embedded in the strings. These characters will be removed before the barcode is created. After they are removed, the tracking data string must have exactly 20 digits, with the second digit being in the range 0–4, and the routing code string either be empty or have 5, 9 or 11 digits. The bar code will be printed at the current secondary position, aligned as specified. It is the users responsibility to insure the required clear zones around the bar code are maintained.

Examples

```
(01 234 567094 987654321)   ( )   USPS4CB
(01234567094 987654321)   (01234)   0   USPS4CB
(01 234 567094987654321)   (01234-5678)   20   USPS4CB
(01 234 567094 987654321 )   (01234 5678 91)   10   USPS4CB
```

The tracking data and routing data strings in the last example are in the format required for printing the data in human readable form as required by some uses of the USPS Intelligent Mail® Barcode 4-state barcode.

Modes

This command is applicable in all modes.

Related commands

MOVETO , MOVEH,MOVEHR

VPDISTPAT

When using the new Correlation vector pattern feature the text may appear visible to the naked eye on some print engines depending on the colors selected for the effect.

In this situation it is possible to use VPDISTPAT to apply a distraction pattern on top of the effect.

Syntax

```
option VPDISTPAT
```

Where

The option can be one of the following integers:

- **0** no distraction pattern, same as not using VPDISTPAT
- **1** light distraction pattern 1 or 2 layers
- **2** heavy distraction pattern 1 or 2 layers
- **11** light distraction pattern using col2 only 2 layers
- **21** light distraction pattern using col3 only 2 layers
- **12** heavy distraction pattern using col2 only 2 layers
- **22** heavy distraction pattern using col3 only 2 layers

Examples

```
500 3000 1750 300 VP02 DRAWB
/NHEB 50 SETFONT
575 2770 MOVETO (Correlation Text) SHL
1 VPDISTPAT
```

Modes

All modes.

Related commands

SETPAT

XGF

XGF forces the PostScript interpreter to load VI Compose. VI Compose is implicitly loaded by these commands:

DUPLEX_on	RUN	SETLMFILE	STARTDBM
FSHOW	SETDBSEP	SETMARGIN	STARTLM
ILAND	SETDLFILE	SETMEDIA	STARTXML
INDEXFONT	SETEPATH	SETMEDIAT	TUMBLEDUPLEX_on
IPORT	SETFONT	SETMPATH	TWOUP
LAND	SETFORM	SETMULTIUP	XGF
MAKEVMFILE	SETEPATH	SETPARAMS	XGFDEBUG
MAKEVMFORM	SETGRID	SETPROJECT	XGFRESDEF
ORITL	SETIPATH	SETTAB	
PAGERANGE	SETJDTSETJPATH	SETUTAB	
PORT	SETJPATH	SHIFT	

In order for the PostScript interpreter to load VI Compose, the VIPP® file submitted for printing must start with one of the commands listed above.

When you need to start the VIPP® file with any other command, it must be preceded by XGF or registered as an initial command by **XGFENTRY** in the `/usr/xgf/src/xgf` file and in the `/usr/xgf/src/ xgfunix.run` file after the end line.

Syntax

```
XGF
```

Modes

This command is applicable in all modes.

Related commands

XGFENTRY

XGFDEBUG

XGFDEBUG forces the printing of Native Mode Prefix (NMP) and Form Feeds for documentation and debugging purposes. **NMP** commands are processed as print data and are not interpreted as VIPP® commands. Form Feeds are printed as <FF> and have no skip effect.

This command is coded in a JDT or before the **STARTLM** command.

Syntax

```
XGFDEBUG
```

Modes

This command is applicable in line mode.

Related commands

None

XGFEND

XGFEND is mandatory at the end of a native mode job when normalization or demographic service is active, or Generic **ZSORT** is used. If not, this command has no action.

XGFENTRY

XGFENTRY registers a command as an initial command. For further information, refer to XGF.

Syntax

```
/command XGFENTRY
```

Examples

This example registers **MOVETO** as an initial command.

```
/MOVETO XGFENTRY
```

Note: Use the command only in the `/usr/xgf/src/xgf` file, and the `/usr/xgf/src/xgfunix.run` file after the end line.

Modes

This command is applicable in all modes.

Related commands

XGF

XGFRESDEF

XGFRESDEF registers the specified resource in the memory of the PostScript interpreter. Use this command to embed resources in the data stream to prepare VIPP® jobs for printing on a decentralized printer. This is referred to as a self-contained VIPP® data stream. The syntax of the command varies depending on the resource type. For further information, refer to VIPP® resources in the *FreeFlow VI Compose User Guide*.

Syntax

```
/formname form_data FBIND XGFRESDEF
/resourcename { resource_data } XGFRESDEF
/imagename [ <image_data1> <image_data2> … ] XGFRESDEF
/psname MAKEVMFILE ps_data %%EOD_XGF XGFRESDEF
/ilname {VIPP_code } /CACHE XGFRESDEF
/ilname {VIPP_code } [ llx lly urx ury ] /CACHE XGFRESDEF
```

Where:

formname	is the name of a VIPP® form.
form_data	is the form contents including encapsulating braces (therefore, not shown in the syntax).
resourcename	is the name of a segment, a DBM, a JDT, a font list, or an encoding table.
resource_data	is the exact contents of the resource.
imagename	is the name of a TIFF or JPEG file.
image_data	is the exact contents of the TIFF or JPEG file converted to hexadecimal fixed length records encapsulated between < and >. The last record may be shorter. The record length must be such that the number of hexadecimal records does not exceed 65535. In general, 128 is sufficient.
psname	is the name of a PostScript file as created by a PostScript driver out of a document processing application.
ps_data	is the PostScript code created by the driver.
ilname	is the name of an in-line resource that needs to be cached. Normally, an in-line resource has no name so it cannot be cached. Use this syntax to give it a name and cache it for subsequent use with **SCALL**, **SETFORM** or **SETBFORM**.
VIPP_code	is the contents of the in-line resource.
llx lly urx ury	is the bounding box of the in-line resource in current VIPP® units, for more details, refer to % %BoundingBox. If not specified, the bounding box defaults to the physical page size.

Examples

This example illustrates various uses of **XGFRESDEF**.

```
/bill.frm { PORT 100 1150 MOVETO (Xerox) SHL ... } FBIND XGFRESDEF
/a1b.jdt { ILAND 120 66 SETGRID (bill.frm) SETFORM ... } XGFRESDEF
/rxlogo.tif[ <49492a00...> <007fff80...> ... ] XGFRESDEF
/doc1.ps MAKEVMFILE %!PS ... showpage %%EOF %%EOD_XGF XGFRESDEF
```

```
/Seg1 {... VIPP code ...} [ -100 -100 400 400 ] /CACHE XGFRESDEF
```

Note: Embedded resources appears in the data stream before they are referenced by any VIPP command.

Tip: **SETLMFILE** and **SETDLFILE** can be embedded in the data stream using **XGFRESDEF**.

Modes

This command is applicable in all modes.

Related commands

- CACHE
- FCALL
- ICALL
- RUN
- SCALL
- SETBFORM
- SETDLFILE
- SETENCODING
- SETFORM
- SETJDT
- STARTDBM
- STARTLM

XMLATL

XMLATL provides a way to process the attributes of an XML tag. It must be placed inside a **BTA** and **ETA** or **BTS** and **ETS** command pair. It takes a process as an argument and calls it for each attribute after setting **XMLATN** and **XMLATV** to the appropriate values for that attribute. The procedure is expected to do something, print, store, set variables, and so on with each **XMLATM** or **XMLATV** pair.

Syntax

```
{ VIPP commands and XMLATN and XMLATV variables } XMLATL
```

Where

XMLATN	is the attribute name.
XMLATV	is the attribute value.

Modes

This command is applicable in XML mode.

Related commands

- BTA
- ETA
- BTS
- ETS

XMLSYN

XMLSYN defines synonyms for XML tags. These synonyms must then be used instead of the original names in the XPD definitions and XML **VXVpaths**.

This command is intended to provide a means to address long tag names in complex XML structures that may cause **VXVpath** to exceed 128 characters, which is the maximum length allowed for **VXVpaths**. You can now assign a synonym for the XML tag name and reference that assigned name in VIPP®.

Syntax

```
[ /tag_name1 /tag_syn1 /tag_name2 /tag_syn2 … ] XMLSYN
```

Where

tag_nameX	is the name of a tag in the current XML data tree.
tag_synX	is the synonym to be substituted for that tag.

Examples

```
[ /ACCOUNT_INVOICE /AINV
/ACCOUNT_INFORMATION /AINF
/ACCOUNT_UNITS /AU
/ACCOUNT_USAGE_CHARGES /AUC ] XMLSYN
```

Modes

This command is applicable in XML mode only.

Related commands

- BEGINXPD
- ENDXPD
- BTA
- ETA
- BTS
- ETS

ZSORT

ZSORT provides the capability to print simplex or duplex multi-up documents in a north and south imposition order or in stacks, which can be taken to a cutter and cut and stacked, maintaining record order through the stack. This is often used to maintain zip code or postal code order. In addition, a slipsheet can be created and inserted between the stacks of output to aid in identifying the stack boundaries. **ZSORT** also provides the capability to maintain the front and back alignment for each record when variable data needs to be placed on both the front and back sides of a multi-up document.

ZSORT must be executed as early as possible in the job. For instance it can be placed at the very beginning of the submission file before the **START** command preceding the data or in the **JDT** or **XJT** called before that **START** command.

Syntax

```
{ slipsheet proc } stacksize options ZSORT
{ slipsheet proc } stacksize [options grouping] ZSORT
```

Where

slipsheet proc	is the VIPP® code to be executed at the end of a stack to produce a slipsheet. This procedure is processed by the **SLIPSHEET** command and must be a valid procedure for that command. This procedure may be empty when no slipsheet is desired.
stacksize	is the maximum number of sheets in a stack. The size of a stack is generally based on the maximum number of sheets that can be cut by the post-processing cutter. This size may be adjusted depending upon the amount of printer storage capacity and average amount of data per stack. A reasonable stack size is between 500 and 1000 sheets.

options	is an integer of the form XXXY, where: XXX is the number of pages per document minus 1. **Y** can be one of: **0 database mode only**: shorten last stack to fit the number of records with continuous duplex. When the document has an odd number of pages and the last page falls on a front the back will contain the first page of the next document. **1 database mode only**: add blank logical pages to last stack to match the stack size with continuous duplex **2 database mode only**: shorten last stack to fit number of records with non-continuous duplex. When the document has an odd number of pages the back of the last page is left blank. **3 database mode only**: add blank logical pages to last stack to match stacksize with non-continuous duplex. **4 all modes**: shorten last stack to fit the number of records. **5 all modes**: add blank logical pages to last stack to match the stack size.
grouping	is an integer that must be a divider of the multi-up number. It enables multiple consecutive logical pages on the sheet to be treated in sequence instead of through the stack. The selection and order of these pages is determined by the multi-up filling order, /FillOrder for **SETLAYOUT** or sequence order for **SETMULTIUP**.

Database mode and generic ZSORT

Options Y=0–3 apply to database mode only while options 4–5 apply to all modes generic **ZSORT**.

Options 0–3 are optimized for database mode so it is recommended to use them when possible. Their behavior is slightly different from options 4–5 and have some specifics and limitations detailed below.

Specifics and limitations for options Y=0–3, database mode

- **DBM calls and multiple page documents**

  Each record will be processed XXX+1 times. The DBM will be called that number of times and is supposed to include conditional logic so that each call produces one and only one page at a time. For that purpose, the document page number is provided on each call by the built-in variable **ZSPAGE**. For backward compatibility using the BACK variable for a 2 page duplex document is still possible.

  Thus, although **ZSORT** can be used for multiple page documents, each record must generate the same number of pages. For example, if printing a four page newsletter, all records must generate four pages. If the number of pages per document is not fixed options 4–5 must be used.

- **Built-in variables**

  4 built-in variables are available that can be used in the DBM only:

  – ZSPAGE delivers the document page number incremented from 1.

  – ZSRECNUM delivers the record number in data file order. This is the current record number, incremented from 0.

  - ZSREPCNT delivers an ongoing repeat count, starting from 0.
  - ZSREPIDX delivers a record repeat index. **ZSREPIDX** increments for each instance but resets to 0 at the start of the next record.

- **Variable record repeat option**

  ZSORT can take into account a repeat value associated with each record. This is accomplished by declaring the repeat field in the JDT or before **STARTDBM** with the /ZSRepeatField parameter, as shown below. The repeat field must contain a numeric value.

  ```
  [ /ZSRepeatField /RecRepeat ] SETPARAMS
  ```

  For example, when printing business cards, you may have one record per customer, each with a variable repeat field called *CardVolume*. By setting the ZSRepeatField to *CardVolume*, **ZSORT** will process each record the number of times indicated by *CardVolume*. So one record may produce 250 business cards, where the next record will produce 100.

  ```
  [ /ZSRepeatField /CardVolume ] SETPARAMS
  ```

  Also it is possible to insert a repeat slipsheet after each repeat set by using the following syntax:

  ```
  [ { stack slipsheet } { repeat slipsheet } ] stacksize options ZSORT
  ```

- **Stack size limitation**

  ZSORT reads and buffers the number of records required to print a complete stack into virtual memory (VM). The amount of VM required for a stack can be computed as follows:

  ```
  N-upxstacksizexbuffersize/pagesperdocument
  ```

  Depending on the target device a very large VM allocation may cause the job to run with degraded performances and possibly abort with a VM exhausted error. Xerox recommends reducing the values above especially buffer size and stack size to reasonable and appropriate numbers.

  For example:

  The job set above will consume 60 Megabytes (8 x 2500 x 3000) of VM and is candidate for a VM exhausted error. If the records do not exceed 300 bytes and the cutter cannot accept more than 1000 sheets set the job as follows:

  ```
  8-up
  300SETBUFSIZE
  {}100010ZSORT
  ```

  This will reduce the VM allocation to 2.4 Megabytes and ensure rated speed and completion without error.

- **FFPS and PPR configuration**

  On FFPS, when a job using **ZSORT** is sent to a parallel rip workflow (PPR) each PPR chunk must produce a complete stack including slipsheet if any. It is recommended to normalize the VIPP® job create a.vpn fill before printing or as part of the job printing process. If the job has not been normalized first and the PPR splitting is performed directly on the database file the PPR chunk should be set to the number of records needed to generate a complete stack.

  For example, if the required stack size is 500 sheets, you must set the PPR chunk size to the number of records that will create the 500 sheets stacks. So if the job is a 4-up postcard application and one page per record, the chunk must hold 2000 records (500*4). If the record is repeated then you need less records to make the stack. With a repeat value of 10 you would only need 200 records (500*4/ 10).

Generic ZSORT (options Y=4–5)

Generic **ZSORT** can be applied to any VIPP® job. It requires no other special setting and involves the normalization service under the hood. A temporary normalized file is created and deleted for each stack. It can be used when database **ZSORT** cannot be applied.

NOTES and Recommendations

Adjust the stack size to a multiple of the number of pages per document, divided by 2 if duplex. If this is not the case VIPP® automatically performs this adjustment and the final stack size will be slightly different from the one coded in the **ZSORT** statement.

Logical pages can fit entirely on the physical page. When **SETMULTIUP** is used to define the multi-up layout, the syntax that specifies the size for each logical page must be used. When the correct syntax is not used, placements on the back will be wrong.

Examples

This is a **ZSORT** coding example for a 3 page document:

```
{}50020ZSORT
```

The DBM may be coded as:

```
IF ZSPAGE 1 eq
{ code to produce page 1 } ENDIF
IF ZSPAGE 2 eq
{ code to produce page 2 } ENDIF
IF ZSPAGE 3 eq
{ code to produce page 3 } ENDIF
PAGEBRK
```

This example shows the coding for printing postcards 4-up on US Letter paper in stacks of 100 sheets, with a switch to red media for a page containing the words Slip Sheet at the end of each stack POINT units are assumed. The stack is shortened if the number of input records is not a multiple of 400.

```
612 792 SETPAGESIZE
[ /PageWidth 306
  /PageHeight 396
  /Across 2
  /Down 2
] SETLAYOUT

{ ONEUP (:red) SETMEDIA
{/NHEB 24 SETFONT 300 1650 MOVETO (SlipSheet) SH } SETFORM
} 100 00 ZSORT
```

This is the **SETMULTIUP** statement required to define the same 4-up layout as the previous example:

```
[ [ 0 306] [396 396] 0 1 1 [306 306] [396 396] 0 1 1
[ 0 306] [ 0 396] 0 1 1
[306 306] [ 0 396] 0 1 1
] SETMULTIUP
```

Modes

This command is applicable in all modes.

Related commands

- SETLAYOUT
- SETMULTIUP
- DUPLEX_on
- DUPLEX_off
- SETBUFSIZE

3

Markers

This chapter contains:

Markers are reserved strings or sequences that, when embedded at appropriate places in a VIPP® job, can affect the behavior of certain commands.

%

Any occurrence of the character % outside of parentheses introduces a comment. The comment consists of all characters between the % and the next End of Line (EOL) delimiter such as LF, CR, or CRLF.

Syntax

```
% comment
```

Modes

This command is applicable in all modes.

Related commands

None

%%

The %% command indicates a Document Structuring Convention (DSC) statement. Some print servers take specific actions on a subset of DSC statements. Therefore, to conform to DSC specifications, all VIPP® files should begin with a standard DSC header as illustrated in this example.

```
%!

%%Title: xxxxxxx.ps
%%DocumentMedia: dmFAC 596 841 0 white
%%+ dmBVR 596 841 0 white BVR
%%EndComments
```

The %% DocumentMedia statement allows DocuPrint NPS to check for proper loading of specified media in the printer trays before starting to process the job. When the currently loaded media does not match that required by %% DocumentMedia, the job is put in a MediaWait state until the specified media is loaded.

This statement is mandatory when the SETMEDIA command is used in the job to change the media on a page-by-page basis.

Note: When the %% DocumentMedia statement is omitted, the job prints on the default media, therefore, media selection through SETMEDIA have unpredictable results.

Syntax

```
%%DSC statement
```

Refer to the appropriate Adobe documentation for further information on DSC statements.

Modes

This command is applicable in all modes.

Related commands

SETMEDIA

%!

The %! command indicates the start of a PostScript or VIPP® file. A line beginning with %! must always be the first line of a VIPP® job. It allows the job server on the imaging device to identify a PostScript data stream and invoke the PostScript interpreter.

Syntax

```
%! comment
```

Note: Failing to begin a file with %! may cause the job to print using the wrong interpreter, for example, printing the job as ASCII text in error.

Tip: %! is only mandatory at the beginning of the submitted file. It is not mandatory at the beginning of resource files.

%! also acts as a new line mode delimiter unless SOF_off is used, refer to *VIPP® data streams* in the *FreeFlow VI Compose User Guide* for further information.

Modes

This command is applicable in all modes.

Related commands

- %%EOF
- SOF_off
- STARTDBM
- STARTLM

%%BoundingBox

%%BoundingBox defines the size and position relative to its origin, of the image area of a segment. It is used at the beginning of a segment when it is called with **CACHE/SCALL**. Any mark outside the image area cannot appear on the page.

Xerox recommends that this statement be added at the beginning of a VIPP® job to inform printer controllers and output management services about the page area of the job. For example, use the following code sample when the job is going to be printed on US Letter paper:

```
%%BoundingBox: 0 0 612 792
```

Syntax

```
%%BoundingBox: llx lly urx ury
```

Where:

llx lly	are the coordinates in points, relative to the origin of the segment, of the bottom left corner of the image area.
urx ury	are the coordinates in points, relative to the origin of the segment, of the top right corner of the image area.

Examples

This example shows a segment drawing two concentric circles with the appropriate `%% BoundingBox` statement.

```
%%BoundingBox: -100 -100 100 100

POINT SETUNIT

-100 100 200 200 LT_S1 100 DRAWBR

-50 50 100 100 OW_S1 50 DRAWBR
```

Modes

This command is applicable in all modes.

Related commands

CACHE, SCALL

%%EOD_XGF

%%EOD_XGF is an end-of-data marker that terminates the preceding MAKEVMFILE, MAKEVMFORM or () RUN commands.

Syntax

```
%%EOD_XGF
```

Modes

This command is applicable in all modes.

Related commands

MAKEVMFILE, RUN

%%EOF

The %%EOF command is a End-of-file Document Structuring Convention (DSC) statement. In native mode, VI Compose takes no specific action upon it. It is nevertheless recommended to conform to DSC specifications.

In line mode, %%EOF causes VI Compose to perform a page eject and a RESET, to exit line mode and to resume native mode. VI Compose acts in the same way when the physical end of file is reached with no %%EOF, however, it is highly recommended that you include an %%EOF statement at the end of line mode data for a proper line mode exit. For further information, refer to *VIPP® data streams* in the *FreeFlow VI Compose User Guide*.

Syntax

```
%%EOF
```

End-of-file DSC statement.

Modes

This command is applicable in all modes.

Related commands

- %!
- RESET
- STARTDBM
- STARTLM

% %PagesPerBooklet

% %PagesPerBooklet indicates the number of pages per booklet for jobs made of booklets with a fixed number of pages. Place this marker at the beginning of a VIPP® job for printer controllers that rely on it to properly handle the booklets in the job.

Syntax

```
%%PagesPerBooklet: Number
```

Where:

number	is the number of pages per booklet.

Modes

This command is applicable in all modes.

Related commands

STARTBOOKLET, ENDBOOKLET

%%XGF

%%XGF identifies a Native Mode Prefix, NMP, record in line mode by default. NMP commands apply only to line printer or prefixed records.

NMP records allow native mode commands to be embedded in a line mode data stream in order to dynamically change the layout on a page-by-page basis. An NMP (%%XGF) must be placed at the first position of the record, or at the second position when PCC processing is enabled. For further information, refer to *VIPP® data streams* in the *FreeFlow VI Compose User Guide*.

%%XGF is the default value defined in `/usr/xgf/src/xgf.def`. Change the default value by editing this file or by using the SETNMP command in the JDT.

Syntax

```
%%XGF native mode commands
```

Examples

```
%!
(p1.jdt) STARTLM
............
............
last line of a page
<Form Feed>%%XGF (form127.frm) SETFORM
first line of next page
............
............
%%EOF
```

Note: Using NMP records may mean that data production and data presentation are no longer independent.

Tip: NMP records can also be used to embed comments or information in the line mode data. For example, the comment contained in this example does not print.

```
%%XGF % comment
```

Modes

This command is applicable in line mode.

Related commands

NMP_off, SETNMP

$$name.

The variable substitution string, $$name., consists of a variable name encapsulated between *$$* and *.* A variable name is either a built-in variable, or a variable set by SETVAR or by processing a database or XML file. A variable name defined in a database file can include spaces.

For more information, refer to the VSUB transform function description, and *VIPP® data streams* in the *FreeFlow VI Compose User Guide*.

[=name=]

The variable substitution string, [=name=], consists of a variable name encapsulated between [= and =] A variable name is either a built-in variable, or a variable set by SETVAR or by processing a database or XML file. For further information, refer to *VIPP® data streams* in the *FreeFlow VI Compose User Guide*. A variable name defined in a database file can include spaces.

For more information refer to the VSUB transform function description, and *VIPP® data streams* in the *FreeFlow VI Compose User Guide*.

BBOX

The BBOX marker is used to provide bounding box information to an in-line segment. It is placed at the very beginning of an in-line segment when the align parameter is used.

Syntax

```
{ llx lly urx ury } BBOX
```

Where:

llx lly urx ury	are the coordinates in points, relative to the origin of the segment, using the Lower Left (ll) x and y, and the Upper right (ur) x and y corners of the image area.

Examples

```
DOT3 SETUNIT
1200 2800 MOVETO
{
{ 0 0 240 219 } BBOX
15 15 MOVETO
(VIPPins.jpg) 1 0 10 ICALL
0 0 1000 910 R_S2 DRAWB
} 1 0 22 SCALL
```

EXPAND

The EXPAND marker is used to assist VIPP® Normalization of jobs containing VIPP® segments. When EXPAND is placed at the very beginning of a VIPP® segment a call to this segment is replaced in the Normalized file with an in-line segment containing only the marking commands useful for the call.

Syntax

```
EXPAND
```

When the segment has many external variables and complex nested logic it is more efficient to expand only the relevant code for each call, rather than calling the external segment preceded by all the external variable definitions.

EXTVAR

The EXTVAR marker is used to assist VIPP® Normalization of jobs containing VIPP® segments. You can place EXTVAR at the beginning of a VIPP® segment to declare the list of external variables.

Syntax

```
{ /VARname1 /VARname2 ... /VARnameN } EXTVAR
```

By default, Normalization inspects the segment code to determine the list of external variables. When the code is complex, this method cannot be accurate. EXTVAR provides an accurate way to explicitly declare the list of external variables.

4

Transform Functions

This chapter contains:

A VIPP® transform function is a sequence made up of one or more operands and a VIPP® transform function keyword. A VIPP® transform function operates some transformation on one of the operands and substitutes the entire sequence with the transformed operand. The sequence can replace any operand of any command or transform function, assuming that the value of the result is appropriate to replace that operand. This action allows transform functions to combine.

All transform functions, except F2S and SUBSTFONT, can be used in the Align procedure of an RPE entry. For more information, refer to RPE command information and the FROMLINE and RPEKEY descriptions.

2OF5

2OF5 reformats a string for printing a 2 of 5 interleaved barcode sequence. Insert **2OF5** between the basic string and the printing operator. You have previously selected the barcode font using **SETFONT** or **INDEXFONT**.

Syntax

```
(string) 2OF5 SHx
```

Where:

(string)	is a numeric string to be formatted
SHx	is one of the valid show commands

Note:

- **2OF5** barcode does not accept space or alpha characters and produces an error when one of those characters are processed in the input string.
- Use only the basic SH type commands when printing a barcode. To avoid unwanted interaction between the values in the barcode, the barcode transforms, and **DRAW** commands, use only the basic SH type commands when printing a barcode. Use of **SHP** or **SHMF** can lead to unexpected interaction if any characters in the barcode represent the attribute switch **SETFTSW** to the **SHP** or **SHMF** commands.

Examples

```
(1340224715) 20F5 SHL
```

The example provides this output:

```
!0K9R2”
```

The associated barcode font conform to a specific character mapping. To download fonts, refer to Font download.

Modes

This command is applicable in all modes.

Related commands

- CODE39
- CODE128 / EAN128
- INDEXFONT
- SETFONT
- SHX
- UPCA
- POSTNET

64TO256

64TO256 is a transform function that converts a Base64-encoded string into a binary string. It is intended to provide a way to place binary data, such as small images in data files and have them imaged with the **ICALL** or **SCALL** command.

Example of usage:

Assuming ImgString is a database field containing an image file converted to Base64, the image can be printed in the DBM using the following syntax:

```
/VARImage [ ImgString 64TO256 ] XGFRESDEF
x y MOVETO
(VARImage) 1 0 ICALL
```

Note: This function cannot process strings bigger than 65,535 characters. To process images that are larger than 65,000 split the images into chunks of equal sizes, except the last chunk, then place the chunks in several fields. Each chunk can be called sequentially in the **XGFRESDEF** statement.

Examples

```
/VARImage [ VARchk1 64TO256 VARchk2 64TO256 ...] XGFRESDEF
```

Modes

This command is applicable in all modes.

Related commands

XGFRESDEF, ICALL, SCALL

BIDI

BIDI processes a bi-directional data string for printing:

- The character stream is reversed for printing right to left.
- Context analysis and glyph substitution are performed according to the placement of the character in the word: initial, medial, final, isolated.
- Ligature is supported by identifying and substituting specific pairs or triplets.
- An optional switch to a different font for left-to-right characters is provided.
- An optional switch to a different font for Hindi or European digits is provided.

This command has replaced the depreciated command, ARABIC.

Syntax

```
(Bi-directional data string) opt BIDI (Print-ready string)
```

Where:

(Bi-directional data string)	is the input string containing bi-directional data stored from left to right and not yet processed for printing
opt	is one of the following, and it can be omitted:
	`0` base function; default if omitted
	`1` switch to a different font for left-to-right characters
	`2` substitute European digits for Hindi digits
(Print-ready string)	is the string delivered by the function; the string is ready to be processed by a printing command SHx, using the appropriate font

Examples

```
(Bi-directional data string) BIDI SHR
```

The **BIDI** transform function is configured by the **SETBIDI** command. A default configuration is coded at the end of the bi-directional configuration file located at `xgf/src/arb.def`.

Modes

This command is applicable in all modes.

Related commands

- SETBIDI
- BEGINARBT
- ENDARBT
- FCALL
- MOVETO
- SCALL
- ENDARBM

BSTRIP

You can set **BSTRIP** to strip or not strip extra leading and trailing blanks from a string.

Syntax

```
(string) option BSTRIP
(string) BSTRIP
```

Where:

Option	is one of:
0	do not strip anything
1	strip leading blanks
2	strip trailing blanks
3	strip both leading and trailing blanks; default if omitted

Examples

These are equivalent statements:

```
( Text with extra blanks ) BSTRIP SHL
(Text with extra blanks) SHL
```

These are equivalent statements:

```
( Text with leading blanks) 1 BSTRIP SHL
(Text with leading blanks) SHL
```

These are equivalent statements:

```
(Text with trailing blanks ) 2 BSTRIP SHL
(Text with trailing blanks) SHL
```

These are equivalent statements:

```
( Text with extra blanks ) 0 BSTRIP SHL
( Text with extra blanks ) SHL
```

Modes

This command is applicable in all modes.

Related commands

BSTRIP_off

BTRIM

BTRIM is a transform function that removes heading, trailing, and duplicate blanks inside a string.

Syntax

```
(string) BTRIM
```

Modes

This command is applicable in all modes.

Related commands

BSTRIP, BCOUNT

CASELOW

CASELOW changes all letters in a string to lowercase.

CASELOW supports the international character set. Mapping is based on the first character of the active font character names. The active font is the font that was last selected before the function execution. Before coding **CASELOW**, if the active font does not have the appropriate character names for case mapping, for example: /a/A, /aacute /Aacute, select an appropriate font using **SETFONT** or a font index.

Syntax

```
string CASELOW
```

Examples

This example prints john r. martin:

```
(John R. Martin) CASELOW SHL
```

Modes

This command is applicable in all modes.

Related commands

CASEUP, CASETI

CASETI

CASETI changes all letters in a string to the correct case for a title. All letters are changed to lowercase with the exception of the first letter in the string and any letter preceded by a space, a hyphen, a quote, a double quote, or a slash.

CASETI supports the international character set. Mapping is based on the first character of the active font character names. The active font is the font that was last selected prior to the function execution, so an appropriate font must be selected, using SETFONT or a font index before coding CASETI if the active font does not have appropriate character names for case mapping. For example, / a/A, /aacute /Aacute, and so on.

Syntax

```
string CASETI
```

Examples

This example prints John R. Martin.

```
(JOHN R. MARTIN) CASETI SHL
```

This example prints Scarlett O'Hara

```
(SCARLETT O'HARA) CASETI SH
```

Modes

This command is applicable in all modes.

Related commands

- CASELOW
- CASEUP
- SHL and SH
- SHx

CASEUP

CASEUP changes all letters in a string to uppercase.

CASEUP supports the international character set. Mapping is based on the first character of the active font character names. The active font is the font that was last selected prior to the function execution, so an appropriate font is selected using **SETFONT** or a font index before coding **CASEUP** if the active font does not have appropriate character names for case mapping. For example, / a/A, /aacute, /Aacute, and so on.

Syntax

```
string CASEUP
```

Examples

This example prints JOHN MARTIN.

```
(John Martin) CASEUP SHL
```

Modes

This command is applicable in all modes.

Related commands

- CASELOW
- CASETI
- SHL and SH
- SHx

CODE39

CODE39 reformats a string for printing a code 39 barcode sequence. **CODE39** is inserted between the basic string and the printing operator. The barcode font have been selected previously using **SETFONT** or **INDEXFONT**.

Syntax

```
(string) CODE39 SHx
```

Where:

(string)	is an alphanumeric string to be formatted
SHx	is a valid Show command.

Note: To avoid unwanted interaction between the values in the barcode, the barcode transforms, and **DRAW** commands, use only the basic **SH** type commands when printing a barcode. Use of **SHP** or **SHMF** can lead to unexpected interaction when any characters in the barcode represent the attribute switch from **SETFTSW** to the **SHP** or **SHMF** commands.

Examples

This example illustrates the syntax and the resulting printed output. You have the barcode font to print the actual barcode.

```
(800273400024) CODE39 SHL
*800273400024*
```

This command is used in RPE entries. For more information, refer to FROMLINE RPEKEY.

The associated barcode font conform to a specific character mapping. To download fonts, refer to Font download.

Modes

This command is applicable in all modes.

Related commands

- 2OF5
- EAN13/EAN8
- INDEXFONT
- SETFONT
- SHx
- UPCA
- POSTNET

CODE128 and EAN128

CODE128 and **EAN128** reformat strings for printing a CODE 128 or EAN 128 barcode sequences. **CODE128** or **EAN128** inserted between the basic string and the printing operator. The barcode font have been selected previously using **SETFONT** or **INDEXFONT**.

Syntax

`(string) CODE128` SHx

`(string) integer CODE128` SHx

`(string) EAN128` SHx

`[ (fix_length_string) (var_length_string) <F1> ... ) ] EAN128` SHx

Where:

string	is the alphanumeric string to be formatted. The length of the input string is variable. The input string can consist of the full ASCII range (0–127). Codesets A, B, and C are supported. Start and stop characters cannot be included in the input.
SHx	is a valid Show command.
integer	is an optional operand. The values can be: **0** This is the default. CODE128 uses codeset C when possible (compress a sequence of digits). It compresses numeric characters when it detects four or more adjacent digits **1** CODE128 is NOT compress any sequence of digits (always uses code set A or B)
(fix_length_ string)	is an EAN fixed length field (field ID is part of the string).
(var_length_ string)<F1>	is an EAN variable length field (field ID is part of the string). The field ID is included in the fields.

The output string consists of this information: start_char + code128_sequence + check_digit + stop_ char, in which:

- start_char is an appropriate Codeset (A, B, or C) start character.
- code128_sequence is an input string, compressed if possible, if parts of the input string were compressed, the output includes switch characters.
- check_digit is a checksum character.
- stop_char is a stop character.

Code sets A, B, and C are supported. Output string encoding maps either the Xerox encoding (x20- 7F, A1-AB) or BearRock encoding, x20-7F,95-DF. The command performs an automatic detection to select the appropriate encoding. Selection of Codeset A or B is driven by the data, code set C is the only user selectable option.

These general rules are used to determine code set selection:

- Code set A is used to encode all standard uppercase alphanumeric characters plus control characters.
- Code set B is used to encode all standard uppercase alphanumeric characters plus lowercase alphabetic characters.

- Code set C is used to encode pairs of numeric characters (00—99) as one character, which makes the barcode much more compact when it mainly contains numeric characters.
- When code set A or B can be used because data only contains uppercase alphanumeric characters, CODE128 uses code set B.

This command can also be used in RPE entries. For more information, refer to RPE command information and FROMLINE and RPEKEY command descriptions.

The associated barcode font conform to a specific character mapping. To download fonts, refer to Font download.

Note: To avoid unwanted interaction between the values in the barcode, the barcode transforms, and **DRAW** commands, when printing a barcode, use only the basic **SH** type commands. Use of **SHP** or **SHMF** can lead to unexpected interaction when any characters in the barcode represent the attribute switch **SETFTSW** to the **SHP** or **SHMF** commands.

Examples

These basic examples of CODE128 and EAN128 assume a database field named FIELD1:

```
/MB034 12 SETFONT
FIELD1 CODE128 SHL
FIELD1 EAN128 SHL
```

This is a JDT example:

```
/F128 /MB034 12 INDEXFONT
...
3 FROMLINE
[{BSTRIP 1 CODE128 SH} 0 300 0 1000 150 60 15 /F128 BLACK]
4 FROMLINE [...]
```

In the example above, a record portion (15 characters at position 60) is extracted from line 3. After striping the blanks it is printed as barcodes 128 at position '300 1000' using font F128.

EAN128 examples and information

To image an EAN128 symbol the application has to provide at least an application identifier (AI) and a data part (DATA). Assuming you have these as one field (Field1=AI+DATA) use this code:

```
Field1 EAN128 SH
```

When using two fields for AI and DATA use one of these code samples:

```
($$Field2.$$Field3.) VSUB EAN128 SH
[ Field2 Field3 ] EAN128 SH
```

To concatenate several AI+DATA fields into one symbol remember that DATA can be either fixed length or variable length.

Use these examples when DATA is fixed length to concatenate two fields such as FieldF1=AI1+DATA1 and FieldF2=AI2+DATA2 using the syntax VSUB, or an array:

```
($$FieldF1.$$FieldF2.) VSUB EAN128 SH
[ FieldF1 FieldF2 ] EAN128 SH
```

Use these samples when DATA is variable length to concatenate 3 fields including the variable length field FieldV3=AI3+DATA3 using an array syntax:

```
[ FieldF1 FieldV3 <F1> FieldF2 ] EAN128 SH
[ FieldF1 FieldF2 FieldV3 ] EAN128 SH
```

When it is not the last field, the variable length field followed by the function1 code which represented by <F1>. Although <F1> is the one byte string containing the value 0xF1 it is converted into the appropriate code by the EAN128 function.

This example illustrates a combination of multiple AI and DATA separated in different fields as well as AI being hard coded in the VIPP® code:

```
[ (01) Data01 (21) Data21 <F1> (10) Data10 ] EAN128 SH
```

Modes

This command is applicable in all modes.

Related commands

- INDEXFONT
- SETFONT
- SHx
- UPCA
- POSTNET

CS

Use CS to concatenate two strings into one.

Syntax

```
(string1) (string2) CS
```

Example

This example prints Dear Mr. Martin.

```
(Dear Mr.) (Martin) CS SHL
```

Mode

This command is applicable in all modes.

Related commands

VSUB, SETVAR

DAYS

This built-in transform function delivers an integer representing the number of days of the supplied date since January 1, 1970. It can be used in conjunction with **GETDATE**, **SHIFTDATE** and **SETDATE** to compute various date offsets.

Syntax

```
[ YYYY MO DD ] DAYS
```

Where:

YYYY	is the year (>1970). It can be an integer or a numeric string.
MO	is the month (1-12). It can be an integer or a numeric string.
DD	is the day. It can be an integer or a numeric string.

Example

This example assumes Year, Month, Day are field from a DBF file:

```
[ Year Month Day ] SETDATE
 /VARdate1 ($$D_DWL. $$D_MO./$$D_DD./$$D_YYYY.) VSUB SETVAR
 /VARstart [ Year Month Day ] DAYS SETVAR
VARstart'+'55 SETDATE
 /VARdate2 ($$D_DWL. $$D_MO./$$D_DD./$$D_YYYY.) VSUB SETVAR
GETDATE
(From $$VARdate1. to $$VARdate2. you will be given the opportunity to visit our new shopping
center and purchase any article with a 50% discount.) VSUB 0 SHP
```

Modes

This command is applicable in all modes.

Related commands

GETDATE, SHIFTDATE, SETDATE

EAN13 and EAN8

DRAWBC with its **/EAN13** and **/EAN8** options supersedes these **EAN13** and **EAN8** legacy commands, which are now deprecated, although still supported for backward compatibility. **EAN13** and **EAN8** reformat a string for printing an EAN barcode sequence. **EAN13** and **EAN8** are inserted between the basic string and the printing operator. The relevant font have been selected previously using **SETFONT** or **INDEXFONT**.

Syntax

`(SXXXXXXYYYYY) EAN13` SHx

`(XXXXYYY) EAN8` SHx

Where:

(SXXXXXXYYYYY)	is a 12-digit string consisting of these: **S** the number system **XXXXXX** the left part **YYYYY** the right part
(XXXXYYY)	is a seven-digit string consisting of these: **XXXX** the left part **YYY** the right part
SHx	is a valid Show command.

Note: The functions compute automatically and add a check digit at the end of the returned string. Input strings ending with a pre-calculated check digit of 13-digit or 8-digit strings are accepted but the check digit is ignored and replaced by the check digit that is computed by the function.

Note: To avoid unwanted interaction between the values in the barcode, the barcode transforms, and **DRAW** commands, use only the basic **SH** type commands when printing a barcode. Use of **SHP** or **SHMF** can lead to unexpected interaction when any characters in the barcode represent the attribute switch **SETFTSW** to the **SHP** or **SHMF** commands.

Examples

This example illustrates EAN13 syntax and the resulting printed output.

```
(800273400024) EAN13 SHL
8<ALCSOE=aaacec>
```

This example illustrates EAN8 syntax and the resulting printed output.

```
(7616424) EAN8 SHL
<HGBG=ecei>
```

Also use these commands in the Align procedure of an RPE entry. For more information, refer to RPE command information and to FROMLINE and RPEKEY.

The check digit is computed by the functions. The associated barcode font conform to a specific character mapping. To download fonts, refer to Font download.

Modes

These commands are applicable in all modes.

Related commands

- 2OF5
- CODE39
- INDEXFONT
- SETFONT
- SHx
- UPCA
- POSTNET

F2S

Use F2S to store a field delimited record into a table using the **ADD** command.

Syntax

```
(fdl1:fdl2:...:fdln) (delim) F2S (fdl1) (fdl2) ... (fdln)
(fdl1:fdl2:...:fdln) (delim) { transform code } F2S (fdl1) (fdl2) ... (fdln)
```

Examples

This example of F2S produces (John) (Smith) (Seattle).

```
(John:Smith:Seattle) (:) F2S
```

This example, with ADD, assumes RECORD is a string with field delimited data:

```
/VARtable [[ RECORD (:) F2S ]] ADD
```

In this example all fields are converted to uppercase:

```
[ (aa:Bc:cC) (:) { CASEUP } F2S ]
```

It delivers:

```
[ (AA) (BB) (CC) ]
```

Modes

This command is applicable in all modes.

Related commands

ADD, GETITEM, FOREACH

FORMAT

Use **FORMAT** to format a numeric string for printing by one of the **SHx** commands. **FORMAT** supports enclosing negative numbers in parenthesis. To do so, a pair of parenthesis are present in the format string and the /FNSign parameter are assigned to the closing parenthesis (41).

Syntax

```
(numeric data) (format) FORMAT SHx
(numeric data) (format) [ /param1 value1 /param2 value2 ... ] FORMAT SHx
```

Where:

numeric data is a string containing numeric data, non numeric characters are discarded.

format is a string containing meta characters describing the desired output. Static characters can also be included in the string. The meta characters are defined using specific parameters and can be set by default using the SETPARAMS command or included in the optional array described below.

The format parameters are:

- /DecimalPoint 4 decimal delimiter in numeric data (default: .)
- /NSign 45 negative sign in numeric data (default: -)
- /FDecimalPoint 46 decimal delimiter in format (default: .)
- /FNSign 45 negative sign in format (default: -)
- /FPSign 43 positive sign in format (default: +)
- /FPunctuation 44 thousands delimiter in format (default: ,)
- /FDigit 35 placeholder for digit in format (default: #)
- /FLZDigit 64 placeholder for digit in format (default: @) that are replaced by space if the digit is a leading zero

Note: The value of each parameter is the ASCII decimal value of the character. A value of null indicates that the corresponding character is undefined. When **FNSign** is used with positive numeric data, it generates a space. When **FPSign** is used with negative numeric data, it is replaced by **FNSign**. In the output string, all metacharacters are substituted, removed, or preserved, as appropriate. All non-metacharacters are preserved. Default metacharacters are defined in `/usr/xgf/src/xgf.def` and vary depending on the default media choice at installation time. If the VIPP® job is intended to be rendered on multiple platforms, it is recommended that the metacharacters be defined in the job.

[/paramx valuex ...] This optional parameter array allows format parameters to be set temporarily for this statement only, without impacting the rest of the code.

Examples

This example contains a negative sign (-) after the last number (#) character. This example prints `$1,234.56-`.

```
(-00001234.56) ($@,@@@,@@#.##-) FORMAT SHr
```

These are examples of FORMAT with alignment on the decimal point:

```
[ /FNSign 41 ] SETPARAMS
(123456.78) ((@@@,@@@,@@@,@@#.##)) FORMAT 4 SHMF
```

prints: 123,456.78

```
(-123456.78)((@@@,@@@,@@@,@@#.##)) FORMAT 4 SHMF
```

prints: (123,456.78)

This example:

```
(1234567890) (@@,@@@,@@#.##) [ /FDecimalPoint null ] FORMAT SHx
```

prints: 12,345,678.90

Modes

This command is applicable in all modes.

Related commands

SETPARAMS, SETVAR, SHx

GETINTV

GETINTV extracts a substring or a field from a string.

Syntax

```
(string) recpos length GETINTV
(string) field_nr (field_sep) GETINTV
```

Where:

string	is the source string for the extraction.
recpos	is the position of the first character to be extracted (starting with zero).
length	is the length of the substring. When length is negative, the extraction is performed starting from the last character from right to left instead of left to right.
field_nr	is the field number starting with zero.
field_sep	is the field separator string. field_sep can specify a separator string different than the one specified by the current **SETDBSEP** command.

Examples

To print only the first name from the string Mr. John Martin, use this.

```
(Mr. John Martin) 1 ( ) GETINTV SHL
```

This example produces the string, ABC from the string ABCDEF.

```
(ABCDEF) 0 3 GETINTV
```

This example produces the string, DEF from the string ABCDEF.

```
(ABCDEF) 0 -3 GETINTV
```

This example selects the month from the string, 10/02/00, and prints 10. A previous **GETINTV** or **GETFIELD** command could have been used to capture this field from a larger string or field.

```
(10/02/00) 0 (/) GETINTV SHL
```

In this example, the date is separated into the month, day, and year and assigns variables for later use.

```
/VAR_month (10/02/00) 0 (/) GETINTV SETVAR
/VAR_day (10/02/00) 1 (/) GETINTV SETVAR
/VAR_year (10/02/00) 2 (/) GETINTV SETVAR
```

GETINTV can also be used in a DBM to process lines with fixed length fields typically line mode data. This example shows the construction of a database file using line mode data.

```
%!
(fixed.dbm) STARTDBM
MY_RECORD
John Mary Snow
Tim Victor Sand
%%EOF
```

In the DBM, fields are separated and assigned to variables for later use using the code below. The example shows FIXED length field, where the first names are in column 0 for 6 bytes, the middle

name is in column 6 for 11 bytes, and the last name is in column 17 for 8 bytes. It is important to align the data in this type of fixed position order.

```
/VARname MY_RECORD 0 6 GETINTV SETVAR
/VARname2 MY_RECORD 6 11 GETINTV SETVAR
/VARsurface MY_RECORD 17 8 GETINTV SETVAR
```

Substring expression

As an alternative to **GETINTV**, a substring expression can be applied to a variable or field name with the following syntax:

```
variable_name|recpos,length|
variable_name|field_nr,field_sep|
variable_name|field_nr,0xhexfield_sep|
```

Where

recpos, length, field_nr, field_sep	Have the same meaning as in the GETINTV syntax.

A substring expression must not contain any of the following characters:

space / ( ) < > { } [ ] %

0xhexfield_sep can be used if the separator contains any of the above by coding the hexadecimal value of the string preceded by the 0x tag.

Examples:

```
/VAR1 (date: 07/12/2012) SETVAR

VAR1|8,4| SH       % prints (/12/)

VAR1|1,:| SH       % prints ( 07/12/2012)

VAR1|2,0x2F| SH    % prints (2012)

([=VAR1|2,0x2F|=]:[=VAR1|1,0x2F|=]:[=VAR1|6,2|=]) VSUB SH     % prints (2012:12:07)
```

Modes

This command is applicable in all modes.

Related commands

SETDBSEP, GETFIELD

HMS

Use **HMS** to convert time measured in seconds to the HH:MM:SS format.

Syntax

```
(integer) HMS
integer HMS
```

Examples

This example prints `4456` as `1:14:16`.

```
(4456)HMS SHL
```

Modes

This command is applicable in all modes.

Related commands

SHx

NOHYPHEN

NOHYPHEN is a transform function that prevents hyphenation on spaces for text printed with the **SHP** or **SHp** commands. It is intended to be applied on variables embedded within and **SHP** text.

Syntax

```
string NOHYPHEN
```

Examples

```
/VAR_DATE DATE NOHYPHEN SETVAR
(... will be delivered on $$VAR_DATE. no later than 16:00 ...) VSUB SHP
```

Modes

This command is applicable in all modes.

Related Commands

SHP and SHp, SHT and SHt

POSTJPN

POSTJPN reformats a string for printing a Japanese Postal barcode sequence. This command is inserted between the input string and a VIPP® printing command. Select a Japanese Postal barcode font using **SETFONT** or **INDEXFONT** prior to inserting this command.

Syntax

`(string) POSTJPN` SHx

Where:

String	are an alpha-numeric string made of a postal code followed by an address code. The following characters are allowed: **0–9** **hyphen** **A-Z** **a-z (allowed but converted to uppercase)** Any other character raises a VIPP®_invalid_contents error. The postal code are 7 numeric characters. The address code can be alphanumeric and is not limited in length but are truncated to 13 postal barcode modules. A numeric character or hyphen is one module, an alphabetic character is 2 modules.
SHx	is one of the VIPP® commands for text printing.

Comply with Japan Postal Service specifications

In Japan, the input string comply with Japan Postal Service specifications, refer to http:// www.post.japanpost.jp/ (Japanese only).
Postal code accuracy is determined by the application creating the data stream.

The output string consists of:

`start_code` + `barcode modules` + `check_digit` + `stop_code`

The associated barcode font conform to specific character mapping. To purchase Japanese postal barcode fonts contact the Fuji Xerox representative, the fonts are only available in Japan.

Use only the basic SH type commands when printing a barcode

To avoid unwanted interaction between the values in the barcode, the barcode transforms, and **DRAW** commands, use only the basic **SH** type commands when printing a barcode. Use of **SHP** or **SHMF** can lead to unexpected interaction when any characters in the barcode represent the attribute switch (SETFTSW) to the **SHP** or **SHMF** commands.

Examples

```
(100000131-3-2-503SMITH) POSTJPN SH
```

Modes

This command is applicable in all modes.

Related commands

SETFONT, INDEXFONT, SHx

POSTNET

POSTNET reformats a string for printing a PostNet barcode sequence. This command is inserted between the input string and a VIPP® printing command. You can previously select a PostNet barcode font using **SETFONT** or **INDEXFONT**.

Syntax

`(string) POSTNET` SHx

Where:

string	are a numeric string of length 5, 9 or 11. Lengths of 10 and 12 are also supported with a dash in the sixth position.
SHx	is one of the VIPP® commands for text printing.

Comply with U.S. Postal Service specifications

In the United States, the input string comply with U.S. Postal Service specifications for Zip, Zip+4 or Zip+4+DPBC. For more details, refer to www.usps.gov.

Postal code accuracy is determined by the application creating the data stream.

Use only the basic SH type commands when printing a barcode

To avoid unwanted interaction between the values in the barcode, the barcode transforms, and **DRAW** commands, use only the basic **SH** type commands when printing a barcode. Use of **SHP** or **SHMF** can lead to unexpected interaction when any characters in the barcode represent the attribute switch (SETFTSW) to the **SHP** or **SHMF** commands.

The output string consists of:

- frame_char + input_string + check_digit + frame_char
- frame_char can encoded on position 42 (asterisk)

The associated barcode font conforms to specific character mapping. To download fonts, refer to Font download.

Modes

This command is applicable in all modes.

Related commands

SETFONT, INDEXFONT, SHx

QSTRIP

QSTRIP strips the first and last characters from a string. It can be used as an alternative to QSTRIP_on in database mode when only some fields need to be stripped. This command has no action when the string length is less than two characters.

Syntax

```
(string) QSTRIP
```

Examples

This example prints John Martin without the quotation marks.

```
("John Martin") QSTRIP SHL
```

Modes

This function is applicable in all modes.

Related commands

- BSTRIP
- BSTRIP_off
- GETINTV
- QSTRIP_on
- SHx

REPLACE

REPLACE is a transform function that can be used to replace occurrences of a given substring with another substring within a defined input string.

Syntax

```
(input string) (str1) (str2) REPLACE (output string)
```

Where:

input string	is the original input string
str1	is the substring to be replaced if found
str2	is the new substring to replace str1
output string	is the string delivered by the function

Examples

```
(Hello:dear:friends) (:) (|) REPLACE SH
```

will print:

```
Hello|dear|friends
```

Modes

This command is applicable in all modes.

Related commands

None

ROUND

Use this command to round a numeric variable to the nearest digit at a given position.

Syntax

```
/VARname round_digit ROUND
```

Where:

VARName	is the name of a variable previously initialized with SETVAR.
round_digit	is the rounding position starting from the decimal point. It can be positive or negative.

Examples

```
/VAR1 (1234.5678) SETVAR
/VAR1 3 ROUND % VAR1 is now: (0001234.568)
/VAR1 2 ROUND % VAR1 is now: (0001234.57)
/VAR1 0 ROUND % VAR1 is now: (0001235)
/VAR1 -2 ROUND % VAR1 is now: (0001200)
```

Modes

This variable is applicable in all modes.

Related commands

ADD, SUB, DIV

SUBSTFONT

SUBSTFONT is used for font substitutions. Use this command in the font lists stored in one of the directories referenced by **SETEPATH**. It is only effective on PostScript level 2 devices.

Syntax

```
/font1 /font2 SUBSTFONT
```

SUBSTFONT delivers /font1 if this font is available on the device, if it is not available it delivers /font2.

Define substitutions to avoid default substitution, generally with Courier or error when the font is not available on the device. font2 are as close as possible to font1 so that the VIPP® output is only slightly affected.

Examples

In this example, Helvetica-light is used when it is available, otherwise, Helvetica is used.

```
/Helvetica-light /Helvetica SUBSTFONT
```

Modes

This function is applicable in all modes.

Related commands

SETENCODING

TRIO

TRIO is transform function intended to address a compatibility issue between **LCDS** fonts and **PS/PCF** fonts in the Chinese market. It only supports **EBCDIC** data streams.

The input string is expected to contain 0x0E and 0x0F bytes representing switches between Chinese and Western, for example, English text, to which **TRIO** adds **EBCDIC** spaces (0x40) and possibly **INDEXFONT** switches.

TRIO is intended for use in conjunction with **SHMF**.

Syntax

```
(EBCDIC string) TRIO
```

Examples

This processing example:

```
(-eng-<0E>-chn—<0F>-eng-) TRIO
```

...delivers:

```
(-eng-<40><0E>-chn-<40>-eng-)
```

... with an internal English font.

Or, this example can be used with an external English font assuming that the font switch has been set to (//) e SETFTSW.

```
(-eng-<40>//FI0<0E>-chn-//FIx<40>-eng-)
```

Use the following convention on the fonts defined by INDEXFONT:

The Chinese font are assigned an index ending with 0 (zero). It can be associated in the same string with any English font using the same index but not ending with 0. Therefore:

- If the current **INDEXFONT** ends with zero, the internal English font from the Chinese font will be used.
- If the current **INDEXFONT** does not end with zero, it will be used as an English font and the matching index ending with zero as Chinese font.

```
(//) 3 SETFTSW
/F00 /MSung5550-Light 12 INDEXFONT % Chinese font
/F01 /NCR 10 INDEXFONT             % Western/English font
1 SETRPEPREFIX
10 BEGINRPE
...
/4 RPEKEY                          % line 4 (font index 4)
[{TRIO 0 SHMF} ..... /F00 BLACK]   % use only F00
/5 RPEKEY                          % line 5 (font index 5)
[{TRIO 0 SHMF} ..... /F01 BLACK]   % start with F00 and use F01 for English
```

Modes

This command is applicable in all modes.

Related commands

SHMF,SHMf, and SHmf, INDEXFONT, SETFTSW

UPCA

UPCA reformats a string for use when printing a UPC version A barcode sequence. Insert `UPCA` between the basic string and the printing operator. A UPCA barcode font is required. Select a UPCA barcode using **SETFONT** or **INDEXFONT**.

Syntax

`(SXXXXXYYYYY) UPCA` SHx

Where:

(SXXXXXYYYYY)	is an 11-digit string made up of these elements: • `S` represents the number system • `XXXXX` specifies the vendor number • `YYYYY` specifies the product number
SHx	represents a valid print command

The check digit is computed by the function. The associated barcode font conform to a specific character mapping. To download fonts, refer to Font download.

Note: To avoid unwanted interaction between the values in the barcode, the barcode transforms, and **DRAW** commands, use only the basic **SH** type commands when printing a barcode. Use of **SHP** or **SHMF** can lead to unexpected interaction when any characters in the barcode represent the attribute switch **SETFTSW** to the **SHP** or **SHMF** commands.

Examples

The following are examples of the command structure and resulting output:

```
(08978204466) UPCA SHL
0<kIJHIC=aeeggw>2
```

Modes

This command is applicable in all modes.

Related commands

- 2OF5
- CODE39
- EAN13/EAN8
- INDEXFONT
- SETFONT
- SHx
- POSTNET

UTF8TOLOC

UTF8TOLOC is a transform function used to convert a string from UTF8 to local encoding.

Syntax

`(string) local_code UTF8TOLOC`

Where

`local code` is one of the following:

1	ISO-8859-1 or Windows-1252 (Western European)
2	ISO-8859-2 (Central European)
3	Windows-1250 (Central European)
4	ISO-8859-9 (Turkish)
5	Windows-1251 (Cyrillic)
6	Windows-1258 (Vietnamese)
7	ISO-8859-11 or Windows-CP874 or TIS-620 (Thai)
8	Windows-CP866 (Cyrillic)
9	ISO-8859-15 (Latin-9)
10	Mac OS Roman
11	Windows-1256 (Arabic)
12	Windows-1255 (Hebrew)

Example

`ImageName 1 UTF8TOLOC ICALL`

`ImageName` is the field name of a variable image in a UTF8–encoded database file.

Mode

This command is applicable in all modes.

Related Commands

/LocalToUTF8

VSUB

VSUB substitutes variable and text file references with their contents in a string.

Use this command together with the commands that use a string as an operand, for example, **SHx**, **SETFORM**, **SETMEDIA**, and so on, in a DBM, a VIPP® form, or a native mode job.

A variable reference consists of a variable name encapsulated between `$$` and `.`, or between `[=` and `=]`. A variable name is a built-in variable, or a variable set by **SETVAR** or by processing a database or XML file. For more information, refer to VIPP® data streams in the *FreeFlow VI Compose User Guide*. A variable name defined in a database file can include spaces.

A text file reference is made up of a text file name encapsulated between `[=` and `=]`.

When a reference is not a variable, it is looked up as a file name in the directories defined by the **SETMPATH** or **SETPPATH** project mode commands. The text files referenced in this manner require plain text consisting of eventually valid VIPP® text attribute switches, as defined by the INDEXxxx commands, and variable place holders `$$` and `.`, or `[=`and `=]`. The content of the text file is inserted in the delivered string without changes, except for the substitution of variable references. A typical use of **VSUB** with text file references is to import text blocks from a collection of external text files and print the text blocks on the page using **SETLKF** and **SHP**.

The total length of the string delivered by **VSUB** cannot exceed 65535 characters.

Syntax

```
(printable data with variable and/or text file references) VSUB
(string to merge) (string holding $$.) VSUB
```

Examples

When **FNAME** and **LNAME** contain John and Smith, the following examples deliver Dear John Smith.

```
(Dear [=FNAME=] [=LNAME=],) VSUB 0 SHP
(Dear $$FNAME. $$LNAME.,) VSUB 0 SHP
(John Smith) (Dear $$.,) VSUB 0 SHP
```

In this example, **TITLE** and **NAME** have been assigned to Mr., and Martin by the current record of a database file, and this sequence prints as: As you know, Mr. Martin, you have won, . . .

```
(As you know, $$TITLE. $$NAME., you have won, ...) VSUB SHL
```

Use **VSUB** in an RPE entry align procedure to merge the RPE field in a string by leaving out the variable name between `$$` and `.`.

The following example uses **VSUB** in an RPE align procedure to merge a field into a string and print it.

```
[ { (Amount: -$$.-) VSUB SH } ..... 25 8 /F1 BLACK ]
```

Assuming that the field specified by the start of position 25 and length of 8 characters in the example contains 1,234.50, this entry prints:

```
Amount: -1,234.50-
```

These examples show how to insert a text file into a paragraph:

```
(Your conditions are: [=cnd23.txt=] and [=cnd54.txt=].) VSUB 0 SHP
```

When the file name is contained in a variable, like in the fields CND1 and CND2 in this example, variable and text file substitution can be combined as follows:

```
(Your conditions are: [=$$CND1.=] and [=$$CND2.=].) VSUB 0 SHP
```

Modes

This command is applicable in all modes.

Related commands:

- SETVAR
- STARTDBM
- VSUB2
- VSUB3
- SETMPATH
- SETPPATH

VSUB2

VSUB2 executes VSUB. **VSUB2** considers the resulting string as a variable name and delivers its value. If the variable does not exist, an undefined error occurs.

Variable names are encapsulated between `$$` and `.`, or between `[=` and `=]`.

Syntax

```
(printable data with variable references) VSUB2
(string to merge) (string holding $$.) VSUB2
```

Examples

In this example, when the contents of FIELD 7 is 5, the resulting variable name is VARcode5 and the variable content May prints.

```
/VARcode5 (May) SETVAR
....
(VARcode$$FIELD7.) VSUB2 SHL
```

Use **VSUB2** in an RPE entry align procedure to merge the RPE field in a string by leaving out the variable name between `$$` and `.`.

This example uses VSUB2 in an RPE align procedure to merge a field into a string, retrieve the variable value, and print it. Assuming that the field specified by the start of position 25 and length of 2 characters in the following example contains 02, this example prints February.

```
/VARmonth01 (January) SETVAR
/VARmonth02 (February) SETVAR
...
10 BEGINRPE
...
[ { (VARmonth$$.) VSUB2 SH } ..... 25 2 /F1 BLACK ]
```

Modes

This command is applicable in all modes.

Related commands:

- SETVAR
- STARTDBM
- VSUB
- VSUB3

VSUB3

VSUB3 executes **VSUB**, then **VSUB3** considers the resulting string as a variable name and, if the variable exists, **VSUB3** delivers its value. If the variable does not exist, **VSUB3** delivers the input string.

Variable names are encapsulated between `$$` and `.`, or between `[=`and `=]`.

Syntax

```
(input string) (string holding $$.) VSUB3
```

Use **VSUB3** in an RPE entry align procedure to merge the RPE field in a string by leaving out the variable name between `$$` and `.`

Examples

```
/VARmonth01 (January) SETVAR
(month01) (VAR$$.) VSUB3
```

produces January, but

```
(month13) (VAR$$.) VSUB3
```

produces month13

Modes

This command is applicable in all modes.

Related commands

- VSUB
- VSUB2
- SETVAR
- STARTDBM

VSUB4

FreeFlow VI

VSUB4 has the same behavior as **VSUB**, and **VSUB4** suppresses variable blank lines in a text block. This command is intended for use in name and address blocks but can be used in other text blocks.

A variable blank line is a line delimited by an end-of-line character and contains one or more variables that are evaluated as being blank, empty, or containing spaces only.

Note: The `0 SHP` option is still available and can be used for individual print lines in an address block. **VSUB4** now allows you to create the complete address block in a single **SHP/SHMF** command.

Syntax

```
(printable data with variable and / or text file references) VSUB4 0 SHP
```

Examples

```
/VAR1 (David Kirk) SETVAR
/VAR2 (12 Baker Street) SETVAR
/VAR3 (Los Angeles, CA 90245) SETVAR
300 300 MOVETO
( $$VAR1.
$$VAR2.
$$VAR3 ) VSUB4 0 SHP
```

The example prints the following text, with no suppression because there are no empty variables:

David Kirk

12 Baker Street

Los Angeles, CA 90245

```
/VAR1 (David Kirk) SETVAR
/VAR2 () SETVAR
/VAR3 (Los Angeles, CA 90245) SETVAR
300 300 MOVETO
( $$VAR1.
$$VAR2.
$$VAR3 ) VSUB4 0 SHP
```

The example prints the following text, in which **VAR2** is suppressed because it is empty:

David Kirk

Los Angeles, CA 90245

5

Variables

This chapter contains:

A VIPP® variable is a keyword, either built-in or defined using SETVAR, that represents a value subject to change throughout the job. It can replace any operand of any command or transform function assuming the value of the variable is appropriate to replace that operand. Some built-in variables, such as RPEPOS, can require operands for themselves. Variables can be of these PostScript types, as defined by the PostScript language:

- String
- Integer
- Real number
- Boolean
- Array
- Procedure
- Name

Built-in variables can also be of these VIPP® types:

- Form
- GEPkey
- Colorkey

VIPP® Variables by Type

The following table provides a list of all VIPP® built-in variables and lists each variable type.

	Type						
Built-in Variable	Boolean	Color	Form	GEPKey	Integer	Real	String
AUTOGRID			X				
BACK	X						
BCOUNT					X		
BLGRID			X				
BPCOUNT					X		
CLIP				X			
COLW						X	
CPCOUNT					X		
CURLINE					X		
D_DD						X	
D_DOY						X	
D_DWL						X	
D_DWS						X	
D_MO						X	
D_MOL						X	
D_MOS						X	
D_YY						X	
D_YYYY						X	
DEVRES					X		
DJDECMD							X
DJDEPAR							X
FRCOUNT					X		
FRLEFT						X	
GLT						X	
GRIDSKIP	X						
HCOLOR		X					
HDISP						X	
HPOS						X	
HPOS2						X	

	Type						
Built-in Variable	Boolean	Color	Form	GEPKey	Integer	Real	String
IHEIGHT						X	
IWIDTH						X	
LNCOUNT					X		
LPCOUNT					X		
LPINDEX					X		
LSP						X	
MPR						X	
OTCLIP and ITCLIP				X			
PAGEH						X	
PAGEW						X	
PDFDEVICE	X						
PDFPAGES					X		
PLINES					X		
PPCOUNT					X		
PREV / NEXT							
PRODUCT							X
PSIZE						X	
RPCOUNT					X		
RPEPOS						X	
RPLEFT					X		
SHEETH						X	
SHEETW						X	
SHPOS						X	
SLENGTH					X		
SSIZE						X	
SVPOS						X	
T_AMPM							X
T_HH							X
T_HH2							X
T_MM							X
T_SS							X
T_TZN							X

	Type						
Built-in Variable	Boolean	Color	Form	GEPKey	Integer	Real	String
TLENGTH			X				
TLGRID			X				
TPATH				X			
UV2L (Two-Layer UV Effect)		X					
VARDataFileName							
VDISP						X	
VPOS						X	
XGFVER							X
XMLATN							X
XMLATV							X
XMLDTH					X		
XMLPAR							X
XMLPATH							X
XMLTAG							X
XMLVAL							X
YINIT						X	
ZSPAGE					X		
ZSRECNUM					X		
ZSREPCNT					X		
ZSREPIDX					X		

AUTOGRID

The built-in form, **AUTOGRID**, enables printing of a line mode test print file overlaying a line and column grid.

Use this feature to help determine the record position in the print file when coding **RPE** definitions. Use **AUTOGRID** in a **JDT** with proper grid, margin, and font settings. For an example, refer to the `/usr/xgf/ jdtlib/autogrid.jdt` file.

Syntax

```
(AUTOGRID) SETFORM
```

Examples

```
80 70 SETGRID
130 130 130 130 SETMARGIN          % to allow line/column number printing
(AUTOGRID) SETFORM
/NCR 0 SETFONT
```

Tip: Use **NMP_off** to disable Native Mode Prefix (NMP) command side effects, if any. The grid is printed using the current highlight color on Highlight Color systems.

Modes

This built-in form is applicable in line mode.

Related commands

- NMP_off
- SETFONT
- SETFORM
- SETGRID
- SETMARGIN
- STARTLM

BACK

BACK is a built-in boolean variable that is true when the current page is on the back of the current sheet and false when the current page is not on the back of the current sheet.

Syntax

```
BACK
```

Examples

- `IF BACK { ... action on back pages ... } ENDIF`
- `IF BACK not { ... action on front pages ... } ENDIF`

Note: Information provided by this variable is relative to the current page when used in **BEGINPAGE**, and relevant to the forthcoming page when used in **/P ENDPAGE**.

Modes

This variable is applicable in all modes.

Related Commands

- DUPLEX_on
- SETBFORM
- TUMBLEDUPLEX_on

BCOUNT

BCOUNT delivers the number of blanks in a string.

Syntax

```
(string) BCOUNT
```

Modes

This command is applicable in all modes.

Related commands

BSTRIP, BTRIM

BLGRID

The built-in form, **BLGRID**, enables the printing of a VIPP® job sample on top of a unit grid with a bottom left origin. This grid is useful in native mode form design.

BLGRID helps locate print positions on the page when coding a VIPP® job such as a form, **RPE**, or native mode. The unit used is the current one set by **SETUNIT**.

Syntax

```
(BLGRID) SETFORM
```

The origin (0,0) of the grid is related to the margins. Setting the margins to zero is advised in most cases.

Examples

```
0 0 0 0 SETMARGIN
(BLGRID) SETFORM
```

Tip: Increment **SETMAXFORM** to print the grid on top of other forms. On Highlight Color systems, the grid is printed using the current highlight color.

Modes

This built-in form is applicable in all modes.

Related commands

- SETFORM
- SETMARGIN
- SETMAXFORM
- SETUNIT
- TLGRID

BPCOUNT

This built-in variable provides the page count for the current booklet. The number of pages printed since the last **STARTBOOKLET** command.

Syntax

```
BPCOUNT
```

Modes

This command is applicable in all modes.

Related commands

STARTBOOKLET, ENDBOOKLET

CLIP

CLIP is a reserved, built-in, GEPkey used for clipping. Use **CLIP** with a command that uses a GEPkey such as **DRAWB**, **DRAWBR**, **DRAWPOL**, or **SHx**. The area defined by this command becomes the new clipping area.

The clipping area is the area on the page where printing is enabled. Any mark outside the clipping area is not imaged. The default is the entire page.

To disable clipping and restore full-page printing, use **ENDCLIP**. To disable clipping only, use **PAGEBRK**.

Syntax

```
CLIP
```

Examples

```
[ 200 200 1200 3400 2300 200 ] CLIP DRAWPOL
```

Everything outside of the triangle is not printed after the command.

Tip: You can use the built-in GEPkey to clip patterns. Items already marked on the page cannot be clipped.

Modes

This built-in GEPkey is applicable in all modes.

Related commands

- DRAWB and DRAWBR
- DRAWPOL
- ENDCLIP
- SHx

COLW

COLW is a built-in dynamic variable that provides the current column width as defined by **SETCOLWIDTH**. **COLW** can be used with BATkey definitions.

Syntax

```
COLW
```

Examples

This example draws a horizontal line with the length equal to the current column width.

```
100 200 COLW 0 S1 DRAWB
```

Modes

This variable is applicable in all modes.

Related commands

SETCOLWIDTH

CPCOUNT

CPCOUNT is a built-in variable that provides the current copy number. Use this variable to perform specific actions on a particular copy pass.

Syntax

```
CPCOUNT
```

Examples

In this example, back form is cancelled on the third copy, otherwise, back form produces blank back pages.

```
3 SETCYCLECOPY
(bform.frm) [1 2] SETBFORM
{IF CPCOUNT 3 eq {0 SETMAXFORM} ENDIF }BEGINPAGE
```

Modes

This variable is applicable in all modes.

Related commands

- BEGINPAGE
- ENDIF
- SETBFORM
- SETCYCLECOPY
- SETMAXBFORM

CURLINE

CURLINE is a built-in variable that provides the line number of the current line being processed by an **RPE** entry. This variable is used in **SETRCD** tests only. For other related **RPE** commands, and for more information, refer to RPE command information.

Syntax

```
CURLINE
```

Examples

```
/IF_1STLINE { CURLINE 1 eq } SETRCD
```

Modes

This variable is applicable in line mode only.

Related commands

SETRCD

D_DD

D_DD is used in conjunction with the **GETDATE** command to set the appropriate day of the month from 1–31 for the application.

Syntax

```
D_DD
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_DOY

D_DOY is used in conjunction with the **GETDATE** command to set the appropriate numerical day of the year 1–366 for the application.

Syntax

```
D_DOY
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_DWL

D_DWL is used in conjunction with the **GETDATE** command to set the unabbreviated week date, for example, Monday, Tuesday, and so on for the application.

Syntax

```
D_DWL
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_DWS

D_DWS is used in conjunction with the **GETDATE** command to set the appropriately abbreviated week date Mon, Tue, and so on for the application.

Syntax

```
D_DWS
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_MO

D_MO is used in conjunction with the **GETDATE** command to set the numerical month name, for example, 01–12 for the application.

Syntax

```
D_MO
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_MOL

D_MOL is used in conjunction with the **GETDATE** command to set the appropriate unabbreviated month name, for example, January, February, and so on for the application.

Syntax

```
D_MOL
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_MOS

D_MOS is used in conjunction with the **GETDATE** command to set the appropriately abbreviated month name, for example, Jan, Feb, and so on for the application.

Syntax

```
D_MOS
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_YY

D_YY is used in conjunction with the **GETDATE** command to set the appropriate two-digit year for the application.

Syntax

```
D_DYY
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

D_YYYY

D_YYYY is used in conjunction with the **GETDATE** command to set the appropriate four-digit year for the application.

Syntax

```
D_DYYY
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

DEVRES

DEVRES delivers the current device resolution in dpi (dots per inch).

Syntax

```
DEVRES
```

Examples

```
(Resolution: $$DEVRES.) VSUB SH
```

Modes

This built-in variable is applicable in all modes.

Related commands

GETDATE

DJDECMD

DJDECMD is a built-in variable that represents a **DJDE** keyword that has been extracted from a **DJDE** record by the **PROCESSDJDE** command. **DJDECMD** is available only inside the djde_proc operand of a **PROCESSDJDE** command. **DJDECMD** is always associated with the corresponding **DJDEPAR** variable.

Syntax

```
DJDECMD
```

Examples

```
IF DJDECMD (FORM) eq { ($$DJDEPAR..frm) VSUB SETFORM } ENDIF
```

Modes

This variable is applicable in line mode only.

Related commands

- DJDEPAR
- PROCESSDJDE
- CASE
- IF/ELSE/ELIF/ENDIF

DJDEPAR

DJDEPAR is a built-in variable that represents a **DJDE** parameter that has been extracted from a **DJDE** record by the **PROCESSDJDE** command. **DJDEPAR** is available only inside the djde_proc operand of a **PROCESSDJDE** command. **DJDEPAR** is always associated with the corresponding **DJDECMD** variable.

Syntax

```
DJDEPAR
```

Examples

```
IF DJDECMD (FORM) eq { ($$DJDEPAR..frm) VSUB SETFORM } ENDIF
```

Modes

This variable is applicable in line mode only.

Related commands

DJDECMD, PROCESSDJDE, VSUB

FRCOUNT

FRCOUNT is a built-in variable that provides the current frame number. Use this variable to perform specific actions, depending on a particular frame number.

Syntax

```
FRCOUNT
```

Examples

```
IF FRCOUNT 3 eq
       (  Do this  )
ELSE
       (  Do that  }
ENDIF
```

Note: Frame numbers range from 1 to the number of frames defined by **SETLKF**. When **FRCOUNT** is equal to 0, the current page is not yet initialized, so frame number 1 has not yet started filling.

Modes

This variable is applicable in native mode and database mode.

Related commands

- FRLEFT
- GOTOFRAME
- NEWFRAME
- SETLKF

FRLEFT

FRLEFT is a built-in variable that provides the amount of vertical space available from the current print position to the bottom of the current frame on the stack. The value is expressed in current units.

Syntax

```
FRLEFT
```

Examples

```
IF FRLEFT 100 lt
{ PAGEBRK }
ENDIF
```

Modes

This variable is applicable in native mode and database mode.

Related commands

- FRCOUNT
- GOTOFRAME
- NEWFRAME
- SETLKF

GLT

GLT is a built-in variable that delivers the font size and sets the appropriate line spacing for a given GlossMark or Correlation Text font. Such a font requires using a unique point size that is part of the font definition. Use **GLT** exclusively as the size parameter of a **SETFONT** command. For more information, refer to Specialty Imaging.

Syntax

```
/glossfont_name GLT SETFONT
```

Where:

The font is first installed on the PostScript device using the regular procedure for installing a font.

Limitations

GlossText printing requires using one of the following pre-defined colors exclusively:

- GL_Black
- GL_Gray
- GL_Red
- GL_Green
- GL_Blue
- GL_Yellow
- GL_Cyan
- GL_Magenta
- GL_Maroon
- GL_lightBlue
- GL_lightGreen
- GL_Olive
- GL_Peach

Examples

This example uses a **BCALL** construct, which is recommended to isolate the GlossText sequence from the rest of the VIPP® code:

```
100 500 MOVETO
{ GL_Magenta SETTXC
  /NeueModern-GL-30 GLT SETFONT
  (Hello World) SH
} BCALL
```

Modes

This variable is applicable in all modes.

Related commands:

- SETTXC
- INDEXCOLOR

- SETFONT
- INDEXFONT

GRIDSKIP

GRIDSKIP is a built-in VIPP® boolean variable used to determine how a **PAGEBRK** occurred in line mode. **GRIDSKIP** is true when the lines-per-page, as defined by **SETGRID**, are exceeded. **GRIDSKIP** is false otherwise when a **PCC**, **SETPBRK**, or **SETSKIP** condition occurs.

GRIDSKIP can be used in line mode only in a **BEGINPAGE** or **ENDPAGE** procedure.

Examples

```
{ IF GRIDSKIP
    { (continue.frm) SETFORM }
   ELSE
   { (main.frm) SETFORM }
   ENDIF
} BEGINPAGE
```

Modes

This variable is applicable in line mode.

Related commands

- SETSKIP
- IF/ELSE/ELIF/ENDIF
- BEGINPAGE
- ENDPAGE

HCOLOR

The built-in Colorkey, **HCOLOR**, selects the current highlight color on DocuPrint NPS Highlight Color systems.

HCOLOR defaults to black on other devices. **HCOLOR** requires that you use the command with **SETTXC**, **INDEXCOLOR**, **SETGEP**, and **RPE** elements.

Syntax

```
HCOLOR
```

Modes

This built-in Colorkey is applicable in all modes.

Related commands

- FROMLINE
- INDEXCOLOR
- SETGEP
- SETTXC
- RPEKEY

HDISP

HDISP provides the displacement between the current secondary horizontal print position and those saved by **SAVEPP**. Use **HDISP** with MOVETO, MOVEH, **DRAWBx** and **DRAWPOL**.

Syntax

```
HDISP
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

- SAVEPP
- SHPOS
- SVPOS
- VDISP

HPOS

HPOS provides the current horizontal main print position.

Syntax

```
HPOS
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

- HPOS2
- VPOS
- SVPOS
- HDISP
- SAVEPP
- SHPOS
- VDISP

HPOS2

HPOS2 provides the current horizontal secondary print position.

Syntax

```
HPOS2
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

- HPOS
- VPOS
- SVPOS
- HDISP
- SAVEPP
- SHPOS
- VDISP

IHEIGHT

IHEIGHT delivers the height of an image in the current units as TIFF, JPEG or EPS.

Syntax

```
(image_name) IHEIGHT
```

Modes

This command is applicable in all modes.

Related commands

ICALL, SCALL

IWIDTH

IHEIGHT delivers the height of an image in the current units as TIFF, JPEG or EPS.

Syntax

```
(image_name) IWIDTH
```

Modes

This command is applicable in all modes.

Related commands

ICALL, SCALL

LNCOUNT

LNCOUNT is a built-in variable that provides the number of lines on the current page. Use **LNCOUNT** only inside **BEGINPAGE** or **ENDPAGE** procedures.

Syntax

```
LNCOUNT
```

Examples

```
{ IF LNCOUNT 50 gt
    { (jdt1.jdt) SETJDT }
  ELSE
    { (jdt2.jdt) SETJDT }
  ENDIF
  ...
  BEGINPAGE
```

Modes

This variable is applicable in line mode.

Related commands

BEGINPAGE, ENDPAGE

LPCOUNT

LPCOUNT is a built-in variable or an integer that provides the current logical page number. This is the number printed by the **SETPAGENUMBER** command. **LPCOUNT** can be used to capture the page number on a specific page and refer to it later.

LPCOUNT starts from 0 by default, due to the default **SETPAGENUMBER** in `xgf.def`.

To print the **LPCOUNT**, a VSUB construction is required.

Syntax

```
LPCOUNT
```

Examples

```
(Page #) 1 1 SETPAGENUMBER
IF ... { /VARpage LPCOUNT SETVAR } ENDIF
IF ... { (as mentioned on page $$VARpage., ..) VSUB 0 SHP } ENDIF
```

Note: Information provided by this variable is relative to the current page when used in **BEGINPAGE**, and relevant to the forthcoming page when used in **/P ENDPAGE**.

Modes

This variable is applicable in all modes.

Related Commands

SETPAGENUMBER

LPINDEX

LPINDEX is a built-in variable integer that provides the index of the current logical page on the current physical page. **LPINDEX** ranges from 1 to the number of entries in the Multi-Up definition.

To print an **LPINDEX** a VSUB construction is needed.

Syntax

```
LPINDEX
```

Examples

```
IF LPINDEX 1 eq { ... action on 1st logical page ... } ENDIF
```

Note: Information provided by this variable is relative to the current page when used in **BEGINPAGE**, and relevant to the forthcoming page when used in **/P ENDPAGE**.

Modes

This variable is applicable in all modes.

Related Commands

SETMULTIUP

LSP

LSP is a built-in dynamic variable that provides the current line spacing as defined by **SETGRID** or **SETLSP** to an **RPE** entry in a **FROMLINE** definition. The **LSP** variable enables interaction between **RPE** and **SETGRID**. **LSP** is often used in conjunction with **YINIT**.

Syntax

```
LSP
```

```
/LSP
```

When **SETGRID** can vary during a job, use the second syntax.

Examples

The following is an example of a layout setting, driven by **SETGRID**, which prints the first two lines using a bold font on a zebra background.

```
80 60 SETGRID
LMEDIUM 1 1 SETZEBRA
100 200 200 200 SETMARGIN
/F1 /NCR 0 INDEXFONT
/F2 /NCRB 0 INDEXFONT
2 BEGINRPE
1 FROMLINE [ 0 0 0 null YINIT LSP 0 80 /F2 BLACK ]
3 FROMLINE [ 0 0 0 null YINIT LSP 0 80 /F1 BLACK ]
ENDRPE
```

Modes

This variable is applicable in line mode and database mode.

Related commands

FROMLINE, SETGRID, YINIT

MPR

MPR is a built-in variable that delivers the font size and sets the appropriate line spacing for a given Microprint font. A Microprint font requires a unique point size that is part of the font definition. Use **MPR** exclusively as the size parameter of a **SETFONT** command. For more information, refer to Specialty Imaging.

Syntax

```
/microfont_name MPR SETFONT
```

The font is first installed on the PostScript device using the regular procedure for installing a font.

Examples

This example uses a **BCALL** construct, which is recommended to isolate the Microprinting sequence from the rest of the VIPP® code:

```
100 500 MOVETO
{ /micro_f7 MPR SETFONT
  (Hello World) SH
} BCALL
```

Limitations

Microprinting is limited to solid color CMYK with components equal to 0 or 1. Before **SETFONT** is executed with **MPR**, select the color. Any other color besides a solid color CMYK color raises an error.

Tip: Because the readability of micro-text can vary, depending on the quality of the media, it is recommended that the text is printed in a repeated fashion. You can use the new **SHP** syntax using a count field.

Modes

This variable is applicable in all modes.

Related commands

- SETTXC
- INDEXCOLOR
- SETFONT
- INDEXFONT

OTCLIP and ITCLIP

OTCLIP and **ITCLIP** are reserved, built-in, GEPkeys. **OTCLIP** and **ITCLIP** are used to define a path intended to interact with subsequent **SHP** commands. Use **OTCLIP** to define a shape that the text wraps around. Use **ITCLIP** to define a shape into which text is placed.

Use **OTCLIP** and **ITCLIP** with a command that requires a GEPkey such as **DRAWB**, **DRAWC**, **DRAWBR**, **DRAWPOL**, or **DRAWPATH**.

Several paths that are continuous or discontinuous can be defined before invoking **SHP**.

Use **ETCLIP** to clear the path. The path is cleared automatically at the beginning of each page.

Syntax

As a GEPkey:

```
OTCLIP % defines a shape to place text around
ITCLIP % defines a shape to place text inside
[ GEPkey OTCLIP ] % the path is also painted by GEPkey
[ GEPkey ITCLIP ] % the path is also painted by GEPkey
```

Examples

```
ORITL
/NHE 20 SETFONT
100 SETLSP
800 300 200 200 [ S1 OTCLIP ] DRAWB
500 270 MOVETO
(Text to be printed around the box defined using DRAWC) 800 3 SHP
```

Modes

These built-in GEPkeys are applicable in all modes.

Related commands

- DRAWB and DRAWBR
- DRAWC
- DRAWPOL
- DRAWPATH and DRAWPATHR
- ETCLIP

PAGEH

PAGEH provides the height of the current logical page.

Syntax

```
PAGEH
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

PAGEW, SHEETW, SHEETH

PAGEW

PAGEW provides the width of the current logical page.

Syntax

```
PAGEW
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

PAGEW, SHEETW, SHEETH

PDFDEVICE

PDFDEVICE is a built-in boolean variable that is true when the current imaging device is a PDF producer, and false when the current imaging device is not a PDF producer. **PDFDEVICE** is intended to allow VIPP® jobs to create conditionally paper or PDF documents, or both, with certain parts of the job.

Syntax

```
PDFDEVICE
```

Examples

The following example shows how to detect specific documents within a VIPP® job and to produce the documents as a paper or a PDF document only.

```
{ /VAR_OUTPUT 1 20 3 GETFIELD
  CASE VAR_OUTPUT {}
   (PRT) { IF PDFDEVICE { SKIPPAGE } ENDIF }          % print only
   (WEB) { IF PDFDEVICE not { SKIPPAGE } ENDIF }      % PDF only
  ENDCASE
} BEGINPAGE
```

Modes

This variable is applicable in all modes.

Related commands

- IF/ELSE/ELIF/ENDIF
- SETPCD
- GETFIELD
- SKIPPAGE
- BEGINPAGE

PDFPAGES

PDFPAGES delivers the number of pages of a PDF document.

Syntax

```
(document1.pdf) PDFPAGES
```

Examples

The following example shows how to print all except the last page of a PDF.

```
/VARpages (pdf1.pdf) PDFPAGES SETVAR
{ [ /PDFTpage RPCOUNT'-'1 ] SETPARAMS
(pdf1.pdf) CACHE SCALL
PAGEBRK
} VARpages'-'1 REPEAT
```

Modes

This variable is applicable in all modes.

Related commands

SCALL, RUNPDF

PLINES

PLINES delivers the number of lines or columns in a paragraph according to the current font and current writing mode.

Syntax

```
(string) align PLINES
(string) colwidth align PLINES
```

Where:

align is the alignment option, and can be specified as one of the following values:

0: Left

1: Right

2: Center

3: Justify

+00: Treat new line characters 0x0A as spaces

+10: Strip duplicate blanks between words

+20: Treat new line characters 0x0A as end of line

+30: Strip duplicate blanks between words and treat new line characters 0x0A as end of line

+000: Wrap on Roman words and between any Asian characters

+100: Wrap-down according to Asian rules

+200: Wrap-up according to Asian rules

+300: Wrap-up + hanging punctuation according to Asian rules

+400: Hanging punctuation according to Asian rules

+1000: Set vertical-to-horizontal processing method 0, horizontal in vertical

+2000: Set vertical-to-horizontal processing method 1, 90 degree clockwise rotation

+4000: Set conversion

+5000: Set conversion and vertical-to-horizontal method 0, horizontal in vertical

+6000: Set conversion and vertical-to-horizontal method 1, 90 degree clockwise rotation

Note: Asian language and character rules for the related encoding is previously set by the **SETCJKENCMAP** and **SETCJKRULES** commands.

The list of characters to be changed from vertical to horizontal is previously set by the **SETV2HTABLE** command. The conversion table for the related encoding is previously set by the **SETV2HCONV** command.

Default lists and tables are defined in the configuration file `xgf/src/cjk.def`.

colwidth Column width used for word wrapping. When **colwidth** is not specified, the value defined by a previous **SETCOLWIDTH** is used. When **colwidth** is specified, it overrides and replaces the value defined by a previous **SETCOLWIDTH**.

Modes

This command is applicable in all modes.

Related commands

SHP, PSIZE

PPCOUNT

PPCOUNT is a built-in variable integer that provides the current absolute physical page number, also called face print or image, from the beginning of the job. The variable integer can be printed on each page using a small font, and used later as a reference for reprint using **PAGERANGE**.

To print the **PPCOUNT**, a VSUB construction is needed.

Syntax

```
PPCOUNT
```

Examples

```
{ /NHE 6 SETFONT 80 80 MOVETO ($$PPCOUNT.) VSUB SH } SETFORM
```

Note: Information provided by this variable is relative to the current page when used in **BEGINPAGE**, and relevant to the forthcoming page when used in **/P ENDPAGE**.

Modes

This variable is applicable in all modes.

Related Commands

PAGERANGE

PREV and NEXT

The **PREV_ and NEXT_** prefixes provide built-in access to the previous or next values of any database field inside a **DBM** or table variable inside a **FOREACH** loop. When there is no previous or next value a special value, `none` is returned.

Syntax

`PREV_vname`

`NEXT_vname`

Where:

vname	is one of the following:
	The name of a database field which is available only in database mode
	The name of a variable in a VIPP® table being processed by a **FOREACH** loop which is available in all modes

Examples

```
IF CustomerID PREV_CustomerID ne
   { ... action on CustomerID change ... }
ENDIF

IF NEXT_CustomerID (**none**) eq
   { ... action on last record ... }
ENDIF
```

Related commands

STARTDBM, FOREACH

PRODUCT

PRODUCT is a built-in string variable that identifies the PostScript interpreter where VI Compose is running.

Syntax

```
PRODUCT
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

None

PSIZE

PSIZE delivers the heighth or width in current units of a paragraph, according to the current font and current writing mode. When using **PSIZE**, the undefined dimension is always equal to the selected column width.

Syntax

```
(string) align PSIZE
(string) colwidth align PSIZE
```

Where:

align is the alignment option, and can be specified as one of these values:

0: Left

1: Right

2: Center

3: Justify

+00: Treat new line characters 0x0A as spaces

+10: Strip duplicate blanks between words

+20: Treat new line characters 0x0A as end of line

+30: Strip duplicate blanks between words and treat new line characters 0x0A as end of line

+000: Wrap Roman words and between any Asian characters

+100: Wrap down according to Asian rules

+200: Wrap up according to Asian rules

+300: Wrap up and use hanging punctuation according to Asian rules

+400: Hanging punctuation according to Asian rules

+1000: Set vertical-to-horizontal processing method 0, horizontal in vertical

+2000: Set vertical-to-horizontal processing method 1, 90 degree clockwise rotation

+4000: Set conversion

+5000: Set conversion and vertical-to-horizontal method 0, horizontal in vertical

+6000: Set conversion and vertical-to-horizontal method 1, 90 degree clockwise rotation

Note: Asian language and character rules for the related encoding is previously set by the **SETCJKENCMAP** and **SETCJKRULES** commands.

The list of characters to be changed from vertical to horizontal is previously set by the **SETV2HTABLE** command. The conversion table for the related encoding is previously set by the **SETV2HCONV** command.

Default lists and tables are defined in the configuration file `xgf/src/cjk.def`.

colwidth Column width used for word wrapping. When **colwidth** is not specified, the value defined by a previous **SETCOLWIDTH** is used. When **colwidth** is specified, it overrides and replaces the value defined by a previous **SETCOLWIDTH**.

Modes

This command is applicable in all modes.

Related commands

SHP, PLINES

RPCOUNT

RPCOUNT is an built-in variable that is available only inside a **REPEAT** procedure that provides the number, starting at 1, of the current iteration of the **REPEAT** loop.

Use this variable with **IF**, **ELSE**, **ENDIF** or **CASE** to perform specific actions on specific iterations in the **REPEAT** procedure

Syntax

```
RPCOUNT
```

Examples

```
{ IF RPCOUNT 1 eq
   { do this }           % action on first iteration
  ENDIF
  ...
} 10 REPEAT
```

Modes

This variable is applicable in all modes.

Related commands

REPEAT, RPLEFT

RPEPOS

RPEPOS is a built-in variable that provides the initial or final print position, or the displacement of a specific **RPE** group defined using **FROMLINE** or **RPEKEY**. **RPEPOS** requires that you use it with **MOVERTO**, **DRAWB**, and **DRAWBR**, **DRAWBM**, or **DRAWPOL**.

To draw a box, draw a line, or to place an image relative to the last line of the **RPE** group, use **RPEPOS** with **ENDPAGE**.

For more information and for other related **RPE** commands, refer to RPE command information.

Syntax

`/rpe_def offset HV_option RPEPOS`

Where:

rpe_def	specifies the line number or **RPE** key used in the related **FROMLINE** or **RPEKEY** definition.
offset	specifies the adjustment value that can be added to, or subtracted from, the retrieved value.
HV_option	**HV_option** can be one of the following: **/HI**: The initial horizontal position **/VI**: The initial vertical position **/H**: The final horizontal position **/V**: The final vertical position **/HD**: The horizontal displacement **/VD**: The vertical displacement

Examples

This example draws a horizontal line just below the last line.

```
{ORITL
 1 0 /H RPEPOS          %  final hor. pos. of "1 FROMLINE"
 1 15 /V RPEPOS         % final ver. pos. of "1 FROMLINE"
 2000 0 S1 DRAWB
}ENDPAGE
```

This example draws a box around the /ADR group of lines.

```
{ORITL
 /ADR0 -50 /HI RPEPOS   % init. hor. pos. of "/ADR0 RPEKEY"
 /ADR0 -50 /VI RPEPOS   % init. ver. pos. of "/ADR0 RPEKEY"
 1000                   % fixed width
 /ADR0 100 /VD RPEPOS   % ver. displ. of "/ADR0 RPEKEY"
 S1 DRAWB
}ENDPAGE
```

Note: Use opaque GEPkeys carefully because GEPkeys overlap any data contained in the box.

Modes

This variable is applicable in line mode.

Related commands

- BEGINRPE
- COPYRANGE
- ENDRPE
- FROMLINE
- INDEXRPE
- SETRPEPREFIX

RPLEFT

RPLEFT is a VIPP® built-in variable that is available only inside a **REPEAT** procedure that provides the number of the remaining iterations, including the current one of the **REPEAT** loop.

To perform specific actions on specific iterations in the REPEAT procedure, use the **RPLEFT** variable with **IF**, **ELSE**, **ENDIF**, or **CASE**.

Syntax

```
RPLEFT
```

Examples

```
{ IF RPLEFT 1 eq
   { do this }          % action on last iteration
  ENDIF
  ...
} REPEAT
```

Modes

This variable is applicable in all modes.

Related commands

REPEAT, RPCOUNT

SHEETH

SHEETH provides the height of the current physical page.

Syntax

```
SHEETH
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

SHEETW, PAGEW, PAGEH

SHEETW

SHEETW provides the width of the current physical page.

Syntax

```
SHEETW
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

SHEETH, PAGEW, PAGEH

SHPOS

The **SHPOS** built-in dynamic variable provides the secondary horizontal print position saved by **SAVEPP**. Use **SHPOS** with **MOVETO**, **MOVEH**, **DRAWB**, **DRAWBR**, **DRAWBM**, **DRAWBRM**, and **DRAWPOL**.

Syntax

```
SHPOS
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

- HDISP
- SAVEPP
- SVPOS
- VDISP

SLENGTH

SLENGTH delivers the length or number of characters in a string. The current font determines how characters are delimited and counted, especially for multiple-byte strings.

Syntax

```
(string) SLENGTH
```

Modes

This command is applicable in all modes.

Related commands

SSIZE

SSIZE

SSIZE delivers the width or height in the current units of a string, according to the current font and current writing mode.

Syntax

```
(string) SSIZE
```

Modes

This command is applicable in all modes.

Related commands

SLENGTH

SVPOS

SVPOS provides the vertical print position saved by **SAVEPP**. Use **SVPOS** with **MOVETO**, **DRAWB**, **DRAWBR**, **DRAWBM**, **DRAWBRM**, and **DRAWPOL**.

Syntax

```
SVPOS
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

- HDISP
- SAVEPP
- SHPOS
- VDISP

T_AMPM

T_AMPM is used in conjunction with the **GETDATE** command to set the appropriate time for the application. This variable is used to set 12-hour clocks to an AM or PM designation.

Syntax

```
T_AMPM
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

T_HH

T_HH is used in conjunction with the **GETDATE** command to set the appropriate time or hours for the application. This variable is used to set 24-hour clocks.

Syntax

```
T_HH
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

T_HH2

T_HH2 is used in conjunction with the **GETDATE** command to set the appropriate time or hours for the application. This variable is used to set 12-hour clocks.

Syntax

```
T_HH2
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

T_MM

T_MM is used in conjunction with the **GETDATE** command to set the appropriate time or minutes, 00–59, for the application.

Syntax

```
T_MM
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

T_SS

T_SS is used in conjunction with the **GETDATE** command to set the appropriate time in seconds, 00–59, for the application.

Syntax

```
T_SS
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

T_TZN

T_TZN is used in conjunction with the **GETDATE** command to set the appropriate time zone, such as PST or PDT for the application.

Syntax

```
T_TZN
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

GETDATE

TLENGTH

TLENGTH delivers the length or number of items in a table.

Syntax

```
VARTable_name TLENGTH
```

Modes

This variable is applicable in all modes.

Related commands

- SETVAR
- ADD
- GETITEM
- FOREACH

TLGRID

TLGRID enables the printing of a VIPP® job sample on top of a unit grid with a top-left origin. The grid can be useful for placing data in **RPE** definitions.

This feature helps locate print positions on the page when coding a VIPP® job such as a form, **RPE**, or native mode. The unit used is the current unit set by **SETUNIT**.

The origin 0,0 of the grid is related to the margins. In most cases, setting the margins to 0 is advised.

Syntax

```
(TLGRID) SETFORM
```

Examples

```
0 0 0 0 SETMARGIN

(TLGRID) SETFORM
```

Tip: To print the grid on top of other forms, increment **SETMAXFORM**. On Highlight Color systems, the grid is printed using the current highlight color.

Modes

This built-in form is applicable in all modes.

Related commands

- BLGRID
- SETFORM
- SETMARGIN
- SETMAXFORM
- SETUNIT

TPATH

TPATH is a reserved, built-in, GEPkey. Use **TPATH** to define a path along which **SHPATH** is used to print text. **TPATH** is used with a command that requires a GEPkey, such as **DRAWB**, **DRAWC**, **DRAWBR**, **DRAWPOL**, or **DRAWPATH**.

Several paths, like continuous or discontinuous, can be defined before invoking **SHPATH**.

To clear the path, use **ETCLIP**. The path is cleared automatically at the beginning of each page.

Syntax

As a GEPkey:

```
TPATH                 % defines a transparent path
[ GEPkey TPATH ]      % defines a path also painted by GEPkey
```

Examples

```
[ 200 200 800 1200 1400 200 ] [S1 TPATH] DRAWPOL
(Text to be printed along the triangle defined by DRAWPOL) 0 0 0 SHPATH
```

Modes

This built-in GEPkey is applicable in all modes.

Related commands

- DRAWB and DRAWBR
- DRAWC
- DRAWPOL
- DRAWPATH and DRAWPATHR
- ETCLIP

UV2L for Two-Layer UV Effect

UV2L is a mask option used in the **SETTXC** to create a second layer UV effect on top of a regular UV effect.

Examples

```
103 3342 1116 149 UV_GOLDENROD2 DRAWB  % draw UV box
/NHEB 24 SETFONT                       % select font
659 3213 MOVETO
(Hello World) SHc                      % print UV text
[YELLOW UV2L] SETTXC                   % select color and mask for 2-layer UV effect
268 3242 MOVETO
(Goodbye World) SH                     % print UV 2-layer text
```

To achieve the expected result, select the color for the second layer in accordance with the UV ink. To select a combination of UV color and second layer color of cyan, magenta, or yellow, refer to pages 7–9 of the sample file `xgf/demo/VIS_UVSamples.nm`.

VARDataFileName

VARDataFileName is a built-in variable that holds the name of the submission file. When that name is not available, the value defaults to unknown. Only VI Design Pro, VI Explorer, and VI eCompose provide the submission file name.

Syntax

```
(Data File Name: [=VARDataFileName=]) VSUB SH
```

VDISP

VDISP provides the displacement between the current vertical print position and those print positions that are saved by **SAVEPP**. Use **VDISP** with **MOVETO**, **DRAWB**, **DRAWBR**, **DRAWBM**, **DRAWBRM**, and **DRAWPOL**. To draw variable boxes, **VDISP** is often used with **SAVEPP**, **SHPOS**, and **SVPOS**.

Syntax

```
VDISP
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

- HDISP
- SAVEPP
- SHPOS
- SVPOS

VPOS

VPOS provides the current vertical print position.

Syntax

```
VPOS
```

Modes

This built-in dynamic variable is applicable in all modes.

Related commands

- HPOS
- HPOS2
- SVPOS
- HDISP
- SAVEPP
- SHPOS
- VDISP

XGFVER

XGFVER provides a string that shows the VI Compose release number. **XGFVER** requires that you use it with an **SHx** command.

Syntax

```
XGFVER
```

Examples

This example prints the current VI Compose level that is installed on the device.

```
XGFVER SHL
```

Modes

This built-in string is applicable in all modes.

Related commands

None

XMLATN

XMLATN provides the XML attribute name and is usually coupled with **XMLATV**.

Syntax

Refer to XMLATL.

Modes

This built-in variable is applicable in XML mode.

Related commands

- BTA
- BTS
- ETA
- ETS

XMLATV

XMLATV provides the XML attribute value and is usually coupled with **XMLATN**.

Syntax

Refer to XMLATL.

Modes

This built-in variable is applicable in XML mode.

Related commands

- BTA
- BTS
- ETA
- ETS

XMLDTH

XMLDTH requires that you place the command inside a **BTA/ETA** or **BTS/ETS** command pair. **XMLDTH** provides the depth of the current node.

Syntax

```
XMLDTH
```

Modes

This built-in variable is applicable in XML mode.

Related commands

- BTA
- BTS
- ETA
- ETS

XMLPAR

XMLPAR requires that you place the command inside a **BTA/ETA** or **BTS/ETS** command pair. **XMLPAR** provides the name of the parent node of the current node.

Syntax

```
XMLPAR
```

Modes

This built-in variable is applicable in XML mode.

Related commands

- BTA
- BTS
- ETA
- ETS

XMLPATH

XMLPATH requires that you place the command inside a **BTA/ETA** or **BTS/ETS** command pair. **XMLPATH** provides the VXVpath of the current node.

Syntax

```
XMLPATH
```

Modes

This built-in variable is applicable in XML mode.

Related commands

- BTA
- BTS
- ETA
- ETS

XMLTAG

To use **XMLTAG**, place the command inside a **BTA/ETA** or **BTS/ETS** command pair. **XMLTAG** provides the name of the current node.

Syntax

```
XMLTAG
```

Modes

This built-in variable is applicable in XML mode.

Related commands

- BTA
- BTS
- ETA
- ETS

XMLVAL

To use **XMLDTH**, place the command inside a **BTA/ETA** or **BTS/ETS** command pair. **XMLDTH** provides the depth of the current node.

Syntax

```
XMLVAL
```

Modes

This built-in variable is applicable in XML mode.

Related commands

- BTA
- BTS
- ETA
- ETS

YINIT

YINIT is a built-in dynamic variable that provides the current vertical position as defined by **SETGRID** to an **RPE** entry in a **FROMLINE** definition. The **YINIT** variable enables interaction between **RPE** and **SETGRID**. **YINIT** is often used in conjunction with **LSP**.

Syntax

```
YINIT
```

```
/YINIT
```

When **SETGRID** can vary during the job, use the second syntax.

Examples

The following is an example of a layout setting, driven by **SETGRID**, where the first two lines are printed using a bold font on a zebra background.

```
80 60 SETGRID
LMEDIUM 1 1 SETZEBRA
100 200 200 200 SETMARGIN
/F1 /NCR 0 INDEXFONT
/F2 /NCRB 0 INDEXFONT
2 BEGINRPE
1 FROMLINE [ 0 0 0 null YINIT LSP 0 80 /F2 BLACK ]
3 FROMLINE [ 0 0 0 null YINIT LSP 0 80 /F1 BLACK ]
ENDRPE
```

Modes

This variable is applicable in line mode.

Related commands

FROMLINE, SETGRID, LSP

ZSPAGE

ZSPAGE is a built-in variable used with database **ZSORT** and in the **DBM** only, to produce the document page number, starting from 1. For backward compatibility, using the **BACK** variable for a two-page duplex document is still possible.

Refer to the `xgf/demo` folder for the file, `ZSORT_Example.vpc`, which contains information about using **ZSPAGE** with **ZSORT**.

ZSRECNUM

ZSRECNUM is a built-in variable used with database **ZSORT** and in the **DBM** only, to produce record numbers in data file order. This is the current record number, incremented from 0.

Refer to the `xgf/demo` folder for the file, `ZSORT_Example.vpc`, which contains information about using **ZSRECNUM** with **ZSORT**.

ZSREPCNT

ZSREPCNT is a built-in variable used with database **ZSORT** and in the **DBM** only, to produce an ongoing repeat count, starting from 0.

Refer to the `xgf/demo` folder for the file, `ZSORT_Example.vpc`, which contains information about using **ZSREPCNT** with **ZSORT**.

ZSREPIDX

ZSREPIDX is a built-in variable used with database **ZSORT** and in the **DBM** only, to produce a record repeat index. **ZSREPIDX** increments for each instance but resets to 0 at the start of the next record.

Refer to the `xgf/demo` folder for the file, `ZSORT_Example.vpc`, which contains information about using **ZSREPIDX** with **ZSORT**.

6

Parameters

This chapter contains:

A VIPP® parameter is a keyword that represents an internal value that affects the behavior of one or several commands or transform functions. It can only be set using the SETPARAMS command.

The option parameter of the DRAWBAR, DRAWCRV, DRAWPIE, and SETPARAMS commands is available in three forms:

- /code
- [/param1 value1 /param2 value2..../paramN valueN]
- [/code /param1 value1 /param2 value2..../paramN valueN]

Where:

/code	is a numeric value computed by adding all codes related to the parameters required. Parameter values enabled by the code are indicated in the following table (print `xgf/demo/samddg.ps` for a complete sample of codes).
/paramX valueX	are key/value pairs from the following table used to change the default value of a given parameter.

All parameters set by either DRAWBAR, DRAWCRV or DRAWPIE can temporarily override the default value set by SETPARAMS and only apply to that command. The default values is restored for subsequent commands.

Examples

Option parameters, which result in equivalent outputs:

```
/67
[/66 /3D true ][/3D true /SliceSepWidth 0 /PrintValue true ]
```

Parameter Categories

Parameters are used with several different commands, and can be grouped into these categories:

- Chart parameters, which apply to **DRAWPIE**, **DRAWBAR**, and **DRAWCRV**
- Format parameters, which apply to the **FORMAT** command and Format parameter
- Duplex parameters, which apply to **DUPLEX_on** and **DUPLEX_off**
- Booklet parameters, which apply to **STARTBOOKLET** and **ENDBOOKLET**
- Job reprint parameters, which apply to **PAGERANGE** and **BOOKLETRANGE**
- Image rendition parameters, which apply to **ICALL** and **CACHE SCALL**
- Media selection parameters, which apply to **SETMEDIA**
- **LCDS** migration parameters, which apply to **SETFORM** and **SETBFORM**
- **PIF** parameters, which apply to **SETPIF**, **INDEXPIF**, and **BOOKMARK**
- Caching parameters, which apply to **CACHE**, **PRECACHE**, and **FSHOW**
- Optical Mark Reading (OMR) parameters, which apply to **FILLOMR**.
- Date and time parameters, which apply to **GETDATE**.
- Layout parameters, which apply to logical page layout commands.
- Page composition parameters, which apply to page formatting commands.
- Barcode parameters, which apply to barcode commands.

Parameters that do not fit easily into other defined categories are listed as Miscellaneous.

Parameter Descriptions

VIPP® parameter descriptions include this set of information:

- Parameter name
- Category
- Default value
- Unit or format in which the parameter is entered
- Code
- Code value
- Description

When any of part of the list is omitted from a specific Parameter description, you can infer that there is no value to be entered.

/3D

Category: Chart
Default value: false
Unit or Format: true/false
Code: +1/
Code value: true
Description: 3D effect

/3DAngle

Category

Chart

Default value

0.3

Unit or Format

Real integer between 0 and 2

Description

- 3D vision angle
- **0 to 1**: Show the right side of bars
- **1 to 2**: Show the left side of bars

/3DThickness

Category: Chart

Default value: 0.35

Unit or Format:
- % of Radius
- % of bar width
- % of Width/5

Description: Thickness (depth) when /3D is true. Refer to /KeepRatio.

/Across

Category: Layout

Default value: 1

Unit or Format: integer

Description: The number of logical pages across the printed sheet in a Multi-Up application.

/Align

Category: Table

Default value: 0

Unit or Format: integer

Description: align attribute; same as SHP

/AlignChar

Category: Page Composition

Default value: Value of /DecimalPoint

Unit or Format: ASCII code

Description: Defines alternate align character for SHMF align=4.

Possible values: Any ASCII code, for example, 0–255

/AmPm

Category: Date

Unit or Format: An array of two strings

Description: AM and PM designations

Examples

```
[ (a.m.) (p.m.) ]
```

/BarSpace

Category: Chart

Default value: 0.4

Unit or Format: % of bar width

Code: +32/

Code value: 0

Description: Spacing between bars as a percentage of the bar width.

Possible values: **0**: No space between bars.

1: The space between bars is equal to the width of the bars.

10: The space between bars is 10 times the width of the bars.

/BarSpace2

Category: Chart

Default value: 0

Unit or Format: integer

Description: Sets the spacing between a cluster of bars.

Possible values: Same as /BarSpace

/BGColor

Category: Chart

Default value: WHITE

Unit or Format: Colorkey or null

Code: +16

Code value:	.99
	.95
	.70
Description:	Color of background for bar and curve charts. For transparencies, use null instead of a Colorkey.

/BGLineColor

Category:	Chart
Default value:	WHITE
Unit or Format:	Colorkey
Description:	Color for horizontal background scale lines. Possible values: Any plain Colorkey.

/BookletMismatch

Category:	Booklet
Default value:	0
Unit or Format:	integer
Description:	Sets the action when the number of booklet pages do not match the PagesPerBooklet parameter.
Possible values:	**0**: Ignore mismatch
	1: Add pages until the booklet pages match.
	2: Abort and send a VI Compose error message.
	3: Add pages with the current form until the number of pages match.
	4: Add blank pages up to PagesPerBooklet when the number of pages in the booklet is less than PagesPerBooklet.
	5: Add pages with the current form up to PagesPerBooklet when the number of pages in the booklet is less than PagesPerBooklet.

/BottomBleed

Category:	Layout
Default value:	0
Unit or Format:	integer
Description:	The value for the amount of bleed allowed at the bottom margin of a logical page in a Multi-Up application.

/BurstList

Category:	Chart
Unit or Format:	[(label1) (label2) ...]
Description:	List of labels to be burst when /SliceBurst is not zero.
Possible values:	List of any label present in the label/value list.

/CacheICALL

Category:	Caching
Default value:	false
Unit or Format:	boolean
Possible values:	**false**: No caching occurs when you use **ICALL**. **true**: Enables caching transparently when using **ICALL**.

Examples

```
[ /CacheICALL true ] SETPARAMS  % Enable Caching
                                % on ICALL.
```

/Caching

Category:	Caching
Default value:	12
Unit or Format:	integer
Description:	Caching options are: **0** no caching (CACHE can be used but no caching will occur). **Adobe caching (will occur according to Adobe specifications).** **1** caching through CACHE and PRECACHE. **2** caching through CACHE, PRECACHE and FSHOW. **VIPP® caching (will occur according to VIPP® specifications).** **11** caching through CACHE and PRECACHE. **12** caching through CACHE, PRECACHE and FSHOW (default).

Note: When VIPP® caching is not implemented in the PostScript interpreter on which VIPP® is running, caching defaults automatically to Adobe caching.

/CellImage

Category: Table

Unit or Format: (image file name)

Description: image to be placed in the cell

/CellFill

Category: Table

Unit or Format: Color name

Description: Defines the color used to fill the cell.

/CellStroke

Category: Table

Unit or Format: [top bottom left right] (array or GEPkeys)

Description: Defines the GEPkeys to stroke the borders of the cell.

/CellText

Category: Table

Unit or Format: (text)

Description: Text to be entered into the cell.

/ChartDir

Category: Chart

Default value: 0

Unit or Format: integer

Description: Chart direction for **DRAWBAR** and **DRAWCRV**.

Possible values:

0: Bottom-up

1: Left-to-right

2: Top-down

3: Right-to-left

/ChartOrder

Category: Chart

Default value: 0

Unit or Format: integer

Description: **DRAWBAR** and **DRAWCRV** only.

Possible values: **0**: Print items from left to right or top to bottom.

1: Print items from right to left or bottom to top.

/CheckLabelOverlap

Category: Chart

Default value: true

Unit or Format: boolean

Description: Enables and disables label overlap control for pie charts.

Possible values: **true**: Prevents label overlap.

false: Does not prevent label overlap.

/ChkResources

Category: Resource checking

Default value: 0

Unit or Format: integer

Description: This parameter is supported for backward compatibility. To gather resource information, refer to the demographics feature described in the *FreeFlow VI Compose User Guide*.

/CJKunitcount

Category: Miscellaneous

Default value: false

Unit or Format: true or false

Description:	Set multiple-byte character count method, for example for Chinese, Japanese, or Korean.
Possible values:	**false**: Each multiple-byte character is one unit. **true**: Applies only to **SETRCD**, **SETPCD**, **GETFIELD**, and **RPE**. A single byte character is one unit. A multiple-byte character is two units.

/ClearSubst

Category:	Image rendition
Default value:	0
Unit or Format:	integer
Description:	Set rendering of Clear Dry Ink.
Possible values:	The following values are supported: **0** best fit behavior (default): If Clear is present, apply. On VDP/VIE/VDEP, desktop printer substitute text pattern. On production printer (FFPS, ...) w/o Clear, ignore. On FFPS/Normalizer, preserve or ignore (depending if Clear is present or not). On VIeC, ignore **1** same as 0 but preserve on VIeC (PDF intended for printing) **2** ignore on all devices except when Clear is present **3** substitute text pattern on all devices except when Clear is present **4** preserve on all devices **5** ignore on all devices even when Clear is present **6** substitute text pattern on all devices even when Clear is present

Note: VIPP® Xerox Specialty Inks support assumes that the Clear colorant is available as a spot color named Clear.

Examples

```
[ /ClearSubst 3 ] SETPARAMS    % substitute text pattern
                               % (for proof printing)
```

/ClusterMode

Category: Chart

Default value: 0

Unit or Format: integer

Description: Sets the presentation mode for a cluster of bars.

Possible values: **0**: Stacked bars

1: Side-by-side bars

/ColorCycle

Category: Chart

Default value: 0

Unit or Format: integer

Description: Sets the color cycle mode.

Possible values: **0**: Automatic; usually the most appropriate value.

1: Cycle an item, cluster

2: Cycle an entire chart

/ColorTable

Category: Chart

Default value: XLGREEN LRED LGREEN RED MBLUE LMGREEN XDBLUE MRED DGREEN LBLUE

Unit or Format: [Colorkeys...]

Description: Colorkeys list used in a cyclical manner.

For transparencies, use null instead of a Colorkey.

/DaylightSaving

Category: Date

Unit or Format: array 6-integer arrays

Description: Start and end times for Daylight Saving Time.

Examples

```
[ year +-hrs startday starttime endday endtime ]
```

/DaysLong

Category:	Date
Unit or Format:	An array of seven strings.
Description:	List of day names in long format, starting with Sunday.

/DaysShort

Category:	Date
Unit or Format:	An array of seven strings.
Description:	List of day names in short format, starting with Sunday.

/DecimalPoint

Category:	Format
Default value:	. (US) , (A4)
Unit or Format:	ASCII code
Description:	Decimal delimiter in numeric data.

/DefaultDate

Category:	Date
Default value:	[2003 1 1 00 00 0 0]
Unit or Format:	array of 7 integers
Description:	Date used when no file system is available.

Examples

```
[ year month day hrs mns sec daylightsaving(0/1)]
```

/DefinedDate

Category: Date

Default value: array of 6 integers

Unit or Format: When present, this date overrides the system date.

Description: `[ year month day hrs mns sec ]`

/DotsPerModule

Category: Barcode

Default value: 3

Unit or Format: integer

Description: The number of dots per bar or space in the barcode.

/Down

Category: Layout

Default value: 1

Unit or Format: integer

Description: The number of logical pages down the printed sheet in a Multi-Up application.

/DrawMode

Category: Chart

Default value: 0

Unit or Format: integer

Description: Supports stroke or fill of the radar area (DDG).

Note: To allow a different option for each stacked area, use the second syntax.

Examples

```
/DrawMode opt
```

or

```
/DrawMode [ opt1 opt2 ... optn ]
```

Where:

`optx` is one of:

- 0 stroke the radar area (default)
- 1 fill the radar area

/EmptyJobReport

Category:	Miscellaneous
Default value:	0
Unit or Format:	integer
Description:	This parameter is used to indicate if an empty job, for example a job that produces no pages at all is valid or not. 0 - unknown 1 - a job producing no pages is valid 2 - a job producing no pages is invalid

Note: This parameter is intended mainly to help applications that produce PDF files, such as VIeC or FFCore, to return appropriate messages in the user interface.

Examples

`[ /EmptyJobReport 1 ] SETPARAMS%` an empty job is valid.

/ELevel

Category:	Barcode
Default value:	23
Unit or Format:	integer between 0 and 99
Description:	Error correction level

/ExtraSpace

Category:	Chart
Default value:	1
Unit or Format:	% of font size
Description:	Top and bottom extra space when /FitSpace > 0.

/FDecimalPoint

Category: Format

Default value: . (US)

, (A4)

Unit or Format: ASCII code

Description: Decimal delimiter in format.

/FDigit

Category: Format

Default value: #

Unit or Format: ASCII code

Description: Placeholder for digit in format.

/FillOrder

Category: Layout

Description: The order for filling the logical pages.

Possible values: /RD (right-down) left to right then top to bottom

/LD (left-down) right to left then top to bottom

/RU (right-up) left to right then bottom to top

/LU (left-up) right to left then bottom to top

/DR (down-right) top to bottom then left to right

/UR (up-right) bottom to top then left to right

/DL (down-left) top to bottom then right to left

/UL (up-left) bottom to top then right to left

/FitSpace

Category: Chart

Default value: 1

Unit or Format: integer

Description: Fit RPE space when ddg_index is used

Possible values: **0** fixed size

+1 fit RPE space

+2 cancel 3D when fitting radius is less than maximum

+4 force 3D when fitting radius is greater than maximum

+8 adjust 3D thickness for a better fit

/FLZDigit

Category: Format

Default value: @

Unit or Format: ASCII code

Description: Placeholder for a digit in a format that is replaced by a space when the digit is a leading zero.

/FNSign

Category: Format

Default value: -

Unit or Format: ASCII code

Description: Negative sign in format.

/Format

Category: Chart

Default value: string

Description: When /PrintValue is set to true, use this parameter to specify the printing format for the label values and scale. The string associated with the /Format parameter follows the rules described in the **FORMAT** command. Text attributes switches, for example font, color and so on are allowed inside the format string.

/FormatV

Category: Chart

Unit or Format: string

Description: When /PrintValue is set to true, use this parameter to specify the format for values. The string associated with the /FormatV parameter follows the rules described in the **FORMAT** command. The string can include font and color switches which is set by INDEXFONT and INDEXCOLOR.

/FormatVI

Category: Chart

Unit or Format: string

Description: When /PrintValue is set to true, use this parameter to specify the format for individual values. The string associated with the /FormatVI parameter follows the rules described in the **FORMAT** command. The string can include font and color switches which is set by INDEXFONT and INDEXCOLOR.

/FormatPC

Category: Chart

Unit or Format: string

Description: When /PrintValue is set to true, use this parameter to specify the printing format for percentages. The string associated with the /Format parameter follows the rules described in the **FORMAT** command.

/FPSign

Category: Format

Default value: +

Unit or Format: ASCII code

Description: Positive sign in format.

/FPunctuation

Category: Format

Default value: , (US)

. (A4)

Unit or Format: ASCII code

Description: Thousands delimiter in format.

/HalfPie

Category: Chart

Default value: 0

Unit or Format: integer

Code: +512

Code value: 1

Code: +1024

Code value: -1

Possible values: **0** Full pie

1 Top half

-1 Bottom half

/Height

Category: Table

Default value: 0

Unit or Format: real

Description: Minimum height of a table cell. Default is current units.

/HGutter

Category: Layout

Default value: 0

Unit or Format: integer

Description: The value for the horizontal gutter between logical pages in a Multi-Up application.

/ImageDefRes

Category: Image rendition

Default value: 300

Unit or Format: integer

Description: Set default image resolution in dots per inch when not present in the TIFF and JPEG parameters.

/IMPmode:

Category: Imposition

Default value: 0

Unit or Format: integer

Description: Set order for **BEGINIMP** and **ENDIMP**

Possible values: **0**: 1-to-N order

1: N-to-1 order

/Interpolate

Category: Image rendition

Default value: false

Unit or Format: true / false

Description: Request image interpolation and enhanced quality on color images.

Note: This parameter can impact job performance. To minimize job impact, use the parameter with the **CACHE** command.

/KeepRatio

Category: Chart

Default value: true

Unit or Format: true / false

Code: +256

Code value: false

Possible values: **true** 3D Thickness is based on bar width

false 3D Thickness is based on Width/5

/LabelColw

Category: Chart

Default value: 0.2

Unit or Format: Real

Description: Column width in percentage of the chart width for label wrapping of horizontal charts (ChartDir=1/3).

/LabelDashColor

Category: Chart

Default value: Black

Unit or Format: Colorkey

Description: Color of label dashes (WHITE=no dash).

/LabelDashWidth

Category: Chart

Default value: 0.5

Unit or Format: points

Description: Line width of label dashes.

/LabelOffset

Category: Chart

Default value: 0.1

Unit or Format: % of Radius

Description: Offset of labels from the pie (when SpotSize=0).

/LayoutMarks

Category: Layout

Default value: 0

Unit or Format: integer

Description: Choose one of the following options:

Possible values:

0 no marks (default)

1 crop marks

2 bleed marks

+ option to print the marks on front, back or both

+0 print marks on front page only

+10 print marks on back page only

+20 print marks on front and back pages

To print crop marks on back only, enter: `/LayoutMarks 11`

+100 disable inner crop marks

+200 disable inner bleed marks

+300 disable inner crop and bleed marks

To print crop marks on back only, with inner marks disabled, enter: `/LayoutMarks 111`

/LCDSmode

Category: LCDS migration

Default value: false

Unit or Format: true / false

Description: When set to true SETFORM and SETBFORM apply all forms to the physical pages instead of the logical pages. Use this setting to facilitate migration from LCDS to VIPP®.

Note: The **PROCESSDJDE** command sets this value to true automatically.

/LeftBleed

Category: Layout

Default value: 0

Unit or Format: integer

Description: The value for the amount of bleed allowed at the left margin of a logical page in a Multi-Up application.

/LineDash

Category: Chart

Unit or Format: [On1 Off1 On2 Off2 ...]

Description: Defines the line dash for **DRAWCRV**. Size is in points of consecutive solid and blank segments.

Possible values: array of reals.

/LocalToUTF8

Category: Miscellaneous

Default value: 0

Unit or Format: integer

Description: Database mode only. Provides automatic conversion from local encoding to UTF8. This option is intended mainly for direct submission of data files with local encoding that reference a project designed using VI Design Express.

Note: Using this parameter removes the need to convert the data file prior to submission.

Possible values:

0 no conversion (default)

1 ISO-8859-1 or Windows-1252 (Western European)

2 ISO-8859-2 (Central European)

3 Windows-1259 (Latin-2)

4 ISO-889-9 (Turkish)

5 Windows-1251 (Cyrillic)

6 Windows-1258 (Vietnamese)

7 Windows-CP874/TIS-6

8 Windows-CP866 (Cyrillic)

9 IDSO-8858-15 (Latin-9)

10 Mac OS Roman

11 Windows-1256 (Arabic)

12 Windows-1255 (Hebrew)

13 UTF16

/Margins

Category: Table

Default value: 0[0 0 0 0]

Unit or Format: [top bottom left right] (array or reals)

Description: value for table cell margins in current units

/MarkLength

Category: Layout

Default value: 0

Unit or Format: integer

Description: The length of layout marks in a Multi-Up application.

/MarkOffset

Category: Layout

Default value: 0

Unit or Format: integer

Description: The offset for layout marks from the corner of the page.

/MarkWidth

Category: Layout

Default value: 0

Unit or Format: integer

Description: The width of the layout marks in a Multi-Up application.

/MaxHeight

Category: Table

Unit or Format: real

Description: Maximum height of a table cell. Default is current units.

/MaxVal

Category:	Chart
Unit or Format:	integer
Description:	Maximum value of the chart scale

/MediaSubst

Category	Media selection
Unit or Format	substitution list
Description	Defined device-specific actions for **SETMEDIA** requirements. Use on devices that do not support media selection through **SETPAGEDEVICE**

Note: For more information, refer to Media Support.

/MergeValue

Category:	Chart
Default value:	false
Unit or Format:	true / false
Description:	Combine values with equal labels.

/MinDim

Category:	Barcode
Default value:	1
Unit or Format:	integer
Description:	Minimum value of a barcode dimensions

/MinVal

Category:	Chart
Unit or Format:	integer
Description:	Minimum value of the chart scale

/MixPlexCount

Category: Duplex

Default value: 0

Unit or Format: integer

Description: Number of pages to delay simplex mode, refer to DUPLEX_off

/MonthsLong

Category: Date

Unit or Format: array of 12 strings

Description: List of months names (long).

/MonthsShort

Category: Date

Unit or Format: array of 12 strings

Description: List of months names (short).

/MUPduplex

Category: Multi-Up

Default value: 0

Unit or Format: integer

Possible values: **0** (default) logical page positions are identical on the front and on the back

1 logical page positions on the back side are computed so that they physically face their counterparts on the front. Logical pages can fit entirely on the physical page, either through explicit specification of the logical page sizes or by scaling down.

/NSign

Category: Format

Default value: -

Unit or Format: ASCII code

Description: Negative sign in numeric data.

/OffsetValue

Category:	Chart
Default value:	0
Unit or Format:	real or [real1 real2]
Description:	DRAWBAR only. Sets a vertical or horizontal (depending on ChartDir) positive or negative offset based on the font size (1 = font size). When coding a stacked bar chart an array of two reals can be coded. The second real allows individual values of each stacked bar to be printed (using different offsets) in addition to the total value. In addition, two specific format parameters can be used to set formats for the individual at total values: /FORMATV formats the total values /FORMATVI formats the individual values Format strings can include font and color switches to apply specific font and color attributes to these values.
Recommended values:	**Vertical**: 1.4 **Horizontal**: 0.8

Examples

```
/I /NHEB 11 INDEXFONT
/V /NHEB 14 INDEXFONT
/L 60 INDEXLSP
/W WHITE INDEXCOLOR
[ labx valx ....]
width height
[ /OffsetValue          [-2 1.3]
  /FORMATV              (//V//LTotal\n$@@@@#.00)
  /FORMATVI             (//I//W$@@@@#)
]DRAWBAR
```

/OMRconfig

Category:	OMR
Unit or Format:	[width height spacing (config)]
Description:	Defines the configuration of the OMR code for OMRINIT/OMRSHOW. For more information, refer to OMRINIT.

/OMRDir

Category: OMR

Default value: /V

Unit or Format: key

Description: The direction of the OMRMap string characters in the OMR grid.

Possible values:

/H OMR map string characters are shown horizontally (and form each row of) the OMR grid

/V OMR map string characters are shown vertically (and form each row of) the OMR grid. (default)

/OMRHdisp

Category: OMR

Default value: 6

Unit or Format: CPI

Description: The number of columns per inch (CPI) in the OMR grid.

/OMRHskip

Category: OMR

Default value: 0

Unit or Format: integer

Description: Indicates the number of columns skipped.

/OMRMap

Category: OMR

Default value: (ABCDEFGHIJKLMNOPQRSTUVWXYZ)

Unit or Format: map string

Description: Indicates which characters make up the OMR grid and in which order.

Note: Every character in the response string can be a character in the map string.

/OMRMode

Category: OMR

Default value: 12

Unit or Format: points

Possible values: **0** Character mapping (default)

1 Binary decimal mapping

2 Binary litho mapping

3 P-slugging (P,7,4,2,1)

Where:

Option 0 slugs each character of the response string that can be made up of characters from OMRMap.

Option 1 slugs each digit of a numeric string into appropriate binary bits 1, 2,4 and 8. OMRMap can be set to (1248) or (8421).

Option 2 slugs a numeric string (up to 1,073,741,823) into appropriate binary bits 1, 2, 4, 8, 16, 32, etc. up to 536,870,912. OMRMap is irrelevant with this option.

Option 3 slugs each digit of a numeric string into P,7,4,2,1 slugging scheme.

Examples

```
(4523456) [ /OMRMode 1 /OMRMap (8421) ] FILLOMR
(234523456) [ /OMRMode 2 ] FILLOMR
```

/OMRSlugChar

Category: OMR

Default value: (D)

Unit or Format: string

Description: The shape of the OMR grid bubbles.

Possible values: **(A)** square

(B) vertical rectangle

(C) horizontal rectangle

(D) circle (default)

(E) vertical oval

(F) vertical condensed oval

(G) horizontal oval

(H) horizontal condensed oval

Note: The letters A–H in this example are not related to the response or map strings. To choose a shape that is defined in the /XOMR font, use letters A–H.

/OMRSlugFont

Category: OMR

Default value: /XOMR

Unit or Format: /Fontname

Description: The OMR font used to fill in the OMR grid bubbles.

/OMRSlugSize

Category: OMR

Default value: 12

Unit or Format: points

Description: The size in points of the OMR font used to fill in the OMR grid bubbles.

/OMRVdisp

Category: OMR

Default value: 6

Unit or Format: LPI

Description: The number of lines/rows per inch (LPI) in the OMR grid.

/OMRVskip

Category: OMR

Default value: 0

Unit or Format: integer

Description: Indicates the number of rows skipped.

/OMRWriteResp

Category: OMR

Default value: false

Unit or Format: true/false

Description: Indicates how response boxes are filled

Possible values: **true** the response boxes must be filled in with the response string

false no response box (default)

/OriLine

Category: Chart

Default value: 10

Unit or Format: integer or string for backward compatibility

Description: Sets the origin and position of the line for DRAWCRV and DRAWPAR.

Possible values: **00** line goes from the left of the first item to the left of each item

10 line goes from the middle of the 1st item to the middle of each item

20 line goes from the right of the 1st item to the right of each item

01 line goes from chart origin (0,0) to the left of each item

11 line goes from chart origin (0,0) to the middle of each item

21 line goes from chart origin (0,0) to the right of each item

false same as 10 (for backward compatibility)

true same as 11 (for backward compatibility)

/PageClip

Category: Layout

Default value: false

Unit or Format: true/false

Description: enable page clipping

/PageHeight

Category: Layout

Unit or Format: integer

Description: The height of the logical page in a Multi-Up application.

/PageRange

Category: Job reprint

Default value: 0 (1 on PDF devices)

Unit or Format: integer

Description: /Pagerange is used to define the PAGERANGE behavior after the last page in the range has been printed. This command may be placed in the xgf.def file, in the JDT, at the beginning of the data file along with the PAGERANGE command.

Possible values: **0** abort the job with an error message (warning)

1 flush the data file and end without error

2 process the data file to the end but do not image pages after the last page in the range

/PagesPerBooklet

Category: Booklet

Default value: 1

Unit or Format: integer

Description: Number of pages a booklet should have, the value range begins at 1, 0 is invalid.

/PageWidth

Category: Layout

Unit or Format: integer

Description: The width of a logical page in a Multi-Up application.

/PDFCropping

Category	Miscellaneous
Default value	/CropBox
Unit or Format	name
Description	This parameter is used to select a PDF cropping box when a PDF is processed by SCALL or RUNPDF. The following values are available for selection: /TrimBox /CropBox (default) /ArtBox /MediaBox /BleedBox If the selected cropping box is not available in the PDF, then /MediaBox is used since a /MediaBox always exists in a PDF. Example: [/PDFCropping /TrimBox] SETPARAMS % this statement selects /TrimBox as the cropping box

/PDFTpage

Category:	Layout
Default value:	0
Unit or Format:	integer
Code:	none
Code value:	none
Description:	Target PDF page number minus one. Use to select a page number when a multi-page PDF is called with SCALL.

Examples

```
[ /PDFTpage 23 ] SETPARAMS      % select PDF page 24
(mydoc.pdf) CACHE SCALL         % print page 24
```

/PDFXembed

Category:	Image Rendition
Default value:	**3** on VI eCompose, VI Design Pro, and VI Design Express **2** on FFPS Normalizer Irrelevant on any other interpreters
Unit or Format:	Integer
Code:	None
Description:	On VI eCompose and Normalizer enable PDF/TIFF/JPEG resources called by SCALL or RUNPDF to be embedded in the resulting PDF file.
Supported values:	**0** Do not embed PDF/TIFF/JPEG resources **1** Embed PDF resources only **2** Embed TIFF/JPEG resources only **3** Embed PDF/TIFF/JPEG resources

Note: Setting /PDFXembed is intended only on the FFPS/APCSDK/APPE workflow to take full advantage of using external image resources when generating the PDF by reducing file size and latency, since the PDF can then be submitted to APPE for printing.

When using this option on FFPS make sure that the version of APPE on which VI Compose is running supports external image references. When a not embed option is selected make sure the resources are present on the platform at the time the PDF is viewed or printed.

Modifying the default setting on any other platform is not currently supported.

Examples

```
[ /PDFXembed 1 ] SETPARAMS          % embed PDF resources only
(mydoc.pdf) CACHE SCALL             % embed mydoc.pdf in the target PDF
```

/PDFXobject

Category:	Layout
Default value:	**1** on VI eCompose and FFPS Normalizer **0** on other interpreters
Unit or Format:	integer
Description:	On VIeCompose and Normalizer, enable PDF resources called by SCALL or RUNPDF to be processed as XObjects in the resulting PDF file.

Supported values:	**0** do not process PDF resources as XObjects. Use embedded EPS images if present, otherwise display a text pattern box. **1** process PDF resources as XObjects

Note: When a PDF resource is processed as an XObject the resulting PDF is placed in a trusted folder for correct viewing. For more details, refer to Adobe Acrobat, **Edit→Preferences→Security(Enhanced)**.

Examples

```
[ /PDFXobject 1 ] SETPARAMS  %enable PDF XObjects

(mydoc.pdf) CACHE SCALL      %include mydoc.pdf as an XObject in the
                             target PDF
```

/PlotSymbol

Category:	Chart
Unit or Format:	[(string) size Colorkey]
Description:	Defines a plot symbol to be printed on node points on a line chart drawn with DRAWCRV.
Possible values:	**(string)** single character from the Zapf Dingbats font. Can be a character between parenthesis ( ), or hex value between < and >. (Example: (l) = bullet, (s) = up triangle, (t)=down triangle, (u) = diamond, (n) = square, <AA> = heart, ...) **size** real **Colorkey** any Colorkey or null (use color from ColorTable)

/PrintLabel

Category:	Chart
Default value:	1
Unit or Format:	integer
Description:	Print labels on bar, pie and curve charts:
Possible values:	**0** do not print label **1** Print labels below/beside the X axis **2** Print labels at end of bars. This option automatically disables PrintValue.

Note: Text attributes switches (font, color, …) are allowed inside the text.

/PrintScale

Category:	Chart
Default value:	1
Unit or Format:	integer
Code:	+128
Code value:	0
Description:	Print scale on bar and curve charts:
Possible values:	**0/false** do not print scale **1/true** print scale at left/bottom **2** print scale at right/top

Note: For formatting options, refer to /Format.

/PrintValue

Category:	Chart
Default value:	false
Unit or Format:	true / false
Code:	+64/
Code value:	true
Description:	Print value (bar and curve) or percentage (pie). For formatting options, refer to /Format.

/ResCaseSense

Category:	Miscellaneous
Default value:	true
Unit or Format:	true/false

Description: true: enable resource name's case sensitivity (default).

false: disable resource name's case sensitivity.

Note: This parameter is used on UNIX systems only, and has no effect on Windows systems. When set to false, an unsuccessful call to resource xyz.ext, or any case combination, triggers the following additional access attempts:

- xyz.ext
- Xyz.ext
- XYZ.ext
- xyz.EXT
- Xyz.EXT
- XYZ.EXT

Examples

```
[ /ResCaseSense false ] SETPARAMS          % disable resource names
                                           % case sensitivity
```

/ResolvePath

Category: PIF
Default value: 0
Unit or Format: integer
Description: Sets the path resolution mode for fileref in PIF links
Possible values • **0** (default) Do not resolve the file reference. Leave it as provided. • **1** Try to resolve the file reference in the current VIPP® context (SETIPATH or SETMPATH). When it exists replace it with the full path. If not, leave it as provided. :

/RightBleed

Category: Layout

Default value: 0

Unit or Format: integer

Description: The value for the amount of bleed allowed at the right margin of a logical page in a Multi-Up application.

/Rotate

Category: Layout

Default value: 0

Unit or Format: integer

Description: The degrees of rotation for the logical pages.

Possible values: 0 (default), 90, 180, 270 clockwise.

/RowHeight

Category: Barcode

Default value: 3 or 4

Unit or Format: integer

Description: The height of one row in the barcode. The default value is 4 when the error level is low, given the amount of data encoded. The default is 3 when the error level is appropriate for the amount of data encoded.

/ScaleStep

Category: Chart

Unit or Format: integer or integer string

Description: Sets the scale increment.

Possible values:

Value is an integer:

0 automatic computing of the scale step.

1-10 force a scale increment between 1 and 10 or the most appropriate multiple of 10.

Value is an integer string (integer between parenthesis):

The maximum value is set to the highest value of the collection of values and the scale step is computed by dividing this value by the integer provided.

/ShadeAdjust

Category: Chart

Default value: .5

Unit or Format: real between -1 and +1

Description:	Enables shading control when 3D is true.
Possible values:	**[-1 <0]** lighter shading. The smaller values are lighter. **0** no shading **[>0 1]** darker shading. The larger values are darker.

/SHPWrap

Category:	Page composition
Unit or Format:	[(char1) ... (charN)] /ALL
Description:	Specifies a collection of wrapping characters for the SHP command in addition to the space. charx is always a single character. /ALL can be used to write vertically in a very narrow column.

Note: /SHPWrap forces wrapping only when the column width is exceeded. If the text string fits within the specified column width, no action is taken.

Examples

```
[ /SHPWrap [ (.) (@) ] ] SETPARAMS
[ /SHPWrap /ALL ] SETPARAMS
```

/SliceBurst

Category:	Chart
Default value:	0
Unit or Format:	% of Radius
Code:	+8
Code value:	0.1
Description:	Burst value (0=no burst)

/SliceSepColor

Category:	Chart
Default value:	BLACK
Unit or Format:	Colorkey
Description:	Color of slice or bar borders

/SliceSepWidth

Category:	Chart
Default value:	0.5
Unit or Format:	points
Code:	+2
Code value:	0
Description:	Line width of slice or bar borders

/SpotLabels

Category:	Chart
Unit or Format:	[(label1) (label2) … (labelN)]
Description:	Provide labels to be associated with spot colors when DRAWBAR is fed with multiple values per items. (3rd and fourth syntax examples in DRAWBAR description). Text attributes switches (font, color, …) are allowed inside the text.

/SpotOffset

Category:	Chart
Default value:	Horizontal: 1.4
Unit or Format:	% of Radius
	% of Width/2
Default value:	Vertical: 0
Unit or Format:	% of Height/2
Description:	Hor_Offset
	or
	[Hor_offset Vert_offset]
	Offset of spots from the center of the graphic (positive=offset to the right/top, negative=offset to the left/bottom)
	When omitted, Vert_offset defaults to .5.

/SpotSize

Category:	Chart
Default value:	0
Unit or Format:	% of font size
Code:	+4
Code value:	1
Description:	Size of aside spots (0=no spots, labels around or below)

/Stack

Category:	Chart
Unit or Format:	true / false
Description:	True indicates that the chart will be stacked on top of another chart and requires a transparent background.

/TableStroke

Category:	Table
Default value:	no stroke
Unit or Format:	GEPKey
Description:	Intended to stroke the external borders of a table.

Examples

```
/TableStroke GEPKey
%GEPKey = table border (stroke not fill) color
/TableStroke [GEPKey GEPKey GEPKey GEPKey ]
%Top Bottom Left Right
```

/TextAtt

Category:	Table
Unit or Format:	{ VIPP® code }
Description:	VIPP® code used to set text attributes

/TextFilter

Category:	XML
Default value:	true
Unit or Format:	true/false
Possible values:	**true** enables filtering of extra spaces and control characters in XML node contents. **false** disables filtering of extra spaces and control characters in XML node contents.

Note: For more information, refer to Stripping Blank and Control Characters in the *FreeFlow VI Compose User Guide*.

/TextFit

Category:	Page Composition
Default value:	0
Unit or Format:	integer
Description:	Select text fitting method for SHMF and SHP.
Possible values:	**0** anamorphic scale (horizontal only) **1** isomorphic scale (horizontal and vertical)

/TimeZone

Category:	Date
Unit or Format:	integer
Description:	Time zone, +- minutes from UTC. (ex: -480 = PST)

/TimeZoneName

Category:	Date
Unit or Format:	array of 2 strings
Description:	Time zone names: Standard time, Daylight Saving time.

Examples

```
[ (PST) (PDT) ]
```

/TopBleed

Category: Layout

Default value: 0

Unit or Format: integer

Description: The amount of bleed allowed at the top margin of a logical page in a Multi-Up application.

/TransWeight

Category: Chart

Default value: 1

Unit or Format: integer

Description: Improves black pixels on soft edges of transparent areas.

Possible values:

1 eliminate black pixels (default)
< 1 reduce black pixels

0 keep black pixel (current behavior)

/ValueColor

Category: Chart

Unit or Format: Colorkey

Description: DRAWBAR/DRAWCRV only. Set a different Colorkey for the values (default is the current color).

/VGutter

Category: Layout

Default value: 0

Unit or Format: integer

Description: The value for the vertical gutter between logical pages in a Multi-Up application.

/Width

Category:	Table
Unit or Format:	real
Description:	Width of a table cell. Default is current units.

/XFloat

Category:	Chart
Default value:	false
Unit or Format:	boolean
Description:	Enables/disables floating of the X axis on DRAWBAR and DRAWCRV.
Possible values:	**false** the X axis is always at zero **true** the X axis is floating. It is close to the minimum or maximum value of the set of values.

/XMLMisVal

Category:	Miscellaneous
Default value:	empty string
Unit or Format:	string
Code:	none
Description:	Sets the string delivered when an XML node is missing.

Examples

```
[ /XMLMisVal (** Missing **) ] SETPARAMS % set missing node value
```

/ZSRepeatField

Category:	Miscellaneous
Unit or Format:	field name or variable

Note: ZSRepeatField cannot be set in the DBM. You can set ZSRepeatField before the **STARTDBM** command in the submission file or in a **JDT** that is called before the **STARTDBM** command.

Description:	Declares repeat field for ZSORT

7

Composite Constructs

This chapter contains:

Composite constructs use expressions and test operators with the VIPP® language to provide greater functionality in the VIPP® application. The VIPP® language uses the test operators available in the PostScript language, and has several VIPP® test operators, which are described below.

For further information on PostScript test operators see the PostScript Language Reference produced by Adobe Systems Incorporated. The latest edition of this manual is downloadable from this site: https://partners.adobe.com/public/developer/ps/index_specs.html

Arithmetic Expressions

Arithmetic expressions allow you to combine numeric variables, numeric constants, and arithmetic operators into a single operand that is passed to a VIPP® command.

A VIPP® arithmetic expression can be formally defined as:

```
expression          ::= member | unop member |
                        expression (binop expression)* |
                        «expression» | unop «expression»
member              ::= numeric-variable | numeric-constant
binop               ::= '+' | '-' | '*' | ':' | 'M' | 'm' | 'q' | 'r'
unop                ::= - | + | #
numeric-variable    ::= any valid VIPP variable with numeric contents
numeric-constant    ::= any valid PostScript integer or real
```

These are the available binary arithmetic operators:

+	addition
-	subtraction
*	multiplication
:	division
M	maximum
m	minimum
q	quotient
Q	quotient ceiling rounded to the next integer
r	remainder

Note: To avoid conflict with variable names, bracket the binary operators with single quotes, for example: '+', '-', and so on.

The following unary operators are available. Unary operators do not require single quotes.

+	positive value
-	negative value
#	absolute value

The following limitations apply to an arithmetic expression:

- Cannot exceed 127 characters in length
- Fits on one line
- Cannot contain PostScript delimiters such as: , space, /, [,], {, }, <, >, (,), or %.

Generally, expressions are evaluated from left to right, however, multiplication and division expressions are evaluated before most other expressions. Sub-expressions, which are encapsulated between « and » signs, are evaluated first. To produce the symbols on a PC keyboard, ensure that the Num Lock feature is on, then use the numbers on the alphanumeric keyboard, and use ALT+174 and ALT+175.

The following is the order of evaluation:

- + (unary)
- - (unary)
- # (unary)
- 'r'
- 'q'
- 'm'
- 'M'
- ':'
- '*'
- '-'
- '+'

An arithmetic expression always delivers an integer or real number, whichever applies, regardless of the types of the members contained in the expression: integer, real, or numeric string.

A useful application for unary operators is to convert a numeric string to an integer or real number for accurate numeric comparisons in a condition statement such as `IF`/`ELSE`/`ENDIF`.

Do not use when you require numbers to print accurately

Do not use arithmetic expressions to compute numbers that must be printed accurately. Arithmetic expressions are reserved for placement calculations, repeat counts, and so on.

To accurately compute numeric strings for printing purposes

Use the VIPP® commands `ADD`, `SUB`, `MUL`, and `DIV`, which provide up to 25 integer and 15 decimal digits.

Examples

These examples assume these definitions:

```
/VAR1 100 SETVAR

/VAR2 23 SETVAR
```

Expressions and results:

```
VAR1'+'VAR2                          123
VAR1'-'VAR2                          77
VAR1'*'2                             200
VAR1'+'VAR2'*'.35                    108.05
«VAR1'+'VAR2»'*'.35                  43.05
-VAR2'*'10                           -230

VAR1'q'VAR2                          4
VAR1'Q'VAR2                          5

/VAR1 «VAR1':'2'+'.5»'q'1 SETVAR     % divide by 2 and round
```

Commands using expressions as an operand:

```
«VAR1'+'VAR2»'*'.35 VAR3 MOVETO
VAR_LM VAR_TM'+'270 VAR_RM'-'VAR_LM 0 S1 DRAWB
COLW'-'420 0 440 90 XLTR_S1 DRAWB
```

Using unary operator and assuming that CopyCount is a database field:

```
IF +CopyCount 50 gt ...
```

Test Operators and Conditional Expressions

You can build conditional expressions using variables, constants, and test operators. The result is always a boolean value of true or false. Conditional expressions can be used as a condition in an `IF/ELSE/ENDIF` statement, or stored for later reference using `SETVAR`.

Examples

```
IF CITY (Paris) eq { … } ENDIF
IF CITY (Paris) eq COUNTRY (France) eq and { … } ENDIF
/VAR_PAR_FR CITY (Paris) eq COUNTRY (France) eq and SETVAR
IF VAR_PAR_FR { … } ENDIF
IF VAR_PAR_FR not { … } ENDIF
IF VAR1'+'VAR2 200 gt { … } ENDIF
```

You can use test operators in `SETRCD` and `SETPCD` statements when you precede the operators with a slash (/).

PostScript test operators include:

eq	equal
ne	not equal
gt	greater than
ge	greater than or equal
lt	less than
le	less than or equal
or	combine two tests results or boolean with `or`
and	combine two tests results or boolean with `and`
not	negates the result of a test or boolean statement

The VIPP® language has been expanded to include these test operators:

CIEQ	Case Insensitive Equal
CINE	Case Insensitive Not Equal
HOLD	searches for a string

CIEQ and CINE

CIEQ (Case Insensitive EQual) and **CINE** (Case Insensitive Not Equal) are test operators that allow you to compare strings, regardless of the letter case within the string. You can use the operators to compare strings in the same way that the PostScript operators eq and ne are used.

The operators rely on the encoding of the current active font, for example, the last font set by **SETFONT** or **INDEXFONT** index. Two characters in the strings being compared are considered equal when their character names in the encoding array are equal, regardless of the letter case: /a=/A, /eacute=/Eacute.

Examples

```
(Hello World) (hello WORLD) CIEQ
```

returns `true`.

Related commands

IF/ELSE/ELIF/ENDIF, SETPCD, SETRCD

/HOLD

The HOLD command searches for the second string IF/ELSE or the comparison string RCD/PCD anywhere in the first string IF/ELSE or in the selected record portion of the data RCD/PCD.

Examples

This example is true when the word DIVISION appears anywhere in the first 100 positions of the record. A search of every record and every byte in the record for a string can affect performance.

```
/IF_CND1 0 100 /HOLD (DIVISION) SETRCD
```

```
IF ADDRESS2 (street) HOLD
```

Related commands

IF/ELSE/ELIF/ENDIF, SETPCD, SETRCD.

8

Error Messages

This chapter contains:

VI Compose error messages are specific VI Compose errors that you can encounter in addition to standard PostScript errors when using VI Compose. Refer to the appropriate documentation for descriptions of PostScript errors.

Error Handling

PostScript and VI Compose errors are printed on error sheets. This section contains information on how error messages are formatted on the error sheets.

PostScript Errors

In general, PostScript errors are documented on an error sheet produced by the printer when the error occurs. When a PostScript error is printed on an error sheet, this information is included:

- Offending Command: Lists the PostScript operator that encountered the error
- Error Type: Lists the error type

In addition, the contents of the interpreter stacks can be listed.

If you are unable to determine the cause of an error using the information contained on the error sheet, attempt to isolate the source of the error by commenting out some part of the job. To comment out a line in a job, insert `%` at the start of the line. If the source of the error cannot be determined using this method, contact a Xerox representative for assistance.

Error Sheet

Occasionally, a job cannot print, even when it is clear that the printer received the job. When this is the case, it is likely that a PostScript error has occurred, but that the error sheet printing option is not enabled. Refer to the printer documentation for information on how to enable the error sheet option. When the error sheet printing option is enabled, print the job again.

Link File Execution

When an error occurs during automatic VIPP® link file execution, the error sheet does not specify which linked program generated the error. In this case, disable the automatic VIPP® link file execution, then comment out the sequence `/usr/xgf/src/xgf`. Run the startup file, then submit the link file as a normal VIPP® file. Next, comment out lines in the link file, starting from the bottom, to isolate the file causing the error.

Locating the Cause of an Error

When you are using a large application or having difficulty locating an error, place an invalid string in the application for use in locating the error. For example, enter the string in the middle of the JDT or DBM file to determine whether the error you want to find is located above or below the string you entered. The invalid string you enter cannot be a VIPP® command.

When you resubmit the application after entering the invalid string, and the error condition changes to an offending command that refers to the invalid string you inserted, then the error is located after the invalid string you entered. Determine the file initiating the error and at which line the error is located by moving the invalid string.

With VI Design Pro, lines or sections of code can be commented out until the desired results is achieved. This can quickly isolate problem areas in the code that require modification. For further information, refer to the *FreeFlow VI Design Pro User Guide*.

For information on proper VIPP® syntax usage, refer to VIPP® Commands.

VIPP® Error List

VIPP® errors are a subset of PostScript errors. These errors are listed in the Error field beginning with the text VIPP_ and are documented in the following sections.

VIPP_access_denied

A resource, such as a form, segment, JDT, image, or font list, located, but the interpreter does not have the access rights necessary to access it.

VIPP_AFM_parsing_error

VI Compose encountered an error while parsing an Adobe Font Metrics file for kerning information. Check for integrity and completeness of the AFM files referenced by the SETENCODING command.

VIPP_ambiguous_name in _name

The name pointed out by this error message creates a conflict. In XML mode it means that VI Compose encountered an VXVpath name that is not unique. The VXVname is additionally qualified to remove the ambiguity.

In database mode, this name is already a reserved word and cannot be used as a field name; choose another name for that field.

A mix of upper and lowercase letters, or field names that begin with a prefix such as '_' is recommended. For example, status is a reserved keyword and would cause this error message to appear. However, Status or _status are valid field names, and cannot produce the message.

VIPP_buffer_overflow

An ADD or SUB command caused an overflow on VI Compose internal computing buffers, which are limited to 40 digits. 25 digits for the integer and 15 digits for the decimal.

VIPP_corrupted_or_unsupported_image_file

Indicates that a corrupted or unsupported image file are encountered.

VIPP_invalid_align in SHMF

The SHMF, SHMf, and SHmf align parameter is invalid. This parameter cannot be greater than three.

VIPP_invalid_align in SHP

The SHP and SHp align parameter is invalid. This parameter cannot be greater than five or equal to three.

VIPP_invalid_align in SHX

The SHX align parameter is invalid. This parameter cannot be greater than five.

VIPP_invalid_booklet_length

A booklet is not of the required length specified by /PagePerBooklet in SETPARAMS. The job has aborted as required by the /BookletMismatch parameter.

VIPP_invalid_color

The color used in conjunction with an **MPR** command is not suitable for this command. Black is the only color supported on Monochrome devices.

VIPP_invalid_combination in Multi-up_COLLATE_off

Multi-up and **COLLATE_off** cannot be combined in Database mode

VIPP_invalid_combination in STOREVAR_file_must_exist

When **STOREVAR** is used in project mode, it is necessary that the file exists.

VIPP_invalid_combination in STOREVAR_VIeCmulti

A **STOREVAR** command is used in VIeC multi-instance mode.

VIPP_invalid_combination in UV2L

The underlying UV color does not support a second layer.

VIPP_invalid_contents in ENDPAGE

An ENDPAGE command with the /P option contains marking commands in its procedure. Place the marking commands in an **ENDPAGE** procedure without the /P option.

VIPP_invalid_font

The font used in conjunction with an **MPR** or **GLT** command is not suitable for these functions.

VIPP_invalid_license_file

The file currently referenced by `/usr/xgf/src/xgf.lic` is not a license file or it has been corrupted. Reload the original file.

VIPP_invalid_PIF_type

The /PIFtype parameter of a SETPIF or INDEXPIF command is invalid. Ensure that the parameter is one of the types listed in the **SETPIF** and **INDEXPIF** command descriptions.

VIPP_invalid_PN_option

The SETPAGENUMBER pos or align parameter is invalid. Ensure that this parameter is less than zero or greater than seven.

VIPP_invalid_syntax in RPE

An entry between BEGINRPE and ENDRPE in an RPE definition is invalid. Ensure that the entry is either a table that contains ten items or a Record Criteria Definition (RCD) key defined by SETRCD.

VIPP_invalid_syntax in SETBAT

The number of parameters in a SETBAT statement is not a multiple of 13.

VIPP_invalid_syntax in SETMULTIUP

A SETMULTIUP definition contains an incorrect number of elements. Ensure that the total number of elements in a table is a multiple of five.

VIPP_invalid_syntax in SHP

An SHP and SHp definition is missing an align parameter.

VIPP_invalid_variable_name

An invalid variable name has been detected. Either the name does not start with VAR or ^ or its length is greater than 127 characters.

VIPP_invalid_VSUB

While substituting a variable name, VSUB encountered an incorrect syntax. In general, this occurs when there is no closing period (.) following the two dollar signs ($$).

VIPP_length_error in ENDPCC

An entry in a PCC table between BEGINPCC and ENDRPE contains an incorrect number of elements. Ensure that the entry has three characters.

VIPP_length_error in RPE

An entry between BEGINRPE and ENDRPE in an RPE definition contains an invalid number of elements. Ensure that the number of elements is ten.

VIPP_license_failed

The current VI Compose license is not valid for this device. Contact a Xerox representative to update the license.

VIPP_misplaced in SETPAGESIZE

A SETPAGESIZE command is placed on the page after the first marking command. Place the **SETPAGESIZE** command at the beginning of the page before any marking commands.

VIPP_OMR_invalid_response_string

One or more characters in the response string was not found in the OMR map string.

VIPP_PDF417_data_limit_exceeded

The total number of characters in the string parameters of a PDF417 command exceeds the maximum number allowed.

VIPP_PDF417_invalid_mode

An invalid mode is generated during the compaction of a text string parameter of a PDF417 command.

VIPP_plane_number_out_of_range

A SETFORM or SETBFORM or command has been given a plane number greater than the maximum previously defined by SETMAXFORM or SETMAXBFORM.

VIPP_POSTNET_invalid_digit

The string parameter of a POSTNET transform function contains an invalid character. Only numeric characters, 0–9, and dash (-) in the fifth position are allowed.

VIPP_POSTNET_invalid_length

The string parameter of a POSTNET transform function has an invalid length. Ensure that the string parameter is 5, 9, or 11.

VIPP_RPE_invalid_prefix

A prefixed record with a prefix undefined in the current RPE definition was encountered. This error occurs only when RPE prefix control is enabled. SETRPEPREFIX sets the prefix length with a negative value.

VIPP_RPE_invalid_prefix_length

A prefixed record with a prefix shorter than the length defined by SETRPEPREFIX is encountered.

VIPP_SETVAR_invalid_name

A variable name defined by SETVAR cannot begin with VAR.

VIPP_unable_to_locate

A resource, such as a form, segment, JDT, image, or font list, is not available in any of the libraries referenced by the corresponding **SETxPATH** command.

Miscellaneous Errors

These errors can occur as a result of the interaction between VI Compose and other programs.

Limitcheck Error

This error message can occur when using a VIPP® form created with the mkfrm utility from an EPS file that was created using a Windows driver:

```
%%[Error: limitcheck; Offending Command:f1fff500defff500.........
Stack:
(eform.frm)
-mark-
[1]
0
```

This error can indicate that the form contains bitmap images that are called using the PostScript operators for the current file image.

Apply one of these solutions to solve this problem:

- Eliminate the problem by using the original PostScript file with this syntax:

  ```
  { (eform.ps) CACHE SETFORM
  ```

- Submit the EPS file to the Decomposition Services and use the result as a form. For more details about decomposition services, refer to the *FreeFlow VI Compose User Guide*.

PostScript

There is a runtime error in the PostScript interpreter. For detailed information regarding this error message, refer to the appropriate PostScript manual.

PostScript Error - dictfull / Offending Command:def

This error can occur because of a large number of variables, as defined by SETVAR, GETFIELD, or a large number of fields in the database file, when you are using a PostScript Level 1 interpreter. Use a PostScript Level 2 interpreter to solve this problem.

Selected Pages: <first> <last>

This error indicates one of the following:

- PAGERANGE is used for this job. The message reflects the first and last pages printed.
- The xgf.lic license in use is a demonstration or evaluation license, which causes the job to abort after 200 pages on production printers, and after 10 pages on all other printers. In this case, replace the demonstration or evaluation license with a production license.

Selected Booklets: <first> <last>

This error indicates that BOOKLETRANGE is used for this job. The message indicates the first and last booklets printed.

Stack Overflow Error in Ghostscript

The Stack Overflow error occurs when you have exceeded the maximum number of tokens, that is PostScript words, that can be accommodated on the operand stack.

However, this error does not occur on a DocuPrint NPS, because the operand stack can hold 5000 tokens by default and can be configured to a larger number, when required.

If this error occurs, Xerox recommends that you upgrade to the latest version of the Ghostscript software. The limit for this version of the application is extended to over 10.000.

9

Programming Tips

This chapter contains:

The information found in Programming tips is intended to answer some frequently asked questions about how to use various aspects of the VIPP® language.

Consume vs. Execute

When running VIPP® jobs, data and resources are not processed in the same manner. Data, everything that occurs after a **STARTLM**, **STARTDBM**, or **STARTXML** command is consumed. VIPP® resources such as forms, JDT, DBM. or XJT is executed.

Attribute switches such as font, color, and so on, are VIPP® constructs and interprets VIPP® commands such as **SHMF**, **SHP** and so on without regard to their origin either in consumed or executed strings. Octal character notations are pure PostScript constructs and are interpreted only when the strings are executed. Therefore, when a data field from a DBF field or a GETFIELD contains octal character notations it can be executed to get those octal characters interpreted. This can be done with VIPP® using a VSUB2 construct as in this example:

```
(($$VAR1.)) VSUB2
```

Assuming VAR1 contains xxx\127yyy, the above sequence first substitutes VAR1 with its contents, producing this intermediate string:

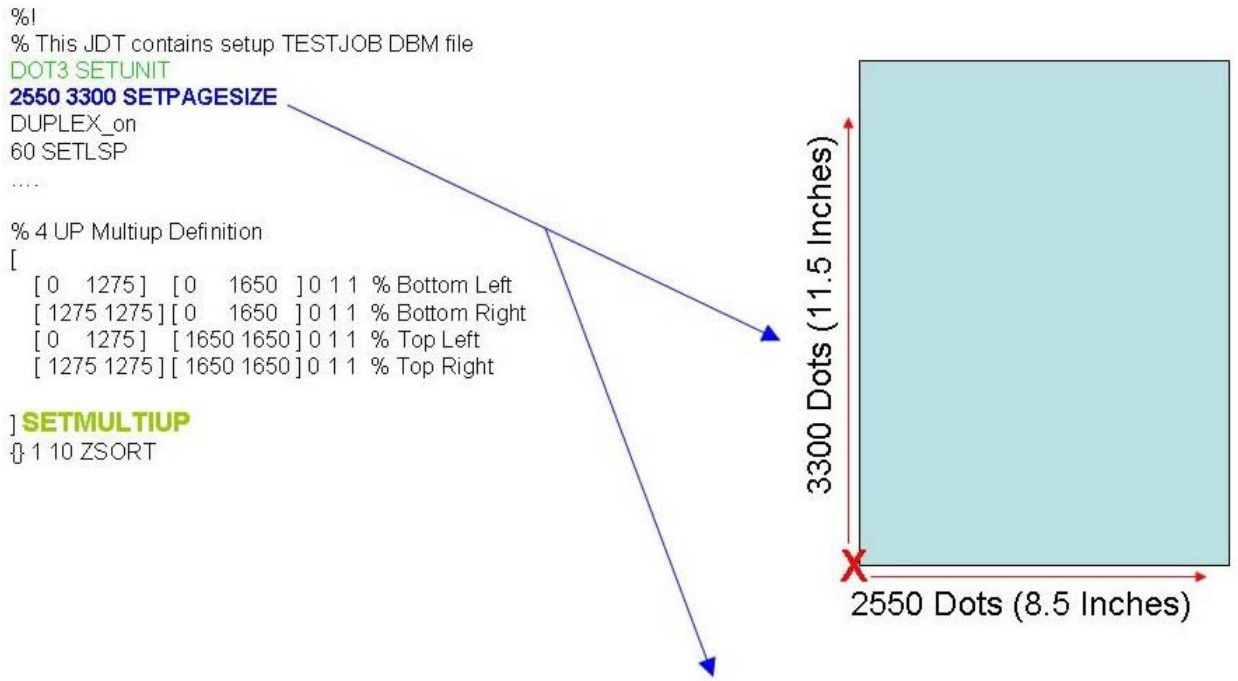
```
((xxx\127yyy))
```

The intermediate string is then executed to produce the final result:

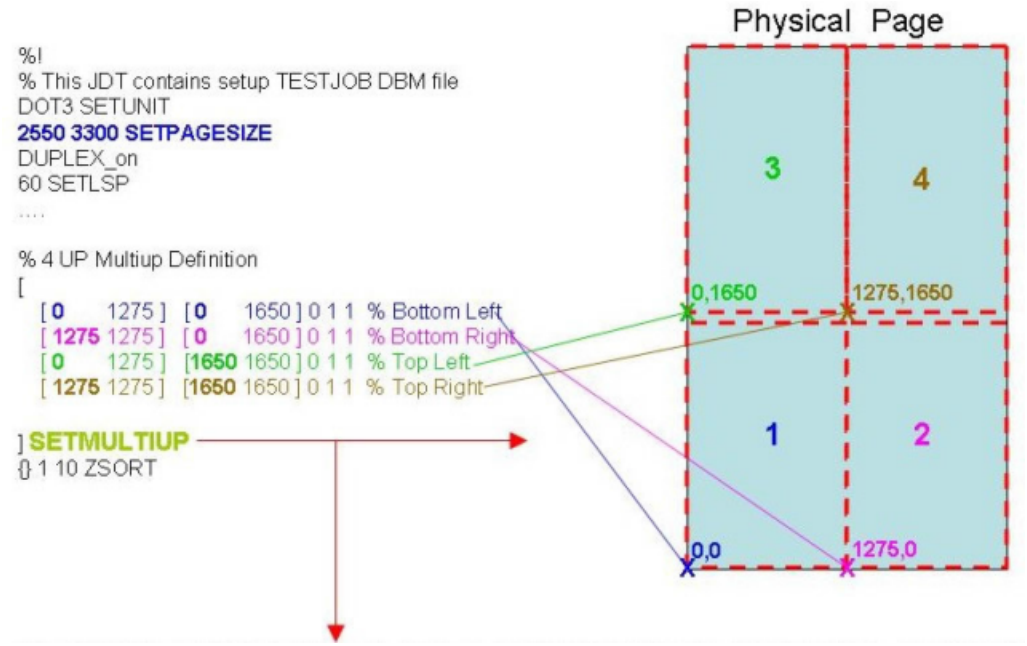
```
(xxxWyyy)
```

0,0 Origin for Object and Logical Page Placement

The graphics below describe the origin (X,Y) position and layout often used to create a four-up postcard application.

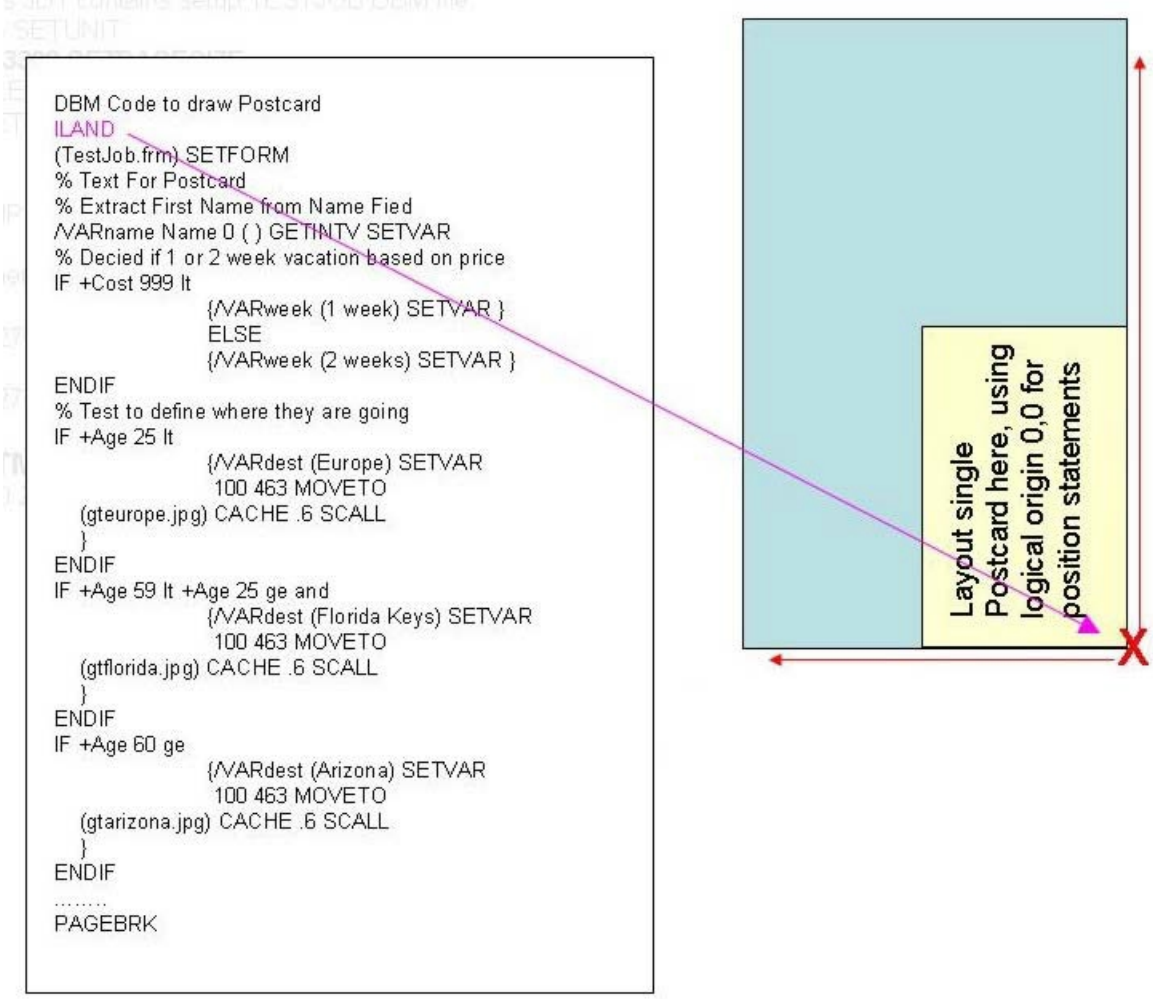


This defines the size of the Physical page. Using the current unit definition, in this case DOT3, 300 dpi, settings. The definition is for a portrait page, USLetter. The red X marks the 0,0 origin.

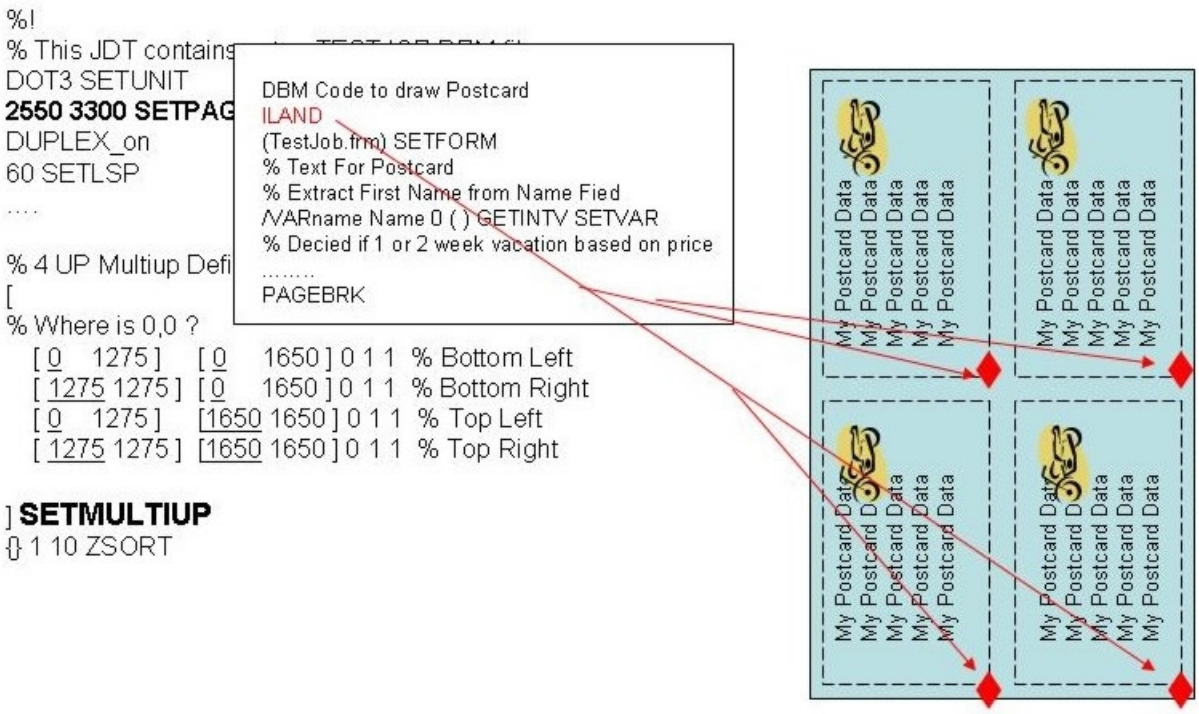


The **SETMULTIUP** statement defines each of the four logical pages. Each logical page is defined by

specifying the starting X,Y coordinate, based on the physical page definition for **SETPAGESIZE**, then the width and height from that coordinate for that logical page. The color X shows the starting X,Y position based on the physical page origin for each of the four logical pages that are defined. From that position based each logical page is 1275 dots in width and 1650 dots in height. The order in which each logical page can be used is based on the order they are defined within the **SETMULTIUP** command.



In the **DBM**, the **ILAND** command sets the 0,0 origin as indicated by the red X. Design a single postcard, using the red X as the page 0,0 origin. If required, design both a front and a back page. Later when implementing the **ZSORT** command you can enclose the front and back page code in an IF BACK IF/ELSE/ENDIF statement.



Based on the combination of the **SETPAGESIZE** command, which defines the size of the physical page and the **SETMULTIUP** command that defines the four logical pages on the physical page in the JDT file and the **ILAND** command in the DBM file which defines an **ILAND** layout for the postcard (optional) the result is shown above. The four postcards placed on the physical page, each in the logical page area defined by the **SETMULTIUP** command. The Postcard is in **ILAND** orientation with the 0,0 origin shown by the red diamond. The order that the records are laid down is defined by the order the logical pages are defined in the **SETMULTIUP** statement.

Cyclecopy Control

Cyclecopy control commands are:

CHKPOINT	COLLATE_on	RPCOUNT	SETMAXCOPY
COLLATE_dbm	COPYRANGE	RPLEFT	SPOOLNAME
COLLATE_off	REPEAT	SETCYCLECOPY	

For more information on Cyclecopy control functions, refer to Using Cycle Forms and Cycle JDTs.

Using Cycle Forms and Cycle JDTs

Forms and JDTs can be used in a cyclical manner to perform tasks equivalent to those provided using cycle forms, for example, to send out five forms to each customer. The first three forms are identical, and forms four and five are different. The form data arrives in Sysout mode. The data for the first three forms is sent down once, followed by pages of data for forms four and five. Use this JDT to perform this task:

```
%!
PORT% or whatever
3 SETCYCLECOPY
COLLATE_off
[ (form1.frm) (form2.frm) (form3.frm) ] SETFORM
..........% layout definition (RPE or listing)
/JDT1 { 3 SETCYCLECOPY } XGFRESDEF
/JDT2 { 1 SETCYCLECOPY } XGFRESDEF
[ (JDT1) (JDT2) (JDT2) ]  SETJDT
```

Date and Time

Date and time variables are:

D_DD	D_MOL	GETDATE	T_HH2
D_DOY	D_MOS	SETDATE	T_MM
D_DWL	D_YY	SHIFTDATE	T_SS
D_DWS	D_YYYY	T_AMPM	T_TZN
D_MO	DAYS	T_HH	

Design and Debugging

Design and debugging commands are:

AUTOGRID	NMP_off
BLGRID	TLGRID
FSHOW	XGFDEBUG

Use this information to help design and debug the VIPP® jobs. The information is divided into these categories:

- Application Design
- Debugging and Documentation Tools
- Resource Creation and Maintenance

Application Design

FreeFlow VI Design Pro (VDP), formerly VIPP® IDE, is a tool that provides three editing modes, a text editor, a smart editor, and a graphical user interface editor. The full set of VIPP® commands, including Data Driven Graphics, is available to VDP users. The graphical interface provides immediate feedback on any changes made in the code, which greatly reduces the time required to design and code a VIPP® application.

VDP is the best diagnostic tool for VIPP® applications, as the VIPP® code can be run in the VDP and the results shown on the GUI window. VDP allows you to make changes and view the results electronically, without printing hardcopy.

VI Design Express (VDE) is an application plug-in for use with Adobe® InDesign. It is designed to allow a graphic artist to:

- Create applications that contain variable information.
- Export the variable data application.
- Print the application at production speeds when exporting the job using the VI Project Container format (.vpc).
- Set in-line feeding and finishing for the application (device dependent).
- Allow further advanced design modification using the code based FreeFlow VI Design Pro tool or allow processing to PDF using VI eCompose.

VDE applications can be opened in VI Design Pro for diagnostic purposes.

Debugging and Documentation Tools

Use these commands to aid in debugging applications and generating documentation:

XGFDEBUG	VIPP® command
NMP_off	VIPP® command
SOF_off	VIPP® command
AUTOGRID	VIPP® built-in variable

BLGRID	VIPP® built-in variable
TLGRID	VIPP® built-in variable
FSHOW	VIPP® command
SHPOS	VIPP® built-in variable

For more information, refer to Locating the Cause of an Error and to the *FreeFlow VI Design Pro User Guide*.

Resource Creation and Maintenance

This section contains information on these topics:

- Document Applications
- Image tools
- Resource libraries
- Text editors

Document Applications

High-level document processing applications such as Word, FrameMaker, Excel, and PageMaker can be used to create resources such as forms or segments. After the document is created, you can generate a PostScript file on disk using an appropriate PostScript driver. Then place the PostScript file into a VIPP® form library and invoke it from the VIPP® job using CACHE/SETFORM or CACHE/SCALL. The file can also be converted to a TIFF file using Decomposition Services.

Note: PostScript drivers do not always produce PostScript code appropriate for production printing. Therefore, the use of the DocuPrint NPS Decomposition Service tool can be more effective.

Image tools

Use scanner control software and TIFF or JPEG editors to create, edit, and transform images. For more information on supported TIFF and JPEG files, refer to ICALL.

Resource libraries

For security purposes, production resources and development resources are stored usually in separate libraries. The VIPP® link file refers to the production libraries. For further information, refer to Link and setup files in the *FreeFlow VI Compose User Guide*. To access the development libraries, test jobs can include SETxPATH commands that force VI Compose to access the resources contained in these libraries for use during the duration of the job.

Text editors

All resources, except images and fonts, can be created using a common text editor available on any system. Text editors are the easiest tools to use for this purpose as they allow use of plain text and the resources can be maintained directly from the libraries in which they are stored.

Fonts and Colors

Font and color commands are:

GLT	INDEXLSP	SETCOL	SETTXB
HCOLOR	INDEXOTL	SETENCODING	SETTXC
INDEXALIGN	INDEXSST	SETFONT	SETTXS
INDEXBAT	MPR	SETFTSW	SETV2HCONV
INDEXCOLOR	SETBAT	SETGEP	SETV2HTABLE
INDEXFONT	SETCJKENCMAP	SETKERN	
INDEXKERN	SETCJKRULES	SETOTL	

Font, color and variable information includes these topics:

- Applying attributes to fonts
- Color tints
- Fixed pitch and barcode fonts
- Kerning
- Multi-byte fonts
- Re-encoded fonts and run/RUN statements
- Setting automatic font size
- Setting bold in the center of a record or line
- Specialty Imaging
- GlossMark and Correlation fonts (GL/CR)
- Using multi-byte (CJK) fonts
- Using PostScript files with embedded fonts as forms

For information about built-in Colorkeys, refer to VIPP® Colorkeys.

Applying Attributes to Fonts

Attributes, such as Bold, Italic, and Bold/Italic, cannot be applied to a font unless the font with that attribute is installed on the system.

The mechanism used to apply attributes to fonts is based on a font family declaration in the fontlist file, `xgf/encoding/fontlist`, or any other font list used in SETENCODING. A font family is made up of up to four fonts encapsulated between the **/STARTFF** and **/ENDFF** commands in that file.

When any of the member fonts is selected, the others can be accessed using the /~REG, /~BLD, /~ITL, or /~BDI font names.

When the font is not part of a family, or when the corresponding entry is missing, the SETFONT/ INDEXFONT command is ignored and the current font remains active.

Color Tints

A color tint can be applied to any CMYK or RGB Colorkey argument to a VIPP® command referencing a Colorkey for **SETTXC**, **INDEXCOLOR**.

Syntax

```
/colorname~T SETTXC
(colorname~T) SETTXC
```

Where:

~	indicates that a tint value is attached to the color name
T	is a tint value with a valid range from 0 to 1

Examples

Three color tint examples are shown here:

```
(customColorkey~.5) SETTXC   % Solid color key with a tint of 50%
(BLUE~.3) SETTXC             % RGB color key with a tint of 30%
/PURPLE~.7 SETTXC            % RGB color key with a tint of 70%
```

The tint factor is applied to each color component in this example:

If:

```
/Color1 [ .3 .6 .2 .8 ] SETCOL
```

then:

```
(Color1~.3)
```

is equivalent to:

```
[ .09 .18 .06 .24 ]
```

For RGB and gray, the formula is more complex because the intensity range is from 1 (no color) to 0 (full color) instead of 0 to 1.

Color Transparency

Color transparency can be applied to any CMYK or RGB Colorkey argument to a VIPP® command referencing a Colorkey for SETTXC, INDEXCOLOR. This is effective only in VI eCompose and APPE RIP. On a PS RIP color transparency is emulated using an opaque tint.

Syntax

```
/colorname#T SETTXC
```

```
(colorname#T) SETTXC
```

Where:

#	indicates that a transparency value is attached to the color name
T	is a transparency value with a valid range from 0 to 1

Examples

Three color transparency examples are shown here:

```
(customColorkey#.5) SETTXC   % Solid color key with a transparency of 50%
(BLUE#.3) SETTXC             % RGB color key with a transparency of 30%

/PURPLE#.7 SETTXC            % RGB color key with a transparency of 70%
```

Spot Colors Versus Process Colors

CMYK colors can be defined as a set of CMYK values used by the Digital Front End (DFE) to render the color. This is called a Process color.

However, some DFEs have the capability to process special colors in the CMYK color space known as Spot colors or Custom colors. Such colors are identified by a specific color name and a set of CMYK approximation values that is used by the DFE when it does not have the capability to process spot colors or when the color name has not been defined in its context.

VI Compose supports both types of color depending on the syntax used to reference them:

- To define the color name and associated CMYK values with the SETCOL command:

  ```
  /MyCustomColor [ c m y k ] SETCOL
  ```

- To use the color as a process color, reference the color in any VIPP® command that requires a Colorkey by the plain color name:

  ```
  /MyCustomColor [ c m y k ] SETCOL
  ```

- To process the color as a spot color reference the color by placing the color name between parenthesis or optionally preceded by a forward slash when the name does not contain any space):

  ```
  (MyCustomColor) SETTXC
  ```

  ```
  /MyCustomColor SETTXC
  ```

Examples

```
/MyPersonnalRed [0 1 1 .8 ] SETCOL
...
(MyPersonnalRed) SETTXC     % creates a spot/custom color
/MyPersonnalRed SETTXC      % creates a spot/custom color
 MyPersonnalRed SETTXC      % creates a process color
```

In order to process a color as a spot color, when the name contains spaces or when a tint is applied, the parentheses are mandatory. Refer to Color Tints.

```
(Another Custom Color) [ c m y k ] SETCOL
...
(Another Custom Color) SETTXC
```

VIPP® comes with a collection of pre-defined spot colors, refer to Solid Coated and Uncoated Color Simulation.

Fixed Pitch and Barcode Fonts

Various fixed pitch type 1 fonts and barcode fonts are available, to download them refer to Font download.

Adobe fonts can be used on the DocuPrint NPS and FreeFlow Print Server printers. Obtain Adobe fonts by contacting Adobe and ordering the fonts directly. For information about downloading fonts from Xerox, refer to Font download.

Kerning

Kerning refers to the ability to adjust the amount of space between characters when imaging a block of text with a given font.

There are two types of kerning, which can be used independently or together:

- Pair-wise kerning is applied to specific character pairs
- Track kerning is applied to all characters uniformly

Kerning information for a PostScript font is available in a file called an Adobe Font Metrics (AFM) file. AFM files are generally supplied with the font kit when the font is purchased. Specifications for the AFM file can be obtained from Adobe, Inc.

Kerning can be enabled for SHX commands using:

- Extended syntax on the entry in the font list supplied to SETENCODING
- SETKERN and INDEXKERN commands

Use this syntax to establish the link between a given font and an AFM file when supplying a fontlist to SETENCODING:

```
/vipp_font_name [ /PS_font_name (AFM_filename) ]
```

Where:

AFM_filename	is the name of the AFM file containing the kerning information for the associated font.

The AFM file is located in one of the libraries referenced by SETMPATH or SETEPATH or SETPPATH in project mode.

Kerning is disabled by default.

Examples

```
/NHE [ /Helvetica (helvetica.afm) ]
```

AFK Files

AFK files are files used by VI Compose. These generic files, which are a subset of AFM files, contain generic kerning information that can be used when an AFM file for a given font is not available. The AFK files only exist in the `xgf/encoding` directory and are referenced in `xgf/encoding/fontlist` in case a user activates kerning with one of the VIPP® fonts listed there. There are two sets of default files, one for serif and one for sans serif fonts.

AFK files can be used with any font. Choose one of them depending on the type (serif or sans serif) and attribute of the font (regular, bold, italic, bold-italic). However, because they are generic, the result cannot be 100 % accurate. For accurate results the original AFM file provided by Adobe must be used.

Examples

This syntax references an AFK/AFM file in a font list:

```
/new_font_name [ /PS_font_name (AFK/AFM_filename) ]
```

This syntax references an AFK/AFM file in-line SETENCODING:

```
[ /new_font_name [ /PS_font_name (AFK/AFM_filename) ]] (encoding table)
SETENCODING
```

Multi-byte Fonts

In the Variable Information Suite documentation, the term multi-byte font is used to describe those fonts which require more characters than can be specified in an 8-bit byte. VIPP® specifications include Chinese, Japanese, and Korean fonts.

Re-encoded Fonts and run or RUN Statements

Some applications use either the run or RUN commands to load fonts. Loading fonts in this manner works as long as the fonts do not need to be re-encoded.

Re-encoding has been optimized and now only occurs the first time the font is called. Since this is likely to be in the middle of a page, the fonts can be defined in global VM to escape save/restore.

When run or RUN are used, the fonts are defined in local VM and cannot be re-encoded with SETENCODING.

Because of this Xerox recommends that existing run/RUN statements be replaced by the SETENCODING syntax with a font file reference.

For example, change this statement:

```
(myfont.pfa) RUN
```

to:

```
[ /R_myfont (myfont.pfa) ] null SETENCODING
```

When running the fonts ahead of VIPP® in some PS startup files, the run statements is encapsulated as follows:

```
currentglobal true setglobal          % save VM mode and force global VM
(.../.../font1) run
(.../.../font2) run
.....
setglobal                             % restore VM mode
```

Setting Automatic Font Size

Use SETGRID and SETMARGIN to change the characters per line (CPL) and the lines per page (LPP). This makes fixed pitch font scaling, and therefore LPP and CPL, easy. Once you set these two parameters, use the SETFONT command, with 0 specified as the size, for automatic scaling.

Setting Bold in the Center of a Record or Line

To specify that you want bold in the center of a line, use SHMF, SHMf, and SHmf in the RPE entry:

```
[ {0 SHMF} 0 Xinit Xdispl Yinit Ydispl rec_pos length /font color ]
```

Define the fonts with a single character using INDEXFONT. The change is caused by the font switch sequence. The default is //. To simulate the XES model, redefine the sequence and assign the bold font to b as shown in this example.

```
<1B> SETFTSW
/p /NHE 12 INDEXFONT % normal font
/b /NHEB 12 INDEXFONT % bold font
```

Solid Coated and Uncoated Color Simulation

The list of predefined Colorkeys available for use in VIPP® jobs includes solid coated and uncoated color simulation as supported by FreeFlow Print Server.

There are more than 1000 predefined solid color simulation keys. Some color differences can be expected from device to device because color output is affected by many factors such as a printer color engine, settings, inks, and paper stock. To get the best possible representation of the solid colors, the Colorkey names in the .cck files have to match the color names used by the calibrated printer. When the Colorkey names match, the calibrated CMYK values on the printer override the CMYK values in the .cck files. A match can occur even when the .cck Colorkey name ends with an extra s character. Refer to Solid coated and uncoated custom colors in the *FreeFlow VI Compose User Guide*.

To select a solid color, use a solid Colorkey as with any of the predefined Colorkeys in `/usr/xgf/xgf.gep`.

Syntax

```
(solidcolorname) SETTXC
```

Example

```
(customColorkey) SETTXC
```

To see samples of the solid Colorkeys, print samccc.ps for solid coated color simulation or samccu.ps for solid uncoated color simulation. Both files are located in `xgf/demo`.

Specialty Imaging

Specialty Imaging is described in detail in Specialty Imaging with VIPP®, which is included in the *FreeFlow VI Compose User Guide*. Read Specialty Imaging chapter carefully, it provides the information you need to set up and use Specialty Imaging. It is important for you to pay particular attention to the Limitations described in that chapter. Some Specialty Imaging feature limitations are also covered in the descriptions of GLT and MPR.

Because these features limitations are dependant on multiple factors, such as media, print engine, calibration, color, font size, and font face, the results can vary from one system to another. Specialty imaging effects cannot be guaranteed to work on all devices and can exceed the device limitations for memory or processing capabilities. When this happens, the effects can degrade, show color shifts, or break down. Xerox highly recommends that you perform appropriate tests and validations before going into production.

Except for MicroText, Specialty imaging fonts can be printed on a white background. This is a requirement for gloss, correlation, and fluorescent effects. To use these effects over an area of color, including black, use the VIPP® DRAWB command to draw a white box under the area of the effect.

Specialty Imaging is supported on the following FreeFlow Print Server devices:

- All effects:
  - iGen
  - DC7000, DC8000
  - DC6060/5252/2060/2045
- Micro Text only:
  - Nuvera
  - 4110/4590 EPS
  - HLC devices (black text only)

GlossMark and Correlation Fonts (GL and CR)

The requirement to print the GlossMark or Correlation effect on a white background is met automatically. When a GL/CR effect is placed anywhere on a document, VIPP® automatically creates a white background under the effect, which is not visible to the end user. The action occurs automatically when a GlossMark font or a Correlation font that does not include the -L2- syntax in the font name is used. This allows the GL/CR effect to print anywhere in the design without the need to worry about underlying colors.

When a correlation font that contains the -L2 syntax in its name is used, VIPP® does not

automatically create a white background. Fonts with this syntax are considered a Layer two correlation font, normally used for the top layer in a two-layer correlation special imaging effect. In this case, VIPP® cannot draw a white box as the bottom correlation layer must interact with the top CR-L2 layer. Using a CR-L2 correlation font in a single layer effect is legal, but not recommended. When that option is used, an area of white on the document is selected, or a white box is drawn below the effect.

When creating a two-layer Correlation effect, it is important to use the CR font for the bottom layer and the CR-L2 font for the top layer because of the two-layer interaction of the effect. Failure to follow this order results in only the top layer effect that is visible.

When creating a Single Layer correlation affect, any CR font can be used. However, a CR-L2 font is not recommended. To create a Single Layer effect, select a white area on the page on which to print the effect, or draw a white box behind the effect. Printing a string with a CR-L2 font over a color area in the document will produce unpredictable results.

The following enhancements also apply when a GlossMark or Correlation font is used:

- the text is automatically padded with spaces up to the defined column width when using:
  - SHMF, SHMf, and SHmf with option +200
  - SHP and SHp with colwidth or fit-in-box syntax
- the text box is vertically padded with lines of blanks when using SHP fit-in-box and a spacing of 0 as in:

```
(string) [ width height 0 ] align SHP
```

Using Multi-byte (CJK) Fonts

VI Compose primarily relies on the PostScript interpreter to handle multi-byte printing. Any CID or OCF font supported by the PostScript interpreter is available to VI Compose.

However, it is important to note that SETENCODING does not apply to multi-byte fonts. Multi-bytes fonts are encoded using PostScript CMaps resources whose name is generally part of the font name (ex: /Ryumin-Light-90ms-RKSJ-H). For this reason multi-bytes fonts can never appear in SETENCODING font lists and the PostScript font name can be used directly with the **SETFONT** and **INDEXFONT** commands. Also the encoding of the data stream can match the encoding of the selected fonts.

All **SHx** commands can be used with multi-byte fonts.

Command syntax for SHP and SHMF is extended with selectable options for behaviors related to Asian languages. These behaviors also rely on encoding and special character lists defined in the `xgf/src/cjk.def` file using new commands such as SETCJKENCMAP, SETCJKRULES, SETV2HCONV and SETV2HTABLE.

Commands such as **FROMLINE**, **RPEKEY**, **GETFIELD**, **SETRCD**, **SETPCD**, **GETINTV** is adapted to work on character boundaries instead of byte boundaries when a multi-byte font is selected.

Note: Multi-byte characters cannot be used as variable names, field names of DBFs, or tag names of XML files.

Vertical Writing

All **SHx** commands can be used with multi-byte vertical fonts. In this situation all horizontal behaviors became vertical behaviors.

In particular:

- Left alignment becomes top alignment.
- Right alignment becomes bottom alignment.
- Horizontal centering becomes vertical centering.
- Horizontal justification becomes vertical justification
- Column width becomes column height.
- Line spacing is applied horizontally instead of vertically.
- Secondary horizontal position becomes secondary vertical position.

Instead of increasing the vertical print position the horizontal print position is decreased. Vertical text is forwarded from right to left.

Using PostScript Files with Embedded Fonts as Forms

When a PostScript file contains fonts, referencing the file as a form in VI Compose can have an affect on virtual memory (VM) usage and printer performance.

To avoid these problems, either permanently install the fonts on the printer disk drive and remove them from the form, or run the PostScript file through Decomposition Services on a DocuPrint NPS. This can be done in ByteCompressed mode. Then reference the form directly using the VIPP® SETFORM command in a JDT or DBM. For further information, refer to Decomposition services in the *FreeFlow VI Compose User Guide*.

Markers

Markers used in the VIPP® language are:

%	% %BoundingBox	% %PagesPerBooklet	[=name=]
% %	% %EOD_XGF	% %XGF	
%!	% %EOF	$$name.	

Miscellaneous Commands and Functions

Creating a VIPP® Self-Contained PostScript File

To print a VIPP® file to a PostScript device that is not VIPP®-enabled, you create a self-contained VIPP® file that is the concatenation of the VIPP® file and all of the resources required. This task requires two steps.

1. Determine all of the resources required for the VIPP® file.

   This can be obtained by running the job through the Demographics service. For details, refer to the *FreeFlow VI Compose User Guide*.

2. Concatenate all of these resources and relevant VIPP® core files with the VIPP® file.

   This implies the use of the **XGFRESDEF** and possibly **MAKEVMFILE** commands. For more information, contact a Xerox representative.

Incorporating Highlight Color in Decomposition Forms

Using Byte Compressed mode, **SaveMaskBC**, is recommended when using NPS Decomposition Services. This mode includes both black and highlight color in the same file and its performance is better than TIFF. The only concern is portability as this option is proprietary to DocuPrint NPS Printers.

Including a Non-VIPP® Object in a VIPP® Job

To include a page object, such as logos, graphics, charts, and so on, created by a foreign application in a VIPP® job, perform the following steps:

1. Save the object as EPS, TIFF, or JPEG.
2. Place the object in a directory that is referenced by **SETPPATH** (project mode) or **SETMPATH/ SETIPATH** (legacy mode)
3. Call the object in the VIPP® job with either:
   a. **CACHE/SCALL** to place the object at any location on the page.
   b. **CACHE/SETFORM/SETBFORM** if the object is a background form covering the entire page.

Printable Text or Reference ( )

Encapsulating parentheses ( ) identify printable text or references as a string operand for the next VIPP® command.

Printable text or references contain variable names that can be substituted with a value using the **VSUB** command.

Printable text or references can contain an octal character preceded by the \ character.

Syntax

```
(printable text or reference)
```

Examples

When the printable text or reference contains single left or right parentheses or a backslash (“\”), they preceded the backslash character as shown in these examples.

This example prints Delimiter is).

```
(Delimiter is \)) SHL
```

This example prints Delimiter is \.

```
(Delimiter is \\) SHL
```

This example prints Delimiter is ($).

```
(Delimiter is ($)) SHL
```

This example prints Xerox©. \251 is the octal value for the copyright symbol.

```
(Xerox\251) SHL
```

Printing With Special Characters

The encoding tables used by VI Compose are located in `/usr/xgf/encoding`. By default, **pcsun**, which is defined in `xgf.def`, is used and supports accented characters for both PC and Sun encoding. To use different encoding, modify the existing translation table or create a new translation table. For details, refer to *Standard lists, tables, keys, and attributes* in the *FreeFlow VI Compose User Guide*.

VI Compose and EBCDIC

VI Compose uses translation tables, which allow processing of VIPP® data streams coded using different character encoding schemes. Several tables that include the EBCDIC table have been provided. These tables can be modified or new ones created. Contact the system analyst for assistance that creates or modifies a table.

These tables are valid for printable data only, meaning that forms, JDTs, and so on, cannot be coded in EBCDIC. Commands included in the data file, such as VIPP® startup commands, **STARTLM**, and embedded commands, **%%XGFSETJDT**, can be coded in ASCII even when the surrounding data is coded in EBCDIC. However, submission from the host provides an option to convert EBCDIC to ASCII for output to the printer.

When using a DocuPrint NPS, you have the option of setting up a virtual printer that prepends the incoming data file with a VIPP® initialization file used to format the data files. For further information, refer to DocuPrint NPS XGFNub in the *FreeFlow VI Compose User' Guide*. This allows you to choose not to insert any VIPP® commands for use in startup formatting, but to send a customer data file in EBCDIC or ASCII format without any required modifications.

SETRCD and SETPCD reference strings must be coded in EBCDIC using the hexadecimal PostScript notation. This example looks for the EBCDIC string, TOTAL, in position 10:

```
/IF_TOTAL 10 5 /eq <E3D6E3C1D3> SETRCD
```

When imaging text with SH commands, both EBCDIC and ASCII re-encoded fonts can be used at the same time, for example:

```
[ /FontA /Helvetica ] (sun8) SETENCODING [ /FontE /Helvetica ] (ebcdic)
SETENCODING /FontA 12 SETFONT (ASCII text ...) SHL /FontE 12 SETFONT VARdata1
SHL % ebcdic text from a dbf field or GETFIELD
```

When imaging text with SH commands, both EBCDIC and ASCII re-encoded fonts can be mixed, for example:

```
/A /FontA 12 INDEXFONT /E /FontE 12 INDEXFONT (//A ASCII text //E$$VARdata1.
//A ASCII text again) 0 SHMF
```

Output Device Control

Page Control

Page control commands are:

ADVPAGE	GRIDSKIP	NEWFRONT	SETPBRK
BEGINIMP	NEWBACK	NEWSIDE	SKIPPAGE
ENDIMP	NEWFRAME	PAGEBRK	SLIPSHEET
GOTOFRAME			

Page control information covers the topic, Processing PCCs and VFUs.

Processing PCCs and VFUs

Use these commands in the JDT to define **PCC/VFU** processing:

```
/ANSI SETPCC
[/skip-key1 line-number1 /skip-key2 line-number2 ...] SETVFU
```

Page Layout

Page layout commands are

BEGINFRAME	ORIBL	SETGUNIT	SETUTAB
BTRIM	ORITL	SETLFI	SETZEBRA
COLW	PAGEH	SETLKF	SHEETH
FORMSHIFT	PAGEW	SETLSP	SHEETW
HPOS	PLINES	SETMARGIN	SHIFT
HPOS2	PORT	SETMAXBFORM	SLENGTH
IHEIGHT	PSIZE	SETMAXFORM	SSIZE
ILAND	SETBFORM	SETMULTIUP	TIFORI_off
IPORT	SETCOLWIDTH	SETPAGENUMBER	TIFORI_on
IWIDTH	SETFORM	SETPAGESIZE	TWOUP
LAND	SETFRAME	SETTAB	VPOS
LSP	SETGRID	SETUNIT	YINIT
ONEUP			

Page layout information covers these topics:

- Assigning Skip to Channel one to Various Top-of-Forms
- Cycle Form Capability in VIPP®

Assigning Skip to Channel one to Various Top-of-Forms

Assigning Skip to channel one to several lines (limited Multi-Up mode) is not currently supported by VI Compose. However, a similar result can be achieved using the SETMULTIUP command.

Cycle Form Capability in VIPP®

This is the cycle form syntax of the SETFORM command.

```
[ (form1) null null (form1) null null (form1) (form2) null null (form1)
(form2)] SETFORM
```

This example produces a 12-page cycle of forms where form1 prints on pages 1, 4, 7, and 11, form2 prints on pages 8 and 12, and neither form prints on pages 2, 3, 5, 6, 9, or 10.

Page Numbering

This is a two-step example of how to count subset pages in a job, which enables page numbering such as 1 of 2 or 15 of 99.

Place code similar to this example in the JDT called before STARTDBM:

```
2 SETMAXCOPY
{ IF CPCOUNT 1 eq
  { /VAR_pctot ++          % compute total pages for document
    SKIPPAGE               % skip pages on 1st pass
  } ENDIF
} BEGINPAGE
```

Add code similar to this example in the DBM at the point at which the first record of a new document (customer) is reached:

```
IF .. new customer ..
   { PAGEBRK                % push last page of previous customer
    2 SETCYCLECOPY          % init new copy cycle (includes implicit CHKPOINT)
    IF CPCOUNT 1 eq
      { /VAR_pctot 0 SETVAR }    % reset page total
    ELSE
      { (Page #/$$VAR_pctot.) VSUB 1 1 SETPAGENUMBER }
    ENDIF
   }
   ENDIF
```

Printing Data on Back Forms

To print variable data on the front and the back of a page with different forms on each side, perform one of these:

- Use the cycle form and cycle JDT syntax in a single JDT, as shown in this example. This is somewhat complex. However, it contains all of the required settings in one JDT and avoids duplication of common settings such as orientation or list of fonts.

```
%!
%%Title: jobxy.jdt
...
DUPLEX_on
...
[ (front.frm) (back.frm) ]  SETFORM

x BEGINRPE
  .....                     % RPE definitions for front page
ENDRPE  /RPEfront  INDEXRPE

y BEGINRPE
  .....                     % RPE definitions for back page
ENDRPE  /RPEback INDEXRPE

/JDTfront { RPEfront } XGFRESDEF
/JDTback { RPEback } XGFRESDEF

[ (JDTfront) (JDTback) ] SETJDT
```

This is the data file example.

```
%!
(jobxy.jdt) STARTLM
1page 1, line 1
 page 1, line 2
.....
1page 2, line 1
 page 2, line 2
....
%%EOF
```

- Create two separate JDTs, front.jdt, and back.jdt, and combine them at the start of the job as in this example. When JDTs are combined in this manner, do not use the DUPLEX_on command since it forces a NEWFRONT command.

```
%!
DUPLEX_on
  [ (front.jdt) (back.jdt) ] STARTLM
1page 1, line 1
 page 1, line 2
.....
1page 2, line 1
 page 2, line 2
....
%%EOF
```

In Multi-Up mode, you can create the data in this order assuming a three-up setting:

- front_page 1
- front_page 2
- front_page 3
- back_page 1
- back_page 2
- back_page 3
- front_page 4, and so on

Printing two Copies at Once Side by Side

To print two copies of each page of a document side by side, the document is output in this manner:

Sheet 1 / Side 1	Page 1 / Page 1
Sheet 1 / Side 2	Page 2 / Page 2
Sheet 2 / Side 1	Page 3 / Page 3
Sheet 2 / Side 2	Page 4 / Page 4
Sheet 3 / Side 1	Page 5 / Page 5
Sheet 3 / Side 2	Page 6 / Page 6

To achieve this, combine these commands:

- TWOUP
- 2 SETCYCLECOPY
- COLLATE_off

Trim the printed job to create two collated copies of the document.

Setting Multiple Forms in a Job

These examples can be used to set multiple forms in a job that:

- Use the planenumber option, which allows you to define several layers of forms, stacking forms on top of each other on the same page

```
(formA) 0 SETFORM
(formA) 1 SETFORM
(formA) 2 SETFORM
```

- Use the cycle form option, which allows you to define a list of forms applied in a cyclical manner on consecutive pages

```
[ (formA1) (formA2) (formA3) ] SETFORM
```

- Combine these options. This example specifies this information:
  - Page 1 receives forms formA1, formB and FormC
  - Page 2 receives forms formA2, formB and FormC

- Page 3 receives forms formA3, formB and FormC
- Page 4 receives forms formA4, formB and FormC, and so on

```
[ (formA1) (formA2) (formA3) ] 0 SETFORM
(formB) 1 SETFORM
(formC) 2 SETFORM
```

In database mode, new forms can be called after each PAGEBRK.

Examples

```
(formA) SETFORM
or
{ x y MOVETO (tiffA) 1 0 ICALL } SETFORM
......
PAGEBRK
(formB) SETFORM
.......
PAGEBRK
(formC) SETFORM
......
PAGEBRK
```

Setting Page Breaks in Line Mode

There are four ways to set up page breaks in a line mode job:

- Number of lines per page, SETGRID
- Form feed, hex 0C
- PCC byte
- % %XGF PAGEBRK

Form feed and PCC are mutually exclusive.

Form feed detection works with any string contained on the line. For further information, refer to SETPBRK.

Switching Between Multi-Up and One-Up

To print pages one and two on the front page in two-up mode, print page three on the back page in one-up mode, print pages four and five on the front page in two-up mode, print page six on the back page in one-up mode, and so on, enter this at the end of the JDT:

```
.......% common normal JDT code
/JDT1 { TWOUP } XGFRESDEF
```

```
/JDT2 { } XGFRESDEF
/JDT3 { ONEUP } XGFRESDEF
[ (JDT1) (JDT2) (JDT3) ] SETJDT
```

Define all of the common settings in the top portion of the JDT and then create the three subJDTs to be applied in a cyclical manner.

When you need to use different layout instructions for each of the three pages, define three RPE tables, index them using INDEXRPE, and then call the indexes in the sub-JDTs.

Using SETMULTIUP in Database Mode

When using SETMULTIUP in database mode, there are two situations to consider:

When you want each call to the DBM to print on the next logical page, it is important to enter the SETMULTIUP command at the top of the DBF file before the STARTDBM command. When you enter this command in the DBM, it reinitializes the Multi-Up sequence on each DBM call.

When you are coding a multi-page DBM with a Multi-Up layout, add a NEWSIDE or NEWFRONT at the beginning of the DBM or place the SETMULTIUP at the beginning of the DBM.

Using Tables to Store Data

There are situations where data can be accumulated into a table to be used later. This can be performed with a dedicated set of VIPP® commands:

Syntax for initializing the table:

```
/VARtablename [[ /VARname1 /VARname2 ... /VARnameN ]] SETVAR
```

Where:

/VARnameX	are the names of the table items.

Syntax for populating the table:

```
/VARtablename [[ name1 name2 ... nameN ]] ADD
```

Where:

nameX	are the variable names that contain the values for one entry in the table.

Syntax for sorting the table (optional):

```
VARtablename /VARkeyX /opt SORT
```

Refer to the SORT command for a complete description.

Syntax for updating the table (optional):

```
{ processing code UPDATE } VARtablename FOREACH
```

Refer to the UPDATE command for a complete description.

Syntax for processing the table:

```
{ processing code } VARtablename FOREACH
```

Refer to the FOREACH command for a complete description.

Examples

At the beginning of the JDT, before STARTDBM:

```
/VARaccountList [[ /VARname /VARacctNumber ]] SETVAR
```

In the DBM:

```
/VARaccountList [[ CustomerName AccountNumber ]] ADD
```

In ENDJOB:

```
VARaccountList /VARname /A SORT% sort alphabetically
100 3000 MOVETO (List of accounts:) SH
100 2800 MOVETO
{ VARname SH 1000 MOVEH VARacctNumber SH NL } VARaccountList FOREACH
PAGEBRK
```

Page Marking

Page marking commands are

ABSPOS	DRAWPIE	MOVEHR	SHJ & SHj
AZTEC	DRAWPOL	MOVETO	SHL & SH
BCALL	DRAWRDR	NL	SHMF, SHMf, SHmf
CLIP	ENDCLIP	OMRINIT	SHP and SHp
CUTMARK	ETCLIP	OMRSHOW	SHPATH
DATAMATRIX	FCALL	OTCLIP and ITCLIP	SHPOS
DEFINELAYOUT	FILLOMR	PDF417	SHR and SHr
DRAWB & DRAWBR	HDISP	QRCODE	SHT and SHt
DRAWBAR	ICALL	SAVEPP	SHX
DRAWBM & DRAWBRM	INDEXOTL	SCALL	SVPOS
DRAWC	IREVERSE_off	SETINDENT	TPATH
DRAWCRV	IREVERSE_on	SETLAYOUT	USPS4CB
DRAWPAR	MAXICODE	SETOTL	VDISP
DRAWPATH	MOVEH	SHC and SHc	

These page marking topics are covered:

Barcodes

The commands described in Transform Functions with the same names as common barcode fonts, are not barcode fonts, but are transforms that can manipulate an input string and output a string that a barcode font of that type would expect. Use the transform functions to avoid having to modify the existing application.

Except for the PDF417, DATAMATRIX, MAXICODE, and QRCODE barcodes, which are created as an image, all other barcodes supported by VI Compose, 2OF5, CODE39, EAN13/EAN8, CODE128/EAN128, UPCA, and POSTNET, require a barcode font to be called in order to print the barcode. Barcode fonts are not supplied as standard VIPP® fonts. To download fonts, refer to Font Download.

Barcodes that do not require data transforms can be printed by printing the text string with the appropriate barcode font. Other barcodes require that the input stream containing special characters. These special characters can be supplied in the text string by the application, or a transform function can be applied to the text string to add the special characters. The VIPP® command syntax includes several VIPP® commands that perform transforms on incoming data, so that when printed with a barcode font the correct barcode output is printed.

When the application does not supply these special characters in the string, using a transform avoids having to change the original application program, as the transform takes care of the barcode requirements. A barcode font is still required to print the correct character set. All Xerox transforms are based on barcode fonts, refer to Font Download.

For example, a typical USA postal code can be one of the following, 5, 9, or 11 characters, and can look like one of these examples:

- 90404
- 90404-2534
- 90404-2534+DPBC

To print this in the Postnet barcode format, you need:

- A PostScript Postnet font. Two postnet fonts, MB043 and MB045, are available for download, refer to Font Download.
- To supply the barcode font a string of acceptable characters. Just passing the string 90404-2534 is not going to work as the printed barcode can contain extra special characters not in the original string, such as frame characters, check digits, and so on. Without these characters, the barcode reader cannot understand the barcode passed to it and rejects it as an error.

Getting the String

Get an acceptable string in one of the following two ways:

- The program that generates the data can supply the complete string that needs to be passed to the barcode font, including all the special characters.
- Use the VIPP® barcode transform commands. These special VIPP® commands enable you to pass the standard text string to the transform command. The transform command modifies the input string and add the required special characters, check digits, and so on, as required, assuming the original input string is valid and the barcode font you are using is a supported barcode font - VI Compose supports fonts supplied by Xerox, refer to Font Download.

To use the VIPP® POSTNET transform command, you cant:

- Call in the Postnet font
- Pass the font the standard postnet code, use the VIPP® POSTNET command to transform the string, and print it using any of the **SHx** type commands

Examples

These examples call in the Postnet font:

```
/MB034 12 SETFONT
(90404-2534) POSTNET SH % with the dash
/MB034 12 SETFONT
(904042534) POSTNET SH % without the dash
/MB034 12 SETFONT
($$ZIP.) VSUB POSTNET SH % if calling in a variable
```

This example uses a font index created for the MB034 font:

```
/F1 /MB034 12 INDEXFONT % Postnet font
F1 (90404-2534) POSTNET 0 SHP % with the dash
```

This example enables a barcode transform using an RPE entry. In the example below the alignment parameter is replaced with a procedure body that contains the POSTNET transform and the SH command:

```
/F1 /MB034 12 INDEXFONT % Postnet font
5 BEGINRPE
...
10 FROMLINE
[ { POSTNET SH } 0 Xinit Xdisp Yinit Ydisp 11 9 /F1 BLACK ]
..
ENDRPE
```

Dynamic TIFF File Placement in Line Mode

Use one of these methods to dynamically call TIFF files in line mode:

- Use the **ICALL** command in an NMP:

  ```
  %%XGF 1000 1500 MOVETO (name.tif) 1 90 ICALL
  ```

  The image is located at a fixed position regardless of where the NMP occurs in the data file.

- Use **ICALL** in an RPE entry align procedure to achieve dynamic placement depending on whether the image name is fixed or must be extracted from the data:

  ```
  [ {1 90 ICALL} 0 Xinit Xdispl Yinit Ydispl 0 (xxxxx.tif) /font color ]
  ```

  or:

  ```
  [ {1 90 ICALL} 0 Xinit Xdispl Yinit Ydispl rec_pos length /font color ]
  ```

In either case, the line using this RPE entry must be identified with a test, refer to SETRCD or using an RPE prefix.

ICALL in a JDT

When you attempt to make a stand-alone ICALL in a JDT, errors occur. Use one of these to make an ICALL in a JDT:

- An in-line VIPP® form, as shown in this example. In this example, the VIPP® form file is placed as plane 0 and the TIFF image is placed as plane 1.

```
2 SETMAXFORM

(myform.frm) 0 SETFORM

{ PORT ORITL 0100 0100 MOVETO (focus.tif) 1 0 ICALL } 1 SETFORM
```

Make the ICALL part of an RPE definition, as shown in this example. In this example, a constant value, image.tif, is used. However, the image file name can be obtained from the data file using the recpos and length parameters to find the name in the record. Refer to FROMLINE and ICALL for further information on how to use these commands in an RPE definition.

```
3 BEGINRPE

[{1 0 ICALL} 0 1400 0 1495 0 000 (image.tif) /TEXTS BLACK]

....

ENDRPE
```

There may be unexpected side effects to marking commands such as SHX, ICALL, and SCALL in a JDT, when you code DIRECT calls.

A JDT should contain only general or global formatting commands for the job. A marking command writes directly on the current page. However, when the JDT is executed in the initialization phase, no current page is defined.

Locating TIFF Images in Multiple Directories

ICALL builds the full path to an image by concatenating the library paths specified in the SETIPATH command with the image name provided as the argument. The default libraries are available in the `/usr/xgf/src/xgfunix.run` file.

Note: Simplify VIPP® resource management

Using VI Projects simplifies VIPP® resource management. Refer to *VI Projects* in the *FreeFlow VI Compose User Guide* for more details.

Do not mix VI Project and VIPP® legacy resource management

VI Project and VIPP® legacy (imglib, formlib, fontlib, jdtlib, etc.) resource management should not be mixed in a single job.

Perform one of these to access TIFF images in multiple directories:

- Place additional libraries in xgfunix.run. Make sure that no two images in any two libraries have the same name. When this occurs, only the first image can be accessed.
- Include a **SETIPATH** command at the beginning of the job to specify which libraries to search.

```
(/disk330/tiff1/) SETIPATH
.....
(image1) 1 0 ICALL
```

- Include an empty SETIPATH at the beginning of the job and specify the full path with each ICALL.

```
() SETIPATH
.....
(/disk330/tiff1/image1) 1 0 ICALL
```

'()' SETIPATH disables all of the libraries defined in `/usr/xgf/src/xgfunix.run` for the current job and provides an empty path. When this syntax is used, no search is performed. Therefore, access to the image is direct for each ICALL.

To have access to the other libraries in the job, enter "()" as the first library in `/usr/xgf/src/xgfunix.run`. The effect is similar, as the image containing a full path is located in the first library; therefore no further search is necessary. In addition, use the base name of the images to continue to access images located in other libraries.

```
[ ()
(/usr/xgfc/imglib/)
(/usr/xgf/imglib)
] SETIPATH
```

- Specify a root path using SETIPATH and enter an additional subpath in the ICALL. Specify the root path in xgfunix.run.

```
(/disk330/) SETIPATH.....(tiff1/image1) 1 0 ICALL
```

Keeping the path, or at least the root path, out of the job may avoid having to make changes to the application later should changes occur in the file system structure (for example, adding new disks, renaming disks, or mounting on a DOS file system).

Main and Secondary Print Positions

Main and secondary print positions (PP) refer to the page location (X,Y) used to place text, images, or segments by the commands detailed below.

Vertical (Y) main and secondary print positions are always identical, and are referred to as the vertical print position. Only horizontal (X) main and secondary print positions differ. Main and secondary print positions are defined as follows:

- Main PP = Horizontal main PP + Vertical PP
- Secondary PP = Horizontal secondary PP + Vertical PP

These commands are used to set or adjust main and secondary print positions, or to place text or images in a VIPP® job:

- MOVETO sets the initial main and secondary print positions to the same values.
- SHC and SHc, SHJ and SHj, SHL and SH, SHMF, SHMf, and SHmf, SHP and SHp, SHR and SHr, SHT and SHt, and SHX use the current main print position to:
  - place text
  - reset the horizontal main PP to the last MOVETO value
  - increase the vertical PP downward by the SETLSP value
- SHL and SH, SHC and SHc, SHJ and SHj, SHMF, SHMf, and SHmf, SHP and SHp, SHR and SHr, and SHT and SHt use the current secondary PP to place text and set the secondary PP to the position reached after the text is printed.
- ICALL and SCALL use the current secondary PP to place images or segments and leave the secondary PP unchanged, regardless of the size of the image or segment.
- MOVEH sets the secondary horizontal PP to a supplied value.
- MOVEHR sets the secondary PP relative to the last horizontal main position.
- NL resets both the main and secondary horizontal PP to the last MOVETO value and increases the vertical PP downward by the SETLSP or supplied value.
- PAGEBRK resets the main and secondary PP to 0,0.
- SAVEPP saves the current secondary PP for later use by HDISP, VDISP, SHPOS, and SVPOS.

Merging Data with TIFF Files

To merge a TIFF file with data, use one of these options:

- Create a form (`/usr/xgfc/formlib/xyz.frm`) with these contents:

  ```
  %!
  { x y MOVETO (ab1.tif) 1 0 ICALL } FSHOW
  ```

  Then call the form in the JDT as follows:

  ```
  (xyz.frm) SETFORM
  ```

- Build the form in the JDT as follows:

  ```
  { x y MOVETO (ab1.tif) 1 0 ICALL } 0 SETFORM
  ```

Note: In the above example, x and y are the origin of the top left corner of the image.

- Reference the TIFF file in an RPE entry. Refer to the align procedure option in the FROMLINE extensions.
- Call the TIFF file in an ENDPAGE procedure.

Printing a Shaded Block of Text

The code example below shows how to create a shaded address block that grows up or down depending on the number of lines in the block of text.

The file prints an address block that is similar to this example:

```
Mr David Kirk
Xerox Corporation
Suite xyz
El Segundo, CA 90245
```

In this example the address block is printed over a shaded background, and the background width is defined by a **SETCOLWIDTH** command.

```
%%Title: batshade.dbm
%%Creator: VI Designer DataBase Mode.
%%CreationDate: 08-24-1999.
%%This sample prints a form letter from database fields.

DOT3 SETUNIT
PORT
/NTMB 20 SETFONT
90 SETLSP

% See SETBAT for SETBAT parameters
% This command sets the background attribute
/DAVEBATkey [XLTR 0  /COLW'+'40 0   /LSP 0  -20 2  -.4 1  0 0   0] SETBAT

% Set the background attribute ON
/DAVEBATkey SETTXB
800 SETCOLWIDTH 175 2910 MOVETO
($$Title.$$Fname. $$Lname.) VSUB 0 SHP
Addr1 0 SHP
Addr2 0 SHP
City  0 SHP
($$State. $$Zip.) VSUB 0 SHP
% Set the background attribute OFF
null SETTXB

175 2254 MOVETO
(Start letter here ...) SH


PAGEBRK
```

Suppressing Vertical Displacement with Empty Fields in a DBM

Use one of these to suppress the vertical displacement when a field in a DBM is empty:

- Use SHP and SHp, as it suppresses the vertical displacement if the string is empty (contrary to other SHX commands):

```
1600 800 MOVETO
($$FIRST_NAME. $$LAST_NAME.) VSUB SHL
ADD1 SHL
ADD2 0 SHP
($$CITY_NAME. $$STATE. $$ZIP.-$$ZIP4.) VSUB SHL
```

In this example, if ADD2 is empty, there is no blank space after ADD1. However, use SHP only when necessary, as it requires more processing than other SHx commands.

- Use IF/ELSE/ELIF/ENDIF:

```
1600 800 MOVETO
($$FIRST_NAME. $$LAST_NAME.) VSUB SHL
ADD1SHL
IF ADD2 () ne { ADD2 SHL } ENDIF
($$CITY_NAME. $$STATE. $$ZIP.-$$ZIP4.) VSUB SHL
```

TIFF File Performance Considerations

When you have several TIFF images to include in a form, call these TIFF images from the form, rather than as separate forms in the JDT. When the TIFF files and the form are static throughout the document or are static in a pattern in the document, as with a cycle form, run them as a separate job through Decomposition Services and reference the resulting file as a single form.

You can achieve the best performance from the printer when TIFF files do not require any scaling or rotation and when the file size is kept to the minimum. Reduce file size in a number of ways, including cropping excess white space from a scanned image.

Although VIPP® can reference TIFF files not sent with the submission file, you can evaluate the network performance when you perform this task, as the files could reside on a network server and could be shared by multiple printers on the same network.

PDF Interactive Features

PDF control commands are:

BOOKMARK	PDFINFO
PDFDEST	PDFOPEN
PDFDEVICE	SETPIF

VIPP® PDF Interactive Features (PIF) are a set of VIPP® commands that allow the designer of a VIPP® job to create interactive elements when the VIPP® job is rendered into a PDF document.

There are three categories of interactive elements:

- Bookmarks
- Links
- Notes

Bookmarks

A list of bookmarks is an optional feature of a PDF document that is displayed on the screen. The list consists of a tree-structured hierarchy of text items. Clicking the text of a bookmark causes the viewer application to jump to its associated destination, which is a specific part either of the current document or another document.

Links

A link associates an element on a page such as a fragment of text, a box, or an image with a destination. Clicking this element causes the viewer application to jump to the destination.

Notes

A note associates an element on a page such as a fragment of text, a box or an image with a text entry that is not part of the page flow. When the note is closed, the text is not visible. A closed note can be invisible or visible as an icon or a stamp. Clicking a visible note, or an element associated with an invisible note, causes the text to be displayed in a pop-up window. There are four categories of notes:

Icon

A visible note represented by a small icon.

Stamp

A visible note represented by a rubber stamp.

Free text

A visible note that displays text directly on the page. Free text notes have no closed state; instead of being displayed in a pop-up window, the text is always visible.

Markup

Markup is visible as background color underneath the associated closed element if a color other

than white or null is specified, otherwise the note is invisible.

Predefined Keys and Keywords

Refer to Standard lists, tables, keys, and attributes in the *FreeFlow VI Compose User Guide* for an overview of the predefined keys and attributes used in VIPP® programming. These additional subjects are covered here:

- Adding GEPkeys
- Improving Shading Differentiation Using DDGs
- VIPP® Colorkeys
- Reserved Keywords

Adding GEPkeys

The most effective manner in which to create a new Colorkey or GEPkey is to print `/usr/xgf/demo/palrgb.ps` using the correct highlight color.

To perform this task from the produced page, locate the colors you want and note the three values that specify each color. These color values are located on top of the box. Edit the `/usr/xgf/src/xgf.gep` file and add a Colorkey that specifies these values.

Then add all of the GEPkeys you want to use with this Colorkey.

Improving Shading Differentiation Using DDGs

Changing the Colorkey set improves the shading differentiation on black and white printers when you are using the **DDG** commands DRAWPIE, DRAWBAR, or DRAWCRV. The default Colorkeys are defined by the parameter /ColorTable. To change the default Colorkeys, use the **SETPARAMS** command to specify a different set of Colorkeys. Mixing the sequence of gray levels from light to dark is recommended.

Examples

This example sets seven levels of gray for use with DDGs.

```
[/ColorTable[XLIGHT MEDIUM XDARK LIGHT DARK LMEDIUM DMEDIUM]]SETPARAMS
```

For more colors, add Colorkeys to xgf.gep. The maximum number of colors that can be defined in the /ColorTable is 65,536.

Reserved Keywords

When a reserved keyword is used as a database field name unpredictable and hard to isolate errors can occur in VIPP® jobs. In order to avoid this conflict, VI Compose performs a check on database names and produces an error message when a conflict is found. The additional line of code checks for database file names that are known in the current PostScript context. Reserved keywords include all:

- PostScript operators and reserved keywords
- VIPP® commands, functions, variables, etc.
- VIPP® internal names (starting with XGF, VIP, ^, and ")

The error message (VIPP_ambiguous_name in _name) can also occur in an XML job when the code uses an ambiguous XML variable name, which is not adequately qualified for VI Compose to identify

to which part of the XML tree it refers.

To avoid using reserved keywords when naming database fields, use an initial uppercase character in the database field name. For example, Fieldname.

VIPP® Colorkeys

VI Compose uses a built in color definition table located in the `/usr/xgf/src/xgf.gep` file, which contains Colorkey definitions.

A Colorkey definition is a set of color values, such as gray, RGB, CMYK, pattern, or gradient, that is linked to a color name. Selecting a Colorkey name actually selects the RGB or CMYK value for that key.

Custom colors can be added to this table using the syntax described in the SETCOL command.

When added to the xgf.gep table, the definition can be permanent and available to all applications. When added to a DBM, JDT or XJT, the color definition is temporary and available just for that job.

A Colorkey can be referenced in numerous VIPP® commands SETTXC, SETGEP,SETBAT, SETFRAME, SETZEBRA. In these commands, the Colorkey can be used as is, or preceded by a slash character.

For more information about using fonts and color, refer to Fonts and Colors.

Examples

In an RGB color definition, the RGB color is defined as a percentage of Red, Green and Blue. The color definition below defines an RGB color (only three values entered in the array). The three values represent 100 % Red, 0 % Green and 0 % Blue. The label RED has been assigned to this definition

```
/RED [1 0 0] SETCOL
```

This color definition defines an RGB color. The three values represent 100 % Red, 90 % Green and 90 % Blue. The label XLRED has been assigned to the definition.

```
/XLRED [1 .9 .9] SETCOL
```

In a CMYK color definition, CMYK color is defined as a percentage of Cyan, Magenta, Yellow and Black. This color definition defines a CMYK color (four values entered in the array). The values represent 0 % Cyan, 100 % Magenta and 90 % Yellow and 0 % Black. The label CMYK_RED has been assigned to the definition.

```
/CMYK_RED [0 1 .9 0] SETCOL
```

This is an example of Colorkey usage in a VIPP® command. In the example the statement sets the color for all subsequent printed text to the Colorkey value called CMYK_RED as defined in the example above.

```
CMYK_RED SETTXC
```

Print File Processing

Print file processing commands are:

BIDI	ENDARBM	PPCOUNT	STARTLM
BACKSP_off	ENDARBT	PROCESSDJDE	STARTXML
BACKSPF_on	ENDBOOKLET	QSTRIP_on	TRIO
BACKSPP_on	ENDJOB	SETBIDI	XGFEND
BEGINARBM	ENDXPD	SETBUFSIZE	XMLATL
BEGINARBT	ETA	SETDBSEP	XMLATN
BEGINXPD	ETS	SETDLFILE	XMLATV
BOOKLETRANGE	EXIT	SETJDT	XMLDTH
BSTRIP_off	FRCOUNT	SETLMFILE	XMLPAR
BTA	FRLEFT	SETNMP	XMLPATH
BTS	INDEXPIF	SETPARAMS	XMLSYN
DJDEBEGIN	LMSKIP	SOF_off	XMLTAG
DJDECMD	OVERPRINT_on	STARTBOOKLET	XMLVAL
DJDEPAR	PAGERANGE	STARTDBM	ZSORT

These Print file processing command functions are discussed:

- Booklet Support
- Media Support
- ZSORT and Record Grouping

Booklet Support

Booklet mode is driven by using the STARTBOOKLET and ENDBOOKLET commands. Initialize booklet mode using this SETPARAMS sequence:

```
[ /PagesPerBooklet integer1
/BookletMismatch integer2
] SETPARAMS
```

Where:

integer1	is the number of pages that a booklet should contain.
integer2	is used to control the action in **ENDBOOKLET** when the number of pages does not match: **0** ignores mismatch and continues **1** adds blank pages until the number of pages match, and continues **2** aborts the job with a VI Compose error message (done through the PostScript error mechanism)

This sequence is coded at the beginning of the DBF or in the JDT.

BOOKLETRANGE works in conjunction with STARTBOOKLET and ENDBOOKLET to select a range of booklets to print.

Note: Not all DFEs (printer controllers) support the **STARTBOOKLET** and **ENDBOOKLET** commands. They are currently supported on EFI and FreeFlow Print Server. However, the finishing and imposition features associated with a booklet delimited by these commands depend on the DFE used. For instance, currently, FreeFlow Print Server can apply subset finishing such as staple, bind, and so on. in conjunction with a **SETFINISHING** command added before STARTBOOKLET.

Verify support and printer software levels before using these commands.

Media Support

In the VIPP® language, media selection is available through the **SETMEDIA** and **SETMEDIAT** commands. These commands default to the PostScript operator, setpagedevice, to perform the appropriate media selection. A workaround is available for those devices on which setpagedevice is not implemented. The /MediaSubst parameter can be set on those devices to substitute a proprietary PostScript sequence for the default selection.

Examples

This is an example of usage and syntax for media substitution:

```
[/MediaSubst<<
(Plain) { 1 setpapertray }
(Letterhead) { 2 setpapertray }
>>]SETPARAMS
```

When setpapertray is implemented on the device, the VIPP® sequence, (Letterhead) SETMEDIA can cause the paper to be fed from tray 2.

Place the /MediaSubst definition in the `/usr/xgf/src/xgf.def` file on the device controller so that it can be easily changed when installing new media on the device.

ZSORT and Record Grouping

ZSORT replaces record grouping as a way to set up database or Multi-Up or duplex jobs.

The initial intent of record grouping was to be able to split very large records into several sub-records to bypass spooler limitations that do not accept long records. Instead, this functionality is most often used in database or Multi-Up or duplex jobs to read N records at a time, matching the N-up and to print variable data from each record on the front and on the back of the Multi-Up pages.

This complex setup is now obsolete and can be easily replaced by placing a **ZSORT** statement with stacksize=1 before **STARTDBM** or in the associated JDT. Then record grouping is removed keeping only one set of field names and the DBM simplified by keeping only the lines of code for one record.

The parameter, /MUPduplex, is related to Multi-Up placements of logical pages on the back. When /MUPduplex=1, 0 is the default for backward compatibility but **ZSORT** sets it to 1 automatically, logical page positions on the back side are computed so that they physically face their counterparts on the front. Therefore, you do not need to worry about computing different placements on the front and on the back even with odd rotations. Logical pages fits entirely on the physical page either through explicit specification of the logical page sizes or by scaling down meaning that when the Multi-Up scale=1 use the **SETMULTIUP** syntax that specifies the page size for each logical page to avoid incorrect placements on the back.

For example, when defining a four-up portrait on a letter page, 2550 x 3300 assuming DOTS units, the logical page size is 1275 x 1650 so the Multi-Up statement is:

```
[ [ 0 1275] [ 0 1650] 0 1 1 [1275 1275] [ 0 1650] 0 1 1 [ 0 1275 ]
 [ 1650 1650] 0 1 1 [1275 1275] [ 1650 1650] 0 1 1 ] SETMULTIUP
```

Printer Carriage Control

PCC commands are:

- **BEGINPCC**
- **SETSKIP**
- **ENDPCC**
- **SETVFU**
- **SETPCC**

Resource Control

Resource control commands are:

- CACHE
- EXIST
- FBIND
- MAKEVMFILE
- MAKEVMFORM
- PRECACHE
- SETEPATH
- SETFPATH
- SETIPATH
- SETJPATH
- SETMPATH
- SETPAT
- SETPAGEDEF
- SETPPAT
- SETPPATH
- SETPROJECT
- STOREVAR
- SUBSTFONT
- XGFRESDEF

RPE Items

Record Processing Entry commands are:

BEGINRPE	FROMLINE	RPEDEF	SETRPE
CURLINE	INDEXRPE	RPEKEY	SETRPEPREFIX
ENDIFALL	NEWGROUP	RPEPOS	
ENDRPE	NEWPOS	SETRCD	

RPE items discussed here include:

Ignoring the Number of Lines at the Beginning of Each File

When the contents of the lines is unique in the SETLMFILE file, a SETRCD condition can be defined and used in a FROMLINE entry with recpos=0 and length=0. Also, use the extended syntax of STARTLM, which allows you to use a JDT for a specific number of pages and then to switch to another JDT for the remainder of the file.

When this is the case, define SETGRID as the number of lines you want to skip in the first JDT and use a **FROMLINE** command that selects no data.

Overprinting a Line for a Bold Font Effect

Some legacy applications designed for impact printers still use the overprint technique to obtain a bold text effect. To obtain the same effect with VIPP®, use the **SETRCD** command to test the overprint code, usually a + PCC character at position 0 of the record, and apply a different print position, 104 and 104 rather than 100 and 100. This prints the two lines on top of each other with a slight offset and provides the bold effect.

This is an example:

```
%!
XGF
/ANSI SETPCC
/F1 /NCR 12 INDEXFONT
/IF_OVP 0 1 /eq (+) SETRCD

10 BEGINRPE
1 FROMLINE
  /IF_OVP [ 0 0 104 null 104 50 0 80 /F1 BLACK ]
  /ELSE   [ 0 0 100 null 100 50 0 80 /F1 BLACK ]
ENDRPE

() STARTLM
 Aaaaaaaaaaaaaa
 Bbbbbbbbbbbbb
+Bbbbbbbbbbbbb}
 Ccccccccccccc
%%EOF
```

RPE Command Information

Record Processing Entry (RPE) is a presentation option used in line mode applications that allows each record to be split into fields that can be printed any number of times and at any location on the page. This option is also described in VIPP® data streams in the *FreeFlow VI Compose User Guide*.

RPEs can contain specific presentation attributes such as position on the page, line spacing, font, color, alignment, and rotation. RPE attributes are initiated by a STARTLM sequence.

The RPE fields are defined horizontally using one of these:

- character position and field length
- field number

The RPE fields are defined vertically using one of these:

- line number FROMLINE
- prefix SETRPEPREFIX, RPEKEY

Field definitions, location on the page, and presentation attributes for a given group of lines or a given prefix are coded in an RPE definition. For more information, refer to RPEDEF and SETRPE.

Page delimiting is controlled by setting a maximum number of LPP or using an explicit page delimiter such as Form Feed (FF) or Skip to channel one when PCC is used. For more information, refer to SETGRID, SETPCC, and SETPBRK.

Line printer files built with one data record per page (referred to as unformatted records) can also be processed, one line per page, using RPE commands.

Extending FROMLINE and RPEKEY Commands

The FROMLINE and RPEKEY commands can be extended by using:

- Conditional processing
- Fixed text
- Align procedure

Conditional Processing

Conditions can be nested at any level using the /ENDIF and /ENDIFALL RPE subcommands. For further information, refer to SETRCDSETPCD condition definitions.

Examples

In this example, /ENDIFALL provides a facility to close all pending /IF statements in one command rather than having to code all matching /ENDIF statements.

```
1 FROMLINE
        /IF_CND1
                [....  rpe entry 1  .... ]
                [ ....  rpe entry 2  .... ]
               /IF_CND2
                        [ ....  rpe entry 3  .... ]
               /ELSE
                        [ ....  rpe entry 4  .... ]
               /ENDIF
         /ELSE
                [ ....  rpe entry 5  .... ]
                [ ....  rpe entry 6  .... ]
               /IF_CND3
                        [ ....  rpe entry 7  .... ]
               /ENDIF
        /ENDIF

10 FROMLINE
        /IF_CND4
                [ ....  rpe entry 11  .... ]
                [ ....  rpe entry 12  .... ]
        /ELSE /IF_CND5
                [ ....  rpe entry 13  .... ]
        /ELSE /IF_CND6
                [ ....  rpe entry 14  .... ]
        /ELSE /IF_CND7
                [ ....  rpe entry 15  .... ]
                [ ....  rpe entry 16  .... ]
        /ELSE
                [ ....  rpe entry 17  .... ]
        /ENDIFALL
```

The previous syntax is still valid for /IF and /ENDIF matching one RPE entry respectively. The new syntax is applied only when /ENDIF or /ENDIFALL is located in an RPE definition.

Old and new syntaxes are exclusive in an RPE definition. However, RPE definitions with both syntaxes can be mixed in an RPE library composed of several RPEKEY or FROMLINE commands placed between BEGINRPE and ENDRPE.

```
2 BEGINRPE          % Begin RPE library

1 FROMLINE          % RPE using old syntax
/CND1
       [ .... rpe entry .... ]
/ELSE
       [ .... rpe entry ... ]

2 FROMLINE          % RPE using new syntax
/IF_CND2
       [ .... rpe entry .... ]
       [ .... rpe entry .... ]
/ELSE /IF_CND3
       [ .... rpe entry .... ]
/ELSE /IF_CND4
       [ .... rpe entry .... ]
/ENDIFALL

ENDRPE               % End RPE library
```

Fixed Text

The recpos and length parameters in an RPE definition for **RPEKEY** or **FROMLINE** commands can be replaced by one of these sequences:

- 0 (fixed text)
- 0 /VARname

Use this feature to print fixed text strings or variables, which are defined using SETVAR or GETFIELD, and are not present in the data stream.

Examples

This example prints Description: at Xpos 70 followed by characters 0 to 29 in the record at Xpos 500 and Ypos 910.

```
/LFA0 RPEKEY
[ 0 0 70 null 910 50 0 (Description:) /F1 BLACK]
[ 0 0 500 null 910 50 0 30 /F1 BLACK]
```

This is an example using a variable.

```
/VAR.LABEL1 (Description:) SETVAR
....
/LFA0 RPEKEY
[ 0 0 70 null 910 50 0 /VAR.LABEL1 /F1 BLACK]
```

Align Procedure

The align parameter in an RPE entry can be replaced by a VIPP® native mode procedure. This procedure calls a native mode command using printable data or fixed text as an operand.

When the procedure is called, the operand can be provided automatically by the extracted field specified by the recpos and length parameters or fixed text feature described above. The print position is set according to the current RPE print position.

The procedure can only supply the additional operands required by the VIPP® command. These are examples of align procedure syntax:

- { rotate GEPkey align SHX }
- { colwidth align SHP and SHp }
- { colwidth align SHMF }
- { scale rotation ICALL }
- { scale SCALL }
- { EAN13 SHX }
- { CODE39 SHX }, SHx represents SH, SHr, SHc, or SHj.

Examples

```
{ DATAMATRIX }
```

```
{ /VARqrc /SWP SETVAR [ /AC VARqrc ] QRCODE }
```

This is an example that prints a TIFF image called signa.tif at Xpos 70 and Ypos 910 every time /LFA0 RPEKEY is selected. In addition, characters 0 to 29 in the record is printed at Xpos 500 and Ypos 910.

```
/LFA0 RPEKEY
```

```
[{1 0 ICALL} 0 70 null 910 50 0 (signa.tif) /F1 BLACK]
```

```
[ 0 0 500 null 910 50 0 30 /F1 BLACK]
```

This example prints the string David Kirk underlined at position 100 100. The N turns off the UNDL BATkey.

```
/U /UNDL INDEXBAT
/N null INDEXBAT
1 BEGINRPE
1 FROMLINE
[ {U SH N} 0 100 0 100 30 0 (David Kirk) /F1 BLACK ]
```

Using Highlight Color to Print Negative Numbers

The SETRCD and SETPCD commands have a /HOLD test operator used to locate the compare string in the record portions of the data.

This example looks for the minus sign in the field and uses the current highlight color to print it:

```
/IF_NEG 50 10 /HOLD (-) SETRCD
```

```
.....
```

```
x FROMLINE
```

```
/IF_NEG [0 0 100 100 100 100 50 10 /F1 HCOLOR ]
```

```
/ELSE [0 0 100 100 100 100 50 10 /F1 BLACK ]
```

```
.....
```

Note: The example assumes that the number is 10 bytes long in position 50.

Transform Functions

Transform function commands are:

2OF5	**CODE39**	**GETINTV**	**REPLACE**
64TO256	**CS**	**HMS**	**UPCA**
AZTEC	**EAN128**	**NOHYPHEN**	**UPC-A**
BSTRIP	**EAN13/EAN8**	**POSTJPN**	**UPC-E**
CASELOW	**F2S**	**POSTNET**	**VSUB**
CASETI	**FORMAT**	**QSTRIP**	**VSUB2**
CASEUP			**VSUB3**
CODE128			