
**Documaker**

# Docutoolbox Reference

version 11.2

**Skywire Software, L.L.C.**
**3000 Internet Boulevard**
**Suite 200**
**Frisco, Texas 75034**
**www.skywiresoftware.com**

| | | |
|---|---|---|
| **Phone:** | **(U. S.)** | **972.377.1110** |
| | **(EMEA)** | **+44 (0) 1372 366 200** |
| **FAX:** | **(U. S.)** | **972.377.1109** |
| | **(EMEA)** | **+44 (0) 1372 366 201** |
| **Support:** | **(U. S.)** | **866.4SKYWIRE** |
| | **(EMEA)** | **+44 (0) 1372 366 222** |
| | | **support@skywiresoftware.com** |


## PUBLICATION COPYRIGHT NOTICE



## TRADEMARKS



## SOFTWARE COPYRIGHT NOTICE AND COPY LIMITATIONS



## PUBLICATION COPY LIMITATIONS



## DISCLAIMER

The contents of this publication and the computer software it represents are subject to change without notice. Publication of this manual is not a commitment by Skywire Software or its subsidiaries to provide the features described. Neither Skywire Software nor it subsidiaries assume responsibility or liability for errors that may appear herein. Skywire Software and its subsidiaries reserve the right to revise this publication and to make changes in it from time to time without obligation of Skywire Software or its subsidiaries to notify any person or organization of such revision or changes.

The screens and other illustrations in this publication are meant to be representative, not exact duplicates, of those that appear on your monitor or printer.

# Contents

## Utility Reference

# Utility Reference

This manual contains information about the various Docutoolbox utilities you can use.

Refer to the Finding the Right Utility on page 2 to quickly locate the utility you want to use. This list includes a short description of each utility.

The remainder of this manual describes each utility, discusses the parameters you can set, and provides examples.

The utilities are discussed in alphabetical order.

**NOTE:** The name *Docutoolbox* refers to the tools you use to create the form sets and resources required for your Documaker solution. This manual discusses the command-line utilities that are a part of Docutoolbox.

## Finding the Right Utility

Use this table to locate the utility you want more information about. All utilities are discussed on the following pages in alphabetical order.

| To... | Use... |
|---|---|
| Add missing header information for Xerox fonts | ATPHDR on page 46 |
| Back up an archive file | ARCSPLIT on page 39 |
| Check the integrity of a CAR file | CARINTEG on page 51 |
| Check a font cross-reference (FXR) file for settings which would cause problems when generating PDF files. | FXRVALID on page 118 |
| Compare two font cross-reference (FXR) files | FXRCMP on page 116 |
| Compile FAP files into AFP overlays | FAP2OVL on page 78 |
| Compile FAP files into overlay files | OVLCOMP on page 182 |
| Compile FAP files into Xerox Metacode FRM files | FAP2FRM on page 71 |
| Compile FAP files into Xerox Metacode print files | FAP2MET on page 73 |
| Convert a database into another database | DB2DB on page 62 |
| Convert a BDF file into a pre-version 11.x format FORM.DAT file | BDF2FDT on page 50 |
| Convert a FAP file into a PDF file | FAP2PDF on page 80 |
| Convert a FAP file into an RTF file | FAP2RTF on page 81 |
| Convert a CompuSet file into a FAP file | CSET2FAP on page 57 |
| Convert a FORM.DAT file into a database file | FDT2DB on page 84 |
| Convert a FORM.DAT file into an EDL file | FDT2EDL on page 89 |
| Convert a FORMDEF and SETRCPTB pair into BDF, GRP, and FOR files | CVTFASR on page 60 |
| Convert a non-VSAM NAFILE and POLFILE dataset into a VSAM dataset | SEQ2KSDS on page 199 |
| Convert a PNG (Portable Network Graphic) file into a logo | PNG2LOG on page 188 |
| Convert a print stream from cut-sheet to continuous-form | AFPCOPY on page 14 |
| Convert a single FAP file into a compiled FAP (CFA) file | FAP2CFA on page 67 |
| Convert a text file from one codepage to another | CPCNV on page 54 |
| Convert AFP fonts into PCL fonts | AFP2PCL on page 10 |
| Convert AFP overlays into FAP files | OVL2FAP on page 181 |
| Convert AFP page segments into LOG files | PSEG2LOG on page 192 |

| To... | Use... |
|---|---|
| Convert AFP print files into variable block format | AFP2VB on page 12 |
| Convert all of the FAP files listed in a FORM.DAT file into a compiled FAP files | FDT2CFA on page 83 |
| Convert an RTF file into a FAP file | RTF2FAP on page 197 |
| Convert DCD files into a FAP files | DCD2FAP on page 65 |
| Convert a Documerge VRF file into a Documaker Workstation-style import file, an export file, or an INI file | VRF2EXP on page 208 |
| Convert DOS archive files into Windows archive files | ARCCNV on page 21 |
| Convert embedded logos into referenced logos | FXLOGREF on page 115 |
| Convert file names into lowercase file names (UNIX) | up2low on page 205 |
| Convert files written in variable block record format into AFP files | VB2AFP on page 206 |
| Convert files written in variable block record format into BARR format files | VB2BARR on page 207 |
| Convert FormMaker II PCL overlays into FAP files | PCL2FAP on page 186 |
| Convert LOG files into AFP page segments | LOG2PSEG on page 153 |
| Convert LOG files into DOS entry LOB logo files | LOG2LOB on page 152 |
| Convert LOG files into JPEG files | LOG2JPG on page 151 |
| Convert LOG files into TIFF files | LOG2TIF on page 155 |
| Convert LOG files into Xerox fonts | LOG2XFNT on page 157 |
| Convert LOG files into a Xerox image files | LOG2IMG on page 150 |
| Convert multi-color logos into JPG files and monocolor logos into TIFF files for use as VIPP printer resources | LOG2VIPP on page 156 |
| Convert PCL fonts into AFP fonts | PCL2AFP on page 185 |
| Convert PCL fonts into Xerox Metacode fonts | PCL2XFNT on page 187 |
| Convert PostScript fonts into PCL fonts | PS2PCL on page 189 |
| Convert the records of a VSAM KSDS into a sequential file | KSDS2SEQ on page 121 |
| Convert TrueType fonts into PCL fonts | TT2PCL on page 200 |
| Convert uploaded AFP files into MVS compatible files | AFP2MVS on page 7 |
| Convert uploaded BARR formatted Metacode output into an MVS compatible file | BARR2MVS on page 47 |
| Convert Xerox fonts into PCL bitmap fonts | XFNT2PCL on page 215 |
| Convert Xerox FRM files into FAP files | FRM2FAP on page 107 |
| Convert Xerox Metacode files into FAP files | MET2FAP on page 158 |

| To... | Use... |
|---|---|
| Convert Xerox Metacode resources into downloadable BARR or PCO files | XERDNLD on page 214 |
| Create DDL files from your DFD files | DFD2DDL on page 66 |
| Create a formatted dump of an AFP file | AFPDUMP on page 15 |
| Create a formatted dump of a Xerox FRM file | FRMDUMP on page 109 |
| Create a library file from a response file | LBRYMGR on page 122 |
| Create a response file | LBRYMGR on page 122 |
| Create AFP coded font files | AFPCF on page 13 |
| Create AFP form definition resource objects | AFPFMDEF on page 19 |
| Create NA and POL files from a DAP archive and call plug-ins to process the transactions in these files | ARCSPLIT on page 39 |
| Create the ERRFILE.DAT and LOGFILE.DAT files from the MSGFILE.DAT file | TRANSLAT on page 203 |
| Create DDT files from FAP files | FAP2DDT on page 69 |
| Debug (and execute) DAL scripts | DALRUN on page 61 |
| Fix field offsets | FIXOFFS on page 97 |
| List the fonts used in specified FAP files | FONTLIST on page 105 |
| List the resources found in a print stream or those not located in the library | MRGCHK on page 178 |
| Maintain FAP files | FIXFORM on page 92 |
| Maintain FXR file | FIXFXR on page 96 |
| Optimize Metacode print streams before they are sent to the printer | METOPT on page 159 |
| Prepare AFP resources to be uploaded | ADDCRLF on page 6 |
| Process library scripts | LBYPROC on page 132 |
| Reindex dBase files | REINDEX on page 194 |
| Reformat MVS-generated Metacode output for submission to a BARR system | BARRWRAP on page 49 |
| Remove embedded logos from FAP files | FXLOGREF on page 115 |

| To... | Use... |
|---|---|
| Rename CAR files | CARREN on page 52 |
| Rename FAP files | RENFORM on page 196 |
| Repair archive files | ARCFIX on page 23 |
| Reset locked user IDs | OPENUSER on page 180 |
| Reset PostScript printers | PSRESET on page 193 |
| Restore an archive backed up or split with ARCSPLIT | ARCMERGE on page 24 |
| Retrieve archive records to produce files to send to plug-in functions | ARCRET on page 26 |
| Split an archive file | ARCSPLIT on page 39 |
| Synchronize a library | LBYSYNC on page 147 |
| Use a MRGCHK list to add missing fonts to an FXR | MRGADD on page 177 |
| View a Documanage archive file | ARCVIEW on page 45 |
| View library versions | FSIVER on page 110 |

# ADDCRLF

Use the ADDCRLF utility to prepare AFP resource files created in a Windows environment for uploading to MVS systems. This utility adds carriage return/line feeds (CR/LF) to the files. The CR/LFs serve as record delimiters.

**NOTE:** Some AFP resources, such as AFPBAT1 output from the GenPrint program, can have carriage return/line feeds (x'0D0A') embedded in them. On these resources, the ADDCRLF utility will fail, giving you a message stating the data has embedded carriage return/line feeds (CR/LFs). If this happens, do the file transfer with *no CR/LF*, and then use the AFP2MVS utility on the MVS system. Refer to AFP2MVS on page 7 for more information.

## Program names

| | |
|---|---|
| Windows | ADDCRLFW.EXE |

## Syntax

**ADDCRLFW /I /F**

| Parameter | Description |
|---|---|
| /I | The AFP file name with the extension TMP, which you can omit. |
| /F | Add this parameter to force the output file to be built, ignoring any embedded CR/LF errors. |

For the input file, the utility looks for the default extension *TMP*. For the output file, the utility will assign the extension *IBM*.

PSF software on the MVS is record-oriented software. It needs each structured field record in an AFP resource such as font, formdef, overlay, page segment, and so on, in discrete records.

PSF2 on a Windows system has no concept of records. The entire datastream is a single record. If you use the system in a Windows environment to create AFP resources, and you want to transfer them to MVS, you *must* insert CR/LFs at the end of each structured field and then use the CR/LF keyword on the file upload command.

## Example

**ADDCRLF /I=afpfile**

This creates a file called AFPFILE.IBM. The input file must have the extension *TMP*. You do not have to enter the extension if you enter the command as shown above. The output file will have the same file name, but with the *IBM* extension.

# AFP2MVS

Use the AFP2MVS utility to *record-orient* an AFP file that has been transferred to z/OS (MVS, OS/390) from a Windows or UNIX-based system.

When an AFP file such as a print stream, font, or a page segment is uploaded from Windows or UNIX to a z/OS (OS/390, MVS) system, the resulting AFP file is not oriented into separate records — instead it looks like one continuous record. For the AFP file to be properly processed on z/OS, it must be oriented into separate records.

Because of the structure of AFP records, these records can be separated fairly easily. Each AFP record usually begins with a 0x5A byte. In the EBCDIC character set, this value (code point) is displayed as an exclamation point (!). Following the 0x5A byte is the length of the record. The AFP2MVS utility reads through the AFP file, separates the file into records and writes the resulting record-oriented file out as an output file. If you were then to browse the file, the 0x5A records (first byte displayed as an exclamation point) would display one per row.

## Program names

z/OS (MVS) AFP2MVS

## Syntax

**AFP2MVS [/I=] [/L=]**

| Parameter | Description |
|---|---|
| /I | If the input file is a Partitioned Data Set (PDS), use this parameter to specify the member within the PDS to process. To process all members, enter /I=*.<br>If the input file is a sequential file, omit this parameter. |
| /L | Include this parameter to tell the utility that the dataset is a Mixed Mode dataset. Unlike regular AFP files, Mixed Mode AFP files typically contain records of line data — not AFP records. |

Keep in mind...

- Sample JCL for this utility is located in member AFP2MVSX in the JCLLIB PDS that is created when Documaker Server is installed on z/OS.
- In Documaker Server version 11.0 and earlier, the DD names RSCOS2 and RSCMVS were used to designate the input and output files. The preferred DD names are now *INPUT* and *OUTPUT*.
- When you transfer AFP files from Windows or UNIX to z/OS, be sure to transfer the files in binary mode, not text mode.
- The dataset you create on z/OS to contain the AFP files you upload should have DCB characteristics similar to the dataset the AFP2MVS utility will write to. For example, for a sequential file you could specify DCB characteristics as shown here:

```
Data Set Name . . . . : FSI.V112.RPEX1.GENPRINT.PRTBAT1.FROMPC

General Data                            Current Allocation
 Management class . . : **None**         Allocated cylinders : 1
 Storage class  . . . : STANDARD         Allocated extents . : 1
  Volume serial . . . : DCI009
  Device type . . . . : 3390
 Data class . . . . . : **None**        Current Utilization
  Organization  . . . : PS               Used cylinders  . . : 0
  Record format . . . : VBM              Used extents  . . . : 0
```

```
Record length . . . : 8209
Block size  . . . . : 23500
1st extent cylinders: 1
Secondary cylinders : 1
Data set name type  :                  SMS Compressible  . : NO
```

### Example

Here are some examples:

### Example 1

This example demonstrates running the AFP2MVS utility on a print stream:

In this example, the AFP2MVS utility reads the print stream associated with the INPUT DD statement (FSI.V112.RPEX1.GENPRINT.PRTBAT1.FROMPC), separates the print stream into records, then writes the new print stream to the file associated with the OUTPUT DD statement (FSI.V112.RPEX1.GENPRINT.PRTBAT1).

Since the file associated with the INPUT DD statement is a Sequential Data Set and not a Partitioned Data Set (PDS), the /I parameter is not necessary.

```
//AFP2MVSD EXEC PGM=IEFBR14
//OUTPUT   DD DSN=FSI.V112.RPEX1.GENPRINT.PRTBAT1,
//         DISP=(MOD,DELETE),SPACE=(TRK,0),UNIT=SYSDA
//*
//AFP2MVS  EXEC PGM=AFP2MVS
//STEPLIB  DD DSN=FSI.V112..LINKLIB,DISP=SHR
//         DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=*
//INPUT    DD DSN=FSI.V112.RPEX1.GENPRINT.PRTBAT1.FROMPC,DISP=SHR
//OUTPUT   DD DSN=FSI.V112.RPEX1.GENPRINT.PRTBAT1,
//         UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(,CATLG),
//         DCB=(RECFM=VBM,LRECL=8209,BLKSIZE=23500)
```

### Example 2

This example demonstrates running the AFP2MVS utility on a PDS that contains several fonts:

In this example, the AFP2MVS utility reads each member of the PDS that is associated with the INPUT DD statement (FSI.V112.RPEX1.FONTLIB.FROMPC). Each member is then separated into records and the resulting record-oriented member is written to the PDS associated with the OUTPUT DD statement (FSI.V112.RPEX1.FONTLIB).

Since the file associated with the INPUT DD statement is a PDS, the /I parameter is included to indicate which members to process. In this case, you specify /I=* to indicate that all members are to be processed.

**NOTE:** In the EXEC statement's PARM parameter, the first forward slash (/) is used to separate any Language Environment (LE) runtime options from the program's command line options. If you need to specify any LE runtime options, place those options before the first forward slash. Any options placed after the first forward slash are considered options for the program being executed.

Documaker-related programs typically use the forward slash (/) or dash (-) as the initial character of a command line parameter, so the number of forward slashes on the command line will total one more than the number of parameters you specified.

```
//AFP2MVSD EXEC PGM=IEFBR14
//OUTPUT   DD DSN=FSI.V112.RPEX1.FONTLIB,
```

```
//          DISP=(MOD,DELETE),SPACE=(TRK,0),UNIT=SYSDA
//*
//AFP2MVS   EXEC PGM=AFP2MVS,PARM='/ /I=*'
//STEPLIB   DD DSN=FSI.V112.LINKLIB,DISP=SHR
//          DD DSN=SYS1.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=*
//INPUT     DD DSN=FSI.V112.RPEX1.FONTLIB.FROMPC,DISP=SHR
//OUTPUT    DD DSN=FSI.V112.RPEX1.FONTLIB,
//          UNIT=SYSDA,SPACE=(CYL,(1,1,1)),DISP=(,CATLG),
//          DSNTYPE=LIBRARY,
//          DCB=(RECFM=VBM,LRECL=8209,BLKSIZE=23500)
```

# AFP2PCL

Use the AFP2PCL utility to convert an AFP bitmap font into a corresponding PCL bitmap font. See following page for more information on AFP fonts and standard font naming conventions.

**NOTE:** You can also perform this task using the Font Manager. See the Docucreate User Guide for more information.

## Program names

Windows AFP2PCLW.EXE

## Syntax

**AFP2PCLW /I /S /D /O**

| Parameter | Description |
|---|---|
| /I | The AFP file name, such as FILENAME.FNT (AFP coded font file), the extension is optional and will defaulted to *FNT*. |
| /S | The source dots per inch (DPI), either 240 or 300 (the AFP font DPI setting default is 300). |
| /D | The destination dots per inch, either 240 or 300 (the PCL font DPI setting default is 300). |
| /O | The output file, such as FILENAME.PCL (a PCL font file) If you omit the output name, the output PCL file name will be the same as the input file name, except for the PCL extension. |

The utility creates a PCL character set file using codepage 1004. You must include the /I parameter.

## Example

**AFP2PCL /I=xofacob9**

This will convert an AFP coded font file, *xofacob9.fnt*, into a PCL font file named *xofacob9.pcl*.

The AFP2PCL utility requires these files:

- AFP coded font file (system coded font files are named *XoDAxxxx.fnt*)
- AFP character set file (system character set files are named CoFAxxxx.240/300)
- AFP codepage file (the system's standard codepage is T1DOC037). The codepage file *cannot* have an extension.

In addition, AFP2PCL uses the FSISYS.INI and CODEPAGE.INI files. AFP2PCL looks for the FMRES control group in the FSISYS.INI file. In the FMRES control group, AFP2PCL looks for DefLib and CodePage options. Set the DefLib option to the directory where your system resources are stored.

Set the CodePage option to the name of the INI file which contains codepage information. The system looks for the CODEPAGE.INI file in the DefLib path specified in the FMRES control group. In this example,

```
< FMRes >
    CodePage    = CODEPAGE.INI
```

```
DefLib      = \newfonts\agfa\
```

the name of the INI file containing codepage information is *CODEPAGE.INI* and will be located in the *\newfonts\agfa* directory.

## AFP terminology

An AFP font is composed of these component files: *coded font*, *codepage*, and *character set files*. The coded font file contains the names of character set and codepage files to use when printing.

> **NOTE:** You can use the AFPDUMP utility to find out the names of character set and codepage files contained in a coded font file. See AFPDUMP on page 15 for more information.

The *codepage* maps text to the characters in a character set file. Each character, like a capital 'A', has a particular name and a particular numeric value known as a *code point*. The character capital 'A' has an AFP name of LA020000. On a PC, a capital 'A' usually has a code point of 65. However, on a mainframe, a capital 'A' usually has a code point of 193.

For both the PC and the mainframe to print the letter 'A', a different codepage may be used. The codepage used to print text from the PC would associate a code point of 65 with the letter 'A' (LA020000). The codepage used to print text from the mainframe would associate a code point of 193 with the letter 'A' (LA020000). The codepage is merely a list of the character names and their associated code points.

The *character set file* contains the characters which can be printed. Like PCL fonts, each character set file can only represent a single font typeface, style, and point size. A character set file also specifies the list of AFP character names, like LA020000, it can print.

So, when you try to print the letter 'A' (code point 65 on a PC) using an AFP coded font file, the system examines the coded font file to determine the names of the codepage and character set files. It then looks up code point 65 in the codepage file to find the AFP character name associated with it. If the codepage is set up for PC printing, it finds the character named LA020000 associated with code point 65. The character set file is then used to print the bitmap information associated with the character name LA020000. An AFP character name, like LA020000, must be present in both the AFP codepage and character set files.

## Font naming convention

The system's AFP coded font files are named XoDAxxxx.FNT, where *Xo* indicates a coded font file, *D* symbolizes Documaker, and *A* denotes the AFP font. The first two *XX*s indicate the font type family. The next *X* indicates style. The last *X* indicates the font point size.

AFP character set files are named CoFAxxxx.240 or CoFAxxxx.300. The *Co* indicates a character set file. The extensions *240* and *300* indicates the dots per inch or DPI setting. The rest of the naming convention is the same as previously discussed.

# AFP2VB

Use this utility if you have a printable (native) AFP file and you want to view it to use it with Docuview LFS. This utility converts the file into variable block format.

## Program names

Windows AFP2VB.EXE

## Syntax

**AFP2VB /I=inputfile /O=outputfile /D**

| Parameter | Description |
|---|---|
| /I | The name of the native AFP file. |
| /O | The name of the output AFP file which will have variable block record format. |
| /D | (Optional) This parameter tells the utility to add Docusave comments. |

# AFPCF

Use the AFPCF utility to create coded font files for AFP printers.

## Program names

| | |
|---|---|
| Windows | AFPCFW32.EXE |

## Syntax

**AFPCFW32 /C /T /X**

| Parameter | Description |
|---|---|
| /C | The AFP character set file name. |
| /T | The AFP codepage file name. |
| /X | The AFP codefont file to create (.FNT). |

## Example

You would enter:

**AFPCFW32 /c=CoFATINo /t=T1DOC037 /x=XoDATINo**

In this example, AFPCF creates an AFP coded font file called XoDATINo.FNT.

To print using this coded font, all of these files must be resident on the printer. On MVS, it should be placed in a PDS which is available to the PSF procedure. On a Novell print server, it is must be installed into the coded font group within PSF2.

To use this coded font for printing with Documaker Server, enter the name of the coded font file (without the FNT extension) into the Coded Font File field on the Printers tab of the Font Maintenance window in the Font Manager. For more information about the Font Manager, see the Docucreate User Guide.

# AFPCOPY

Use the AFPCOPY utility to convert a print stream from *cut-sheet* to *continuous-fo*rm. The utility does this by taking an AFP print stream which contains multiple BEGINDOC and ENDDOC statements and generating a print stream with only one BEGINDOC and one ENDDOC statement.

## Program names

| | |
|---|---|
| Windows | AFPCOPY.EXE |
| MVS | AFPCOPY |

## Syntax

**AFPCOPY /I /O**

| Parameter | Description |
|---|---|
| /I | The input file name (with page breaks). |
| /O | The output file name. |

No parameters are required.

The input AFP print stream is read from the INFILE DD statement and the output AFP print stream is written to the OUTFILE DD statement. Look in member AFPCOPYX of JCLLIB to find an example of this utility.

On MVS systems, use *DD:INFILE* as the input file in the JCL. Use *DD:OUTFILE* as the output file in the JCL. Use *VBM,8205,8209* on the SysUT2 DD statement.

## Example

**AFPCOPY /infile /outfile**

# AFPDUMP

Use the AFPDUMP utility to create a text file from a print-ready AFP file. This utility produces a formatted dump of an AFP output file.

## Program names

| | |
|---|---|
| Windows | AFPDPW32.EXE |

## Syntax

**AFPDPW32 /I /H**

| Parameter | Description |
|---|---|
| /I | The file name of the AFP file to dump. Include the extension. The output is named *filename.dds* |
| /H | (Optional) Include this parameter to tell the utility to also dump the HEX values. |

The AFPDUMP utility reads a print-ready AFP file and creates a text file which contains English explanations of the AFP printer commands. If you include the /H option, the utility lists the hexadecimal characters for each AFP command before the its English explanation.

In addition to reading a print-ready AFP file, the AFPDUMP utility can also read AFP overlays (page segments) and AFP font files (coded font, codepage, and character set files).

You can use the AFPDUMP utility to check:

- Which fonts are being used in an AFP print ready or overlay.
- If text is correctly output in an AFP print ready or overlay.
- Which codepage and character set files are used by a coded font file.
- What characters and code points are defined in an AFP codepage file.
- If the characters named in the codepage file match those in the character set file.

## Example

**AFPDUMP /I=filename.pds /H**

This converts the *filename.pds* file into the *filename.dds* file which you can read using any text editor.

**NOTE:** If the file to be converted does not have a PDS extension, you must include the extension when you enter the command. It the file has no extension, add a period (.) after the file name.

Here is an example AFPDUMP output file (*filename.dds*) with the /H option:

```
 ....5A 0010 D3A8A8 00 0000
000,Begin,Document,16
....4040404040404040
    ,
....   5A 0010 D3ABCC 00 0001
001,    Map,Medium Map,16
....   D7D3E4D640404040
        ,PLUO
....   5A 0010 D3A8AF 00 0002
002,    Begin,Page,16
....   4040404040404040
        ,
....       5A 0010 D3A8C9 00 0003
003,        Begin,Active Env. Group,16
....       4040404040404040
            ,
....           5A 0017 D3A6AF 00 0004
004,            Descriptor,Page,23
....           00000960096000007F8000A50000000
            ,unit base 0 0,L-Units 2400 2400,Page Size x=2040 y=2640
....           5A 0019 D3A69B 00 0005
005,            Descriptor,Composed Text,25
....           00000960096007F80A5000000000000000

....       5A 0010 D3A9C9 00 0006
006,        End,Active Env. Group,16
....       4040404040404040
            ,
....       5A 0010 D3A89B 00 0007
007,        Begin,Composed Text,16
....       4040404040404040

....           5A 0012 D3EE9B 00 0008
008,            Data,Composed Text,18
....           2BD306F700002D0002F8

            Set Text Orientation,6,0000,2D00
            NOP,2,
....       5A 0010 D3A99B 00 0009
009,        End,Composed Text,16
....       4040404040404040

....       5A 0010 D3A89B 00 000A
010,        Begin,Composed Text,16
....       4040404040404040

....           5A 0012 D3EE9B 00 000B
011,            Data,Composed Text,18
....           2BD306F700002D0002F8

            Set Text Orientation,6,0000,2D00
            NOP,2,
....       5A 0010 D3A99B 00 000C
012,        End,Composed Text,16
```

```
....       4040404040404040

....       5A 0010 D3A89B 00 000D
013,        Begin,Composed Text,16
....       4040404040404040

....           5A 000C D3EE9B 00 000E
014,            Data,Composed Text,12
....           2BD302F8

           NOP,2,
....       5A 0010 D3A99B 00 000F
015,        End,Composed Text,16
....       4040404040404040

....       5A 0010 D3A89B 00 0010
016,        Begin,Composed Text,16
....       4040404040404040

....           5A 000C D3EE9B 00 0011
017,            Data,Composed Text,12
....           2BD302F8

           NOP,2,
....       5A 0010 D3A99B 00 0012
018,        End,Composed Text,16
....       4040404040404040

....       5A 0010 D3A8BB 00 0013
019,        Begin,Graphic,16
....       4040404040404040
           ,
....           5A 0010 D3A8C7 00 0014
020,            Begin,Obj. Env. Group,16
....           4040404040404040
                ,
....               5A 001C D3A66B 00 0015
021,                Descriptor,Object Area,28
....               034301084B000009600960094C020007B1000A60
                   ,Unit Base 0 0,L-Units 2400 2400,Object Area Size
x=1969 y=2656
....               5A 0020 D3AC6B 00 0016
022,                Position,Object Area,32
....               011800000000000000002D000000000000000000002D0000
                   ,Object Area Origin 0 0, Object Area Orientation
(0, 90), Object Content Origin 0 0
                    ,Use the current coordinate system
....               5A 000D D3ABBB 00 0017
023,                Map,Graphic,13
....               0005030420
                    ,Scale-to-fit
....               5A 0020 D3A6BB 00 0018
024,                Descriptor,Graphic,32
....               F6160000000009600960000080007FFF80007FFF00000000
                    ,No absolute picture units ,Unit base 0 ,Window
Coordinates xLeft=-32768 xRight=32767 yBottom=-32768 yTop=32767
```

```
....           5A 0010 D3A9C7 00 0019
025,            End,Obj. Env. Group,16
....           4040404040404040
                ,
....           5A 0056 D3EEBB 00 001A
026,            Data,Graphic,86
....
700C10101010000000400000000000A0168C00A011807C10C1000100030003000500
0700060002208300030000000000C706000000000100C514000000001500400030
000000450040006000000

            Begin Segment Introducer, 12
            Set Color, 01
            Begin Area, c0
            Set Color, 01
            Set Line Type, 07
            Line, Point (4096 4096), Point (12288 12288), Point
(20480 28672),
            End Area, 00
            Set Arc Parameters, Point (12288 12288), Point (0 0),
            Full Arc, Point (0 0), Scale 1.000000
            Fillet, Point (0 0), Point (5376 16384), Point (12288 0),
Point (17664 16384), Point (24576 0),

....       5A 0010 D3A9BB 00 001B
027,        End,Graphic,16
....       4040404040404040
            ,
....       5A 0010 D3A89B 00 001C
028,        Begin,Composed Text,16
....       4040404040404040

....           5A 000C D3EE9B 00 001D
029,            Data,Composed Text,12
....           2BD302F8

            NOP,2,
....       5A 0010 D3A99B 00 001E
030,        End,Composed Text,16
....       4040404040404040

....   5A 0010 D3A9AF 00 001F
031,    End,Page,16
....   4040404040404040
        ,
....5A 0010 D3A9A8 00 0020
032,End,Document,16
....4040404040404040
```

# AFPFMDEF

Use the AFPFMDEF utility to create an AFP form definition resource file from a DAT file (see the example below).

## Program names

| | |
|---|---|
| Windows | AFPFMDEW.EXE |
| MVS | See the Documaker Server Installation Guide |

## Syntax

**AFPFMDEW /I**

| Parameter | Description |
|---|---|
| /I | The data file's first two characters must be *F1*. |

The AFP form definition resource file defines certain print attributes, such as paper size, orientation, and duplex settings. This utility takes a text file with a DAT extension and compiles it into an AFP form definition resource file which is installed on the AFP printer.

## Example

**AFPFMDEW /I=f1fsi.dat**

The output file will have the same name as the data file, with an FDF extension.

Here is an example of a form definition DAT file:

```
*********************************************************************
*
* FIELD LAYOUT
*
*;Medium Map Name;Medium Map Id;X Origin;Y Origin;Paper
Size;Orientation;Copies;
*
*                  Stacking;Tray;Flash;Duplex;Print Quality;
*
* Medium Map Name => up to 8 character long use A-Z 0-9 $ # @
*
* Medium Map Id => 1 to 127
*
* X Origin      => 0 to 32767
*
* Y Origin      => 0 to 32767
*
* Paper Size    => L           Letter (default)
*                  E           Executive
*                  G           Legal
*                  A           A4
*
* Orientation   =>  O1 - Portrait
*                   O2 - Landscape
*                   O3 - Portrait 90
*                   O4 - Landscape 90
*
* Copies        => 1 to 255
*
* Stacking      => Yes or No
*
```

```
* Tray          =>  T1 - Source Drawer #1
*                    T2 - Source Drawer #2
*                    T3 - Source Drawer #3
*                    T4 - Manual Feed
*                    T5 - Envelopes
*
* Flash         => Yes or No
*
* Duplex        => Simplex, Duplex, or Tumble duplex
*
* Print Quality => Default, Lowest, or Highest
*
*
*
*-----------------------------------------------
* Medium Map Name convention
*-----------------------------------------------
*      Position    Values      Description
*      ========    ======      ===========
*          0                   Orientation
*                      L           Landscape
*                      P           Portrait (default)
*          1                   Papersize
*                      E           Executive
*                      G           Legal
*                      A           A4
*                      L           Letter (default)
*          2                   Tray selection
*                      M           Manual
*                      F           Envelope feeder
*                      L           Lower
*                      U           Upper (default)
*          3                   Duplex
*                      L           Long binding
*                      S           Short binding
*                      O           Simplex
*                      ?           Unknown (default)
*
*
*
*****************************************************************
;B1;1;0;0;L;O1;1;Y;T1;Y;S;L;
*----Landscape [Paper Size] Upper Simplex
;LLUO;60;0;0;L;O2;255;Y;T1;Y;S;D;
```

A standard form definition file called *F1FMMST* is included with the system. It contains medium maps of all possible combinations of orientations, paper sizes, duplex mode and so on.

This form definition file should be resident on the printer. On MVS, place it in a PDS which is available to the PSF procedure. On a Novell print server, define it as a device option in the printer profile in PSF2.

# ARCCNV

Use the ARCCNV utility to convert old DOS archive files into the newer FAP file format.

## Program names

Windows ARCCVW32.EXE

## Syntax

**ARCCVW32 /O /N /F /I /R**

| Parameter | Description |
|---|---|
| /O | The old archive directory, where the old archive files are stored. |
| /N | The new archive directory, where the new archive files will be stored. |
| /F | The forms directory, where the corresponding FAP files are stored. |
| /I | (Optional) The file name for a single file conversion. |
| /R | (Optional) The file name restart point. |

Convert all forms and then run this utility. The Windows archive subdirectory must be empty when you start the utility, otherwise you may lose data.

## Example

Here's an example of how you could use this utility:

```
ARCCVW32 /O=\PPS\RESLIB\SAMPCO\ARC /N=\PPSWIN\MSTRRES\SAMPCO\ARC /
F=PPSWIN\MSTRRES\SAMPCO\FORMS
```

After the conversion is complete, you must copy the APPIDX.DFD file to the new archive subdirectory. These conversion steps are necessary for every company library in the DOS PPS system.

## Error messages

Here are explanations of possible error messages you may receive:

### NA load image failure on 'IMAGENAME' at offset OFFSET

This message generally appears if:

- The FAP file for 'IMAGENAME' is missing
- You entered an incorrect forms subdirectory on the command line

To correct this error, do the following:

1. Remove all files from WINDOWS archive subdirectory.
2. Correct command line parameters or find the FAP file for IMAGENAME.
3. Start the ARCCNVW utility again.

### Cannot load file 'FILENAME.DAT'

This message generally appears if the data file is corrupted in DOS PPS archive. To correct this error, do the following:

1. Remove all files from the Windows archive subdirectory.
2. Edit the FILENAME.DAT file using text editor and correct the corrupted data.

3 Rearchive the FILENAME.DAT file using the DOS command line utility, REPLARC. Here is the syntax for the REPLARC utility:

```
REPLARC.EXE  <CARFILENAME> <FILENAME.DAT>
```

For example...

```
REPLARC.EXE ARCHIVE.CAR B225F307.DAT
```

4 Then, restart the archive conversion. If the problem reappears, see the next topic.

## Checking ARCHIVE.CAR file for possible corrupted data files

Run the DOS utility RESTARC.EXE and redirect the output into a file. For example...

```
RESTARC.EXE ARCHIVE.CAR /V > FILE.BAT
```

Edit the FILE.BAT file using a text editor. The file will look similar to this one:

```
DOS Archive Restore Program
Usage
   restarc <archive file name> [/Verbose]
Example
   restarc archive.car
File B1DA85BC.POL is OK
File B1DA85BC.DAT is OK
File B1DA872F.POL is OK
File B1DA872F.DAT is OK
File B1DC356D.POL is OK
File B1DC356D.DAT is OK
File B1E1289C.POL is OK
File B1E1289C.DAT is OK
Done
```

You have to create a batch file using this file. The batch file should look like this:

```
RETRIEVE.EXE ARCHIVE.CAR B1DA85BC.POL
RETRIEVE.EXE ARCHIVE.CAR B1DA85BC.DAT
RETRIEVE.EXE ARCHIVE.CAR B1DA872F.POL
RETRIEVE.EXE ARCHIVE.CAR B1DA872F.DAT
RETRIEVE.EXE ARCHIVE.CAR B1DC356D.POL
RETRIEVE.EXE ARCHIVE.CAR B1DC356D.DAT
RETRIEVE.EXE ARCHIVE.CAR B1E1289C.POL
RETRIEVE.EXE ARCHIVE.CAR B1E1289C.DAT
RETRIEVE.EXE ARCHIVE.CAR B1EA6DB4.POL
RETRIEVE.EXE ARCHIVE.CAR B1EA6DB4.DAT
```

Where the file names at the end of each line are the same names as those in the original FILE.BAT file. Run the batch file to unzip all DAT and POL files from the ARCHIVE.CAR file. Then, enter the following command:

```
GREP -l -v [\ -~] *.DAT
```

**NOTE:** You must enter a space after the backslash (\)

This command displays a list of files which contain non-ASCII characters. These are files you must fix. See the information on the error message *Cannot load file FILENAME.DAT* for information on what to do next.

After you fix all files, run the ARCCNV utility.

# ARCFIX

Use the ARCFIX utility to evaluate and, if necessary, attempt to repair archive files.

**NOTE:** The ARCFIX utility should only be used by those who understand archive well because of the risk of data loss.

Make sure you *back up your archive files* before you run this utility. Support Services will run this utility for you if you encounter problems. For information on how to contact Support Services, see Finding the Right Utility on page 2.

## Program names

Windows ARCFXW32.EXE

## Syntax

**ARCFXW32 /I /A /F**

| Parameter | Description |
|---|---|
| /I | Enter the name of the CAR file. |
| /A | (Optional) Include this parameter to repair all files. |
| /F | (Optional) Include this parameter to fix the table of offsets. |

Please note that:

- Index files must be in current directory.
- You must have the following files in the working directory:
  - APPIDX.DFD
  - ???.CAR (for example. ARCHIVE.CAR)

The ARCFIX utility first checks to see that each record in the CAR file has a corresponding APPIDX record and then deletes any extra APPIDX records.

If you use the /F parameter, the ARCFIX utility rebuilds the table of offsets at the end of the CAR file.

# ARCMERGE

Use this utility to combine two archives that have the same DFD files for the archive index, catalog, and CAR file. This utility can take a secondary archive and merge it into the main archive.

**NOTE:** This utility only works with xBase files. It does not work on SQL, Oracle, or DB2 databases.

## Program names

| | |
|---|---|
| Windows | ARCMW32.EXE |

## Syntax

**ARCMW32 /I=secondary-archive-directory /INI**

| Parameter | Description |
|---|---|
| /I | If the secondary archive is a database, enter the name of the index (APPIDX) file.<br>If the secondary archive is a file, enter the name and path of the file. You can omit the remaining parameters. |
| /INI | Enter the name of the INI file you want the utility to reference. |

Run this utility from your master archive environment. The ARCMERGE utility uses the FSIUSER.INI and FSISYS.INI files, along with your resources. You can include these additional INI options in the FSIUSER.INI file to specify the names of the split archive files/tables.

```
< ArcRet >
    MergeLog =
```

| Option | Description |
|---|---|
| MergeLog | The name of the log file. The default is *ARCMERGE.LOG*. The utility creates this file in the Data directory. |

## Example

Assume that D:\REL10\MSTRRES\ARCB is the secondary archive, to be merged into the main archive. This is the command you would use if the secondary archive is stored in files rather than in a database:

```
ARCMW32 /I=d:\rel10\mstrres\arcb
```

If the archive is in a database archive, you would enter this command:

```
ARCMW32 /I=arcb/appidxs /A=arcb/archives /C=arcb/catalogs
```

Here is an example of the ARCMERGE.LOG file:

```
--- ArcMerge ---

Rows copied from <archives> to<archive>, Day Mon DD HH:MM:SS YYYY
C: Original ArcKey: New ArcKey
.
.
End of copy, Day Mon DD HH:MM:SS YYYY

Rows copied from<catalogs> to<Catalog>, Day Mon DD HH:MM:SS YYYY
C:Catalog ID
.
.
End of copy, Day Mon DD HH:MM:SS YYYY

Rows copied from<catalogs> to<Catalog>, Day Mon DD HH:MM:SS YYYY
C:FIELD1,FIELD2,FIELD3,original ARCKEY:updated ARCKEY
.
.
End of copy, Day Mon DD HH:MM:SS YYYY
```

See also ARCSPLIT on page 39

# ARCRET

Use this utility to retrieve records from your archives and produce files that can then be sent to *plug-in* functions to generate additional output. You can choose from these plug-in functions:

| Plug-in | Description |
|---|---|
| PLGGenPrint | Produces a batch file the GenPrint program can use to produce printed form set output. Use it to reprint your archives. |
| PLGGenArc | Produces a batch file the GenArc program can use to archive transactions to a different archive. Use it to help migrate your archive to another archiving method. |
| PLGTest | Used to test the retrieval results from the archive. This function does nothing with the output of the ARCRET utility. |

For example, you can use the PLGGenPrint plug-in function to take output from the ARCRET utility, produce a batch file, and then run the GenPrint program to produce printed form sets.

You can also use the PLGGenArc plug-in to migrate traditional LAN flat file archives to host or server based SQL database format archives, or as a part of an upgrade or conversion to a Documanage repository.

Before starting any migration, you should...

- Back up your archives.
- Read all of the documentation concerning this utility.
- Run some tests and verify the results. Testing should help you:
  - Determine if archiving and retrieving from the new system is working properly.
  - Decide the optimal size of the batch sets.
  - Estimate how long the entire process will take.

After you verify the test results, change all systems that do archiving and retrieval to use the new system. Do not use the old system while you are migrating from it.

**NOTE:** After you complete the migration, hang onto your original archives for awhile—just in case. Only after you are satisfied that the new system is working properly and that all of your data has been migrated, should you consider dropping the old archive and old archive system.

## Program names

Windows ARCRET.EXE

## Syntax

**ARCRET /INI /R /N /S /P /BEF /OR /AFT /DAL /NC /DB /K /REV /LOAD /RUNDATE**

| Parameter | Description |
|---|---|
| /INI | Specifies the INI file to use. If you omit this option, the utility uses the FSIUSER.INI file as the default. |
| /R | Defines the sequential row or record number to begin processing. The default is one. |
| /N | Tells the utility to stop after retrieving this number of sequential rows or records. The default is to retrieve all rows. |
| /S | Sets the number of records to include in a set or batch. The default is one transaction per set. |
| /P | Tells the utility to pause between sets.The default is off. |
| /BEF | Selects rows/records with a RUNDATE before this date (YYYYMMDD). The date you specify is not included in the set. |
| /AFT | Selects rows/records with a RUNDATE after this date (YYYYMMDD). The date you specify is not included in the set. |
| /OR | Changes the date comparison to mean *before or after* as opposed to the default *before and after*. |
| /DAL | Specifies a script file the utility should run to determine if a row/record is included in a set. |
| /NC | Tells the utility not to compile the DAL script. DAL scripts are compiled by default to improve performance. |
| /DB | Dumps debug information as the script runs. When you include this parameter, use a small set (/S) size. |
| /K | Keeps all intermediate files by renaming them instead of deleting them. Using this option can leave numerous files that you will have to remove manually. |
| /REV | Tells the utility to process the archives in reverse order. The last record is read first and the first record is read last. This only applies to xBase databases. |

| Parameter | Description |
|---|---|
| /LOAD | This parameter lets you filter archive records by examining the fields in the loaded form set using a DAL script. You can update the ARCRET record members passed to DAL as the script executes.<br>The /LOAD parameter names a script the system will execute after the form set is retrieved from archive and loaded. DAL had the ability to query form set field values as well as other form set members. Any changes the script makes to the loaded document are temporary, as the actual archived document cannot be changed. However, the script can change ARCRET member values, which represent the archive index record used to identify the document.<br>By allowing DAL to update the ARCRET record members, the script can change the current record values that will be written to the output files generated by ARCRET. This may later influence any plug-in functions called via ARCRET. |
| /RUNDATE | Lets the user to specify a field to use instead of RUNDATE. The format of the field you designate should be YYYYMMDD. |
| /DATEFMT | Identifies the date format used within the RUNDATE field. If you omit this parameter, the utility assumes the RUNDATE field is stored in the typical YYYYMMDD format.<br>The DATEFMT must be specified using a standard internal FAP date format. Additionally you can use the format X to indicate that the RUNDATE field is stored using the Skywire Software Hex Time format.<br>If a PPS user has an archive that doesn't specify a RUNDATE field but instead uses the WIP CreateTime as the date field, it is usually stored in a Hex Time format. To do a retrieval from this archive, you would specify a command line with these parameters:<br>`ARCRET /RUNDATE=CreateTime /DATEFMT=X`<br>The /RUNDATE parameter is an existing parameter that identifies an alternative field to use as the RUNDATE. The /DATEFMT parameter indicates how to interpret the date information in the specified RUNDATE field. |

Transaction sets are passed to the plug-in functions you define in the INI file for additional processing. If no plug-in functions are defined, nothing happens to the retrieved records. All temporary files created during the run are deleted when the utility stops.

### INI files

By default, the utility loads the FSIUSER.INI and the INI file defined by this option in the FSIUSER.INI file:

```
< Environment >
    FSISYSINI =
```

To specify a different INI file, use the INI parameter as shown here:

```
ARCRET /INI=My.INI
```

### Defining plug-in functions

Define the plug-in functions in the ArcRet control group as shown here:

```
< ArcRet >
    PlugInMod   = NAME.DLL
    PlugInFunc  = Function
    PlugInFunc  = Function2
    PlugInFunc  = Function3
    PlugInFunc  = Function4
    PlugInFunc  = Function5
```

The PlugInMod option defines the DLL file that contains the functions you want to use. The PlugInFunc option defines the function name of a plug-in compatible function.

You can define up to five plug-in functions which will be executed in the order they are defined in the INI file. If no functions are defined, the utility displays this message:

```
Warning: No plug-ins loaded.
```

**NOTE:** The ARCRET utility will continue to run even if no plug-in functions are defined.

Plug-in functions must conform to a specific prototype and will be passed specific information about the files to process.

You can also use the DLL->FunctionName method for naming plug-ins. This lets you keep plug-ins in multiple DLL files. Here is an example:

```
PlugInFunc = DLL->FunctionName
```

If the option does not contain the "->" to indicate a DLL name is specified, the system assumes it should use the PlugInMod option to locate the DLL for this function.

### Archive index field mapping

The ARCRET utility is designed to find all the relevant information about your archive setup under the ArcRet control group. In general, the INI options required by the ARCRET utility are almost identical to those used by the GenArc program.

With a few exceptions, the settings referenced are the almost the same as those used and required by the AFEMAIN program. One exception is the Trigger2Archive control group.

The ARCRET utility uses this Trigger2Archive control group to re-create the NEWTRN file. If you use the GenArc program to produce your archives, you already have this group to map the fields from the NEWTRN file to the APPIDX (application index) file.

Since the AFEMAIN program does not use this control group, you may have to add it to your INI file. Here is an example of what you would need to add:

```
< Trigger2Archive >
    ArcField    = TrnField
    ArcField2   = TrnField2
    ...
```

Each *Trn* field is defined in the TRNDFDFL.DFD file which is specified using the TrnDFDFile option in the Data control group. Each *Arc* field is defined in the APPIDX.DFD file which is specified using the AppIdxDFD option in the ArcRet control group.

If you do not have the Trigger2Archive control group defined, the ARCRET utility tries to match the fields in the TRNDFDFL.DFD and APPIDX.DFD files by name. If no field names match, an error message appears and the ARCRET utility stops.

### Skipping rows/records

By default, the ARCRET utility selects each archive index record in sequence. Since you may not always want this, several parameters are included so you can designate which rows/records you want to process.

Keep in mind that the ARCRET utility presumes every index record is a potential candidate for selection.

The first record in the first index file is read and checked against the selection criteria (unless you included the /REV parameter). The next record is then read and so on until all the records in the index have been examined. Each record is either accepted or rejected based on the parameters you specify.

You can use the /R parameter to skip a specific number of sequential transactions.

You can also use the /N parameter to indicate a maximum number of *accepted* transactions you want to process. By default, the ARCRET utility continues until the last index row/record is processed.

You use the /R and /N parameters to control where and when to stop processing. The /REV parameter lets you specify whether the utility should start at the beginning or the end of the file. These parameters can be useful when you have a large number of rows/records that will take a long time to process.

For example, suppose you can only process 1000 transactions a day with the plug-in you want to use and you have 3000 transactions in your archive. Potentially, that means it would take three days to process these transactions using your plug-in.

Assuming you are not using any other parameters, you would enter these commands:

```
Day 1: ARCRET /N=1000
Day 2: ARCRET /N=1000 /R=1000
Day 3: ARCRET /R=2000
```

Notice, that the first day, you did not have to specify the /R command, but you did specify to stop after processing 1000 transactions. The second and third days, you did specify how many leading transactions to skip. Note that on the final day, the /N parameter was omitted. If you know the remaining set of records will not exceed your maximum, you can omit this parameter.

### Controlling the number of transactions sent to the plug-ins

Each time a set of matching transactions are located and retrieved, the ARCRET utility calls the plug-in functions to process those transactions. By default the ARCRET utility searches until a single matching transaction is found, retrieves the associated form set, and then calls the plug-ins. Therefore, the default *set* size is one transaction.

For some plug-in functions, a larger set size will improve performance—especially if the plug-in has excessive startup or shutdown time requirements. Use the /S parameter to designate the number of rows/records to include in a set before the ARCRET utility calls the plug-in functions.

**NOTE:** If there are not enough matching transactions found in the index file, the plug-ins are called with however many matching transactions were found. If you create a plug-in, keep in mind the plug-in should make no assumptions about the number of transactions in the sets.

You can also use the /P parameter with sets. This parameter tells the utility to pause as each set is processed and wait until you press ENTER before building the next set. This parameter is useful if you need to examine or copy the output files produced by the plug-ins before starting the next set.

## Selecting records by date

There are several parameters you can use to select transactions which fall within or outside a given date range. When specifying dates for these parameters, be sure to use the YYYYMMDD format.

Keep in mind that these parameters assume the RUNDATE or CREATETIME variables are the names of the transaction date fields. The utility first looks for RUNDATE. If it is not found, the utility looks for CREATETIME (the name used by standard AFEMAIN archives).

If the utility finds RUNDATE it looks for the record which should be in YYYYMMDD format. When using CREATETIME, the utility assumes the date is in the internal HEXTIME format and converts it to YYYYMMDD format, comparing it to the dates specified. If neither field is found, the ARCRET utility displays an error and stops processing.

**NOTE:** For the remainder of this topic, assume that RUNDATE means either RUNDATE or CREATETIME and that the data value will be in YYYYMMDD format.

Use the /BEF parameter to tell the utility to select records with a RUNDATE value that falls *before* a given date. Transactions with the specified date are excluded. For instance, to select transactions archived before 2001, you would specify:

```
/BEF=20010101
```

No records after December 31, 2000 are selected.

To select transactions with a RUNDATE that *falls* after a given date, use the /AFT parameter. Transactions with the specified date are excluded. For instance, to select the transactions archived in 2001, specify:

```
/AFT=20001231
```

All records after December 31. 2000 are selected.

You can also use these parameters together to specify a range. When you use both parameters, the utility assumes you want a logical AND comparison, so the /BEF date should fall after the /AFT date. For example, to select all records for the year 2000, you would specify these parameters:

```
ARCRET /AFT=19991231 /BEF=20010101
```

This tells the utility to select all transaction with a RUNDATE that falls within the two dates. You can also include the /OR parameter to omit records that fall within a certain range.

When you use the /BEF, /AFT, and /OR parameters, the utility excludes the transactions which fall within the dates specified. For example, to select all records except those which fall in the year 2000, you would specify these parameters:

```
ARCRET /BEF=20000101 /OR /AFT=20011231
```

Note that when you use the /OR parameter, you identify the earlier date using the /BEF parameter and the later date with the /AFT parameter.

**NOTE:** Where you place the parameters in the command does not matter.

The ARCRET utility tries to validate the date ranges you specify based on the parameters you enter. If the utility detects a combination of parameters or values that do not make sense, it displays an error message and stops.

### Using DAL to select records

You can also include a DAL script to provide the final approval or rejection of a particular transaction. Note that the utility processes all other parameters before it executes the DAL script.

The /DAL parameter names the DAL script you want to execute on each transaction. Before calling the script, the APPIDX record variables are converted into DAL variables using the standard DAL DB naming convention.

The DAL DB nomenclature associates all the row/record variable names with a table name. The table name is typically specified in the script or is the name of the table being referenced. Because DAL does not actually open the database and because there may be more than one index file, all record members are associated with the name *ARCRET*.

For instance, suppose your APPIDX has these members specified in the DFD file:

```
< Fields >
    FieldName = Key1
    FieldName = Key2
    FieldName = PolicyNum
    FieldName = RunDate
    FieldName = ArcKey
    FieldName = FormsetID
```

These fields would be referenced using the following names in any script associated with the ARCRET utility.

```
ArcRet.Key1
ArcRet.Key2
ArcRet.PolicyNum
ArcRet.RunDate
ArcRet.ArcKey
ArcRet.FormsetId
```

Note that since DAL only supports STRING, LONG, and DECIMAL data types, some variables may be converted to the closest matching type. When in doubt, assume the data will be of the STRING type for comparison purposes.

Since DAL has no knowledge of how a field will be used, always use the FORMAT and DEFORMAT functions where appropriate in your DAL scripts. For DFD members that hold DATE values, be sure to use the appropriate DATE functions, such as DATE2DATE, to convert values to a standard format before comparing them.

For example, if you want to select all transactions where the `Key2` variable contains *TEXAS*, you would write a DAL script similar to this one:

```
IF ArcRet.Key2 = "TEXAS"
    RETURN("Yes");
END
RETURN("NO");
```

**NOTE:** The return value from the script is important. If the script returns *Yes*, that means to include the transaction in the set. Any other return value – including omitting a return value—excludes the transaction from the set. The case of the word Yes in the returned value is not important.

If you saved this script using the name *MATCH.DAL*, you would enter this command:

```
ARCRET /DAL=MATCH.DAL
```

For performance reasons, the utility normally pre-compiles your DAL scripts in memory before executing them. This makes each subsequent execution of the script faster than if the script was not pre-compiled. You can, however, turn off this behavior by including the /NC parameter. Typically, you would only include this parameter for debugging purposes.

You can also use the /DB parameter to debug scripts. This parameter produces lots of output, so be sure to send the console output to a file using the <">filename" parameter on the command line. Also, use the /S parameter to limit the run to only a few transactions. Otherwise, the amount of output will be overwhelming.

### Keeping intermediate files

Normally, the utility removes intermediate files after each batch set is complete and before it starts the next set. Use the /K parameter to keep these files for each batch sets. Each set of files is renamed with an extension that matches the set (batch) count. For instance, the files from the first batch of records will be renamed as *.1; the second set of files as *.2; and so on.

Be careful using this parameter. Depending upon the size of each transaction and the number of transactions retrieved, using this parameter can consume a lot of file space. Keep in mind you must remove these files when the ARCRET utility finishes.

### Batch queuing

Normally, the ARCRET utility retrieves transactions interactively while sequentially accessing your archive index systems. In some instances, however, the underlying archive system does not support retrieving transactions while sequentially processing the records.

This is the case with Documanage. Since Documanage serves as both the archive index and storage system, it is cannot currently be accessed in the same manner as a database. Therefore, use the /BQ option to queue sequential transactions in memory before attempting to retrieve the associated files.

Queuing the transactions this way consumes more memory than would otherwise be required. In general, you can determine the index record size by examining the APPIDX.DFD file you use. Add some extra for overhead and then divide that into the maximum amount of memory that you wish to consume—over and above that used by the ARCRET utility itself. To determine this, use the Windows NT Task Manager to get an idea of how much memory is in use while you run the utility on a single transaction.

Batch queuing also affects performance because the utility has to reset the sequential index search between each set of records. For instance, suppose it reads and queues 100 transactions in a set. After the set is processed, it has to re-read sequentially from the beginning of the index back down to the 100th record to prepare for the next set. For each set of transactions, this *resetting* takes longer and longer, because it has to start at the beginning of the index each time.

**NOTE:** Only use the /BQ parameter when the source archive system that ARCRET utility is reading does not support the interactive retrieval of documents while sequential reading the index records.

## Using the PLGTest Plug-in

This plug-in prints a few messages each time it is called. Use it for testing purposes only. For instance, you would use this plug-in before using one of the other plug-ins to make sure you are successfully retrieving records and that those records match your criteria.

You can include the /P (pause) option to temporarily stop processing between batches. This lets you examine the intermediate files.

To use this plug-in, make sure you have these options in your INI file:

```
< ArcRet >
    PlugInMod = PLGW32.DLL
    PlugInFunc= PLGTest
```

## Using the PLGGenPrint Plug-in

This plug-in accepts output from the ARCRET utility and executes the GenPrint program to produce printed output of archived transactions.

**NOTE:** Before you use this plug-in, make sure you have a batch setup that can run the GenPrint program to produce the printed output you want.

To specify this plug-in, include these INI settings:

```
< ArcRet >
    PlugInMod  = PLGW32.DLL
    PlugInFunc = PLGGenPrint
```

Note that case *is* important when specifying the name of the plug-in. You must specify it *exactly* as shown here.

The ARCRET utility produces an NA/POL file set similar to that produced by the GenData program. This includes a NewTrn file that is created from the archive index using the options in the Trigger2Archive control group. The plug-in then converts the NewTrn file into a batch (RCBDFDFL.DFD) compatible file and starts the GenPrint program to complete the process.

The conversion from the NewTrn type output of the ARCRET utility to a recipient batch file is accomplished by matching the field names between the two files. No addition INI options are necessary. The utility simply uses the TRNDFDFL.DFD file and the RCBDFDFL.DFD file (or whatever names you give these files in your INI settings) and match the fields by name. The utility automatically handles any other necessary conversions.

The plug-in produces a recipient batch record for each recipient found in the form set associated with the transaction. Because only one batch file is produced, all recipients are written to the same file.

Once the batch file is produced, the utility internally re-maps the INI options for the following items, writes a temporary INI file, and then executes the GenPrint program specifying that INI file to produce the batch output.

```
< Environment >
    FSISYSINI   =
< Print_Batches >
    ...
< Exclude_Batches >
    ...
< Data >
    NAFile      =
    POLFile     =
    NewTrn      =
```

| Option | Changes |
|---|---|
| Environment control group | |
| FSISYSINI | This option is eliminated. All of the current options from both INI files were placed in memory while executing the ARCRET utility, so the file is not necessary in the subsequent run of the GenPrint program. |
| Print_Batches control group | |
| all options | All batches defined under this group are eliminated except for the first alphabetical group. The utility uses the first option as the batch to be processed. Make sure the printer settings for this batch are appropriate. The utility changes the name of the BCH file associated with this option to match the file produced by the plug-in. |
| Exclude_Batches control group | |
| all options | The utility eliminates all of the options in the control group to make sure the one batch left intact is processed. |
| Data control group | |
| NAFile | This option is changed to match the output from the ARCRET utility. |
| POLFile | This option is changed to match the output from the ARCRET utility. |
| NewTrn | This option is changed to match the output from the ARCRET utility. |

In general, all the temporary files that are produced have the same (8-digit hexadecimal) name and a different extension. This includes the temporary INI file the utility produces for the GenPrint program.

If GenPrint errors occur, remember the output of the GenPrint program is written to the error file specified in the Data control group and is not controlled by the plug-in or the ARCRET utility.

## Using the PLGGenArc Plug-in

This plug-in accepts output from the ARCRET utility and executes the GenArc program to archive your transactions to another system.

**NOTE:** For this discussion, *source* refers to archive system where the ARCRET utility is reading transactions. *Destination* refers to the new archive system where the GenArc program will save transactions retrieved from the *source*.

Before you use this plug-in, make sure your system can successfully...

- Retrieve transactions from your *source* archive system. This involves INI options in a file which this discussion refers to as the *FSIOLD.INI* file.
- Add transactions to your *destination* archive system. This involves INI options in a file which this discussion refers to as the *FSINEW.INI* file.

Make sure the TriggerToArchive control group is in the (source) FSIOLD.INI file and that you map the archive index fields defined in the APPIDX.DFD file to the corresponding fields in the TRNDFDFL.DFD file. The field names between the two DFD files don't have to be the same. In the FSIOLD.INI, add these options:

```
< ArcRet >
    PlugInMod   = PLGW32.DLL
    PlugInFunc  = PLGGenArc
< GenArcPlugIn >
    INIFile     = FSINEW.INI
```

The options in the ArcRet control group identify the plug-in function the ARCRET utility will use. The INIFile option in the GenArcPlugIn control group identifies the INI file that should be supplied to the GenArc program by the plug-in.

Use a command similar to this one to run the ARCRET utility:

```
ARCRET /INI=FSIOLD.INI
```

This tells the ARCRET utility to use the correct INI file to retrieve from your source archive. When the plug-in is called, the batch files created by the ARCRET utility are automatically added to the INI file you named. In part, the changes will be to several options in the Data control group, such as those shown here:

```
< Data >
    NAFile  =
    NewTrn  =
    POLFile =
```

The plug-in then starts a GenArc session and sends to it a temporary INI file created by combining these options. If you have successfully tested your GenArc program setup using the INI file you named, it should work without further intervention.

## EXAMPLES

These examples begin with the base FSIUSER.INI file in RPEX1 and use the standard xBase and CARFile archives. Here is an example of the ArcRet control group:

```
< ArcRet >
    AppIdx      = ARC\AppIdx
    AppIdxDfd   = DefLib\AppIdx.Dfd
    ARCPath     = [CONFIG:Batch Processing] ARCPath =
    Arrangement= Stack
    CARFile     = ARCHIVE
    CARPath     = [CONFIG:Batch Processing] CARPath =
    Catalog     = ARC\Catalog
    ExactMatch  = No
    Key1        = Company
    Key2        = Lob
    KeyID       = PolicyNum
    LBLimit     = 500
    PlugInFunc  = PLGGenArc
    PlugInMod   = PLGW32
    TempIDX     = ARC\Temp
```

Only those changes required to get each archive system to work are shown.

### ODBC/SQL server changes

Make these changes. Options not listed remain the same as those in the standard ArcRet control group from RPEX1. For SQL, you must change the CARData column from the default VARCHAR to a BLOB. You must also have a special restart table DFD to change the LASTRECORD column VARCHAR length.

```
< DBHandler:ODBC >
    CreateTable= Yes
    CreateIndex= No
    InstallMod = sqw32
    InstallFunc= SQInstallHandler
    UserID     = sa
    Passwd     =
    Server     = MS SQL FSINTSRV07
< DBTable:CATALOG >
    DBHandler   = ODBC
    UniqueTag   = catalogID
< DBTable:APPIDX >
    DBHandler   = ODBC
< DBTable:ARCHIVE >
    DBHandler   = ODBC
    UniqueTag   = arckey
< DBTable:RESTART >
    DBHandler   = ODBC
< Archival >
    ArchiveMem  = Yes
< ArcRet >
    RestartTable= Restart
    CARFileDFD  = carfile.dfd
    RestartDFD  = restart.dfd
```

### DB2 changes

Make these changes. Options not listed remain the same as those in the standard ArcRet control group from RPEX1.

```
< DBHandler:DB2 >
    BindFile    = ..\dll\db2lib.bnd
    CreateTable= Yes
    CreateIndex= No
    Database    = ARCDBL
    UserID      = user
    Passwd      = admin
< DBTable:CATALOG >
    DBHandler   = DB2
< DBTable:APPIDX >
    DBHandler   = DB2
< DBTable:ARCHIVE >
    DBHandler   = DB2
< DBTable:RESTART >
    DBHandler   = DB2
< Archival >
    ArchiveMem  = Yes
< ArcRet >
    RestartTable= Restart
```

### Documanage changes

Make these changes. Options not listed remain the same as those in the standard ArcRet control group from RPEX1.

```
< Archival >
    ArchiveMem      = Yes
    UseRestartTable = No
< DBHandler:PO >
    Cabinet         = RPEX1
    Domain          = docucorp
    Password        = password
    UserID          = user
< DBTable:APPIDX >
    DBHandler       = PO
< DBTable:ARCHIVE >
    DBHandler       = PO
< PO:RPEX1 >
    FileType        = DAP
    FolderBy        = Company,Lob,PolicyNum
    NameDocBy       = ARCKEY
```

# ARCSPLIT

Use the ARCSPLIT utility to back up all or part of your archives or split archive data based on a cut-off date or on a DAL script. Use the ARCMERGE utility to combine an archive split with this utility.

**NOTE:** This utility only works with xBase files. It does not work on SQL, Oracle, or DB2 databases.

You can use the following parameters or INI options or a combination of both to control how this utility splits archives.

## Program names

Windows ARCSPLIT.EXE

## Syntax

**ARCSPLIT /D /DAL /ED /INI /L /LOG /N /NCF /NSCF /NSIF /O /P /R /SD /SET /SRCH**

| Parameter | Description |
|---|---|
| /D | (Optional) Include this parameter if you want the utility to delete any existing split archive files before it creates new ones. |
| /DAL | (Optional) Enter the name of the DAL script you want to use.<br>This parameter overrides the DALScript and RunDALScript INI options. |
| /ED | Enter the date after which to end the search, in YYYYMMDD format.<br>This parameter overrides the SplitToDate INI option. |
| /INI | Enter the name of the INI file you want the utility to reference. If you are using the default INI file (FSIUSER.INI) and it is in the current directory, you do not have to specify it. |
| /L | (Optional) Enter the size limit for the new CAR files the utility creates. For instance, it you want to set the maximum size at 100KB, enter *1*. If you want to set the limit at 1,400,000KB, enter *1400*.<br>This parameter overrides the EnableCARFileSize and CARFileSize options. |
| /LOG | (Optional) Enter the name of the file into which the utility should write error messages.<br>This parameter overrides the LogFile INI option. |
| /N | (Optional) Enter the number of records you want to process.<br>This parameter overrides the RecordsToProcess INI option. |
| /NCF | Enter the full name you want to assign to the new catalog file. The utility defaults to the current MRL path.<br>This parameter overrides the SplitCatalog INI option. |
| /NSCF | Enter the full name of the newly split archive (CAR) file. The utility defaults to the current MRL path.<br>This parameter overrides the SplitCARFile INI option. |

| Parameter | Description |
|---|---|
| /NSIF | Enter the full name of the newly split index (IDX) file. The utility defaults to the current MRL path.<br>This parameter overrides the SplitAppIdx INI option. |
| /O | (Optional) Use this parameter to override the required date range when splitting an archive |
| /P | (Optional) Purge from the master archive the records which are split. If you omit this parameter, the specified records are written to the directory defined in the INI options and the master archive is not affected.<br>This parameter overrides the PurgeRecords INI option. |
| /R | Enter the number or records to skip before processing occurs. The default is zero which tells the utility to start with the first record.<br>This parameter overrides the RecordsToSkip INI option. |
| /SD | Enter the date on which to start the search, in YYYYMMDD format.<br>This parameter overrides the SplitFromDate INI option. |
| /SET | Use to specify which configuration settings to use. You can have multiple configuration settings in the INI file. Keep in mind that command line parameters override INI settings, except/SET. |
| /SRCH | Enter *RunDate* to search the records based on the RunDate. Enter *ArchivedDate* to search the records based on the date on which they were archived.<br>This parameter overrides the SearchDateBy INI option. |

This utility uses the current master resource library (MRL) as the base for splitting the archive. If the archive you intend to split is not part of the current MRL, you will get incorrect output.

## INI options

In addition to parameters, you can also set up INI options to use this utility. The utility looks in the INI file defined by the /INI parameter for these options. If you omit the /INI parameter, the utility looks in the FSIUSER.INI file.

These INI options are required. These options tell the ARCSPLIT utility where to find the source archive files you want to split.

```
< ArcRet >
    AppIdx  = arc\appidx.dbf
    ArcPath = arc\
    CARFile = archive
    CARPath = arc\
    Catalog = arc\catalog
```

The utility also looks for options that specify the names of the split archive files and provide other information. These options are located in the ArcSplit and ArcSplitConfig control groups:

```
< ArcSplit >
    ArcSplitConfig     =
    DefConfig          =
< ArcSplitConfig:TEST1 >
    SplitDays          =
```

```
RunDALScript       =
DALScript          =
RecordsToProcess   =
RecordsToSkip      =
SearchDateBy       =
SplitAppIdx        =
SplitCARFile       =
SplitCatalog       =
SplitFromDate      =
SplitToDate        =
PurgeRecords       =
CARFileSize        =
EnableCARFileSize  =
AllowChanges       =
LogFile            =
```

| Option | Description |
|---|---|
| ArcSplit | |
| ArcSplitConfig | Lets you assign a name to your archive split settings. You can create as many ArcSplitConfig options as you need. Each group of settings represents a control group with individual options. Below is an example of a setting named *TEST1*. |
| DefConfig | For the ARCSPLIT utility, if you define the DefConfig option and omit the /SET parameter, the utility uses the DefConfig option to look for the ARCSPLIT configuration settings.<br>If you set up multiple ArcSplitConfig options for Documaker Workstation, you can use this option to designate a default. |
| ArcSplitConfig:*TEST1* | |
| SplitDays | (Optional) This option is not used by the ARCSPLIT utility, but is used by Documaker Workstation. Enter the number of days you want the utility to add to the start date to determine the end date. The utility then splits the archive between the start and end dates. |
| RunDALScript | (Optional) Enter Yes if you want the utility to run the DAL script you specified with the DALScript option.<br>The /DAL parameter overrides this option. |
| DALScript | (Optional) Enter the name of the DAL script you want to run.<br>Use the RunDALScript option to tell the utility if it should run the DAL script you specify with this option.<br>The /DAL parameter overrides this option. |
| RecordsToProcess | (Optional) Enter the number of records to process. Use this option if you have a very large archive and you want to limit the number of records processed at one time.<br>The /N parameter overrides this option. |
| RecordsToSkip | (Optional) Enter the number of the record the utility to process first. If you omit this option, the utility starts with the first record.<br>The /R parameter overrides this option. |

| Option | Description |
|---|---|
| SearchDateBy | (Optional) Enter *RunDate* to search the records based on the RunDate. Enter *ArchivedDate* to search the records based on the date on which they were archived.<br>The /SRCH parameter overrides this option. |
| SplitAppIdx | (Optional) Enter a name for the newly split IDX file, such as APPIDX1.<br>The utility stores the data in the file you specify. If the file exists and you have set the /D parameter, the utility overwrites the file. If you have not set the /D parameter and the file exists, the utility stops.<br>The /NSIF parameter overrides this option. |
| SplitCARFile | (Optional) Enter a name for the newly split CAR file, such as ARCHIVE1.<br>The utility stores the data in the file you specify. If the file exists and you have set the /D parameter, the utility overwrites the file. If you have not set the /D parameter and the file exists, the utility stops.<br>The /NSCF parameter overrides this option. |
| SplitCatalog | (Optional) Enter a name for the newly split catalog file, such as CATALOG1.<br>The utility stores the data in the file you specify. If the file exists and you have set the /D parameter, the utility overwrites the file. If you have not set the /D parameter and the file exists, the utility stops.<br>The /NCF parameter overrides this option. |
| SplitFromDate | (Optional) Enter the date on which you want the split to begin. The default is the current date. The format is MM/DD/YYYY.<br>The /SD parameter overrides this option. |
| SplitToDate | (Optional) Enter the date on which you want the split to end. The default is the current date. The format is MM/DD/YYYY.<br>The /ED parameter overrides this option. |
| PurgeRecords | (Optional) Enter Yes if you want the utility to purge from the master archive the records it split from the archive.<br>The default is No, which tells the utility to copy but not delete those records. The utility stores the copied records in the files and directories you specified.<br>The /P parameter overrides this option. |
| CARFileSize | Enter a number between one (1) to 14,000 to define the size for the CAR file. If you enter one (1), the utility interprets that as 100,000 bytes (100 KB). If you enter 14,000, the utility interprets that as 1,400,000,000 bytes (1,4000,000 KB).<br>Omit this option if the EnableCARFileSize option is set to No. |

| Option | Description |
|---|---|
| EnableCARFileSize | (Optional) This option turns on and off the related radio button field on the ArcSplit window in Documaker Workstation and also tells both Documaker Workstation and the ARCSPLIT utility whether to use the CARFileSize option. The default is No.<br>The /L parameter overrides this option. |
| AllowChanges | (Optional) Lets you change settings from a window. The default is No. |
| LogFile | (Optional) Enter the name of the file into which the utility should write any error messages. Here is an example:<br>`LogFile = c:\errlog.txt`<br>The /LOG parameter overrides this option. |

## Using ARCSPLIT with DAL scripts

You can use DAL scripts to when you split or back up an archive. The ARCSPLIT utility makes available to the DAL script all of the APPIDX column values you specify. This script can only return Yes or No. If anything else is returned, the system defaults to No.

All field names specified in the script file must include the word *ARCSPLIT*, such as *ARCSPLIT.KEY1*. This is required in case multiple index files are in use.

**NOTE:** Refer to the DAL Reference for information on the DAL functions you can use to create the scripts.

Assume the following form sets are stored in the archive, the INI options are set as shown below, and the DAL scripts COMPANY.DAL and COMBINED.DAL exist in the DefLib directory.

**Table 1:**

| KeyID | Key1 | Key2 | Date archived |
|---|---|---|---|
| AA | Sampco | LB1 | 03/01/1999 |
| BB | Sampco | LB2 | 03/01/1999 |
| CC | FSI | GL | 03/02/1999 |
| DD | FSI | GF | 03/02/1999 |
| EE | MyCompany | GO | 03/03/1999 |

Also assume these INI options are set:

```
< ArcRet >
    SplitAppIdx    = arc1\AppIdx1.dbf
    SplitCARFile   = arc1\Archive1.car
    SplitCatalog   = arc1\Catalog1.
```

The COMPANY.DAL script looks like this:

```
If ARCSPLIT.Key1 = “FSI” then Return (“Yes”);
                          Else Return (“No”);
End
```

The COMBINED.DAL script looks like this:

```
if (ARCSPLIT.KEY1 = "FSI " AND ARCSPLIT.KEY2 = "GL ") then
Return ("YES");
Else
Return("NO");
End
```

**NOTE:** Make sure the value you specify matches the field length defined in the DFD (Database Field Definition) file. In this example, the field length of KEY1 is four characters and the search value should be “FSI “ (with a space between I and the ending quotation mark) instead of “FSI”.

Based on these assumptions, this table shows the results if you enter the following commands to run the ARCSPLIT utility:

| If you enter... | The result is... |
|---|---|
| ARCSPLIT /sd=19990301 / ed=19990301 / ini=fsiuser.ini | Records AA and BB are written to the archive files (ARCHIVE.CAR, APPIDX1.DBF, CATALOG1.DBF, APPIDX1.MDX, and CATALOG1.MDX) in the arc1 directory. The records in the master archive are not changed. |
| ARCSPLIT /sd=19990301 / ed=19990301 / ini=fsiuser.ini | Records AA and BB are written to the archive files in the arc1 directory. The records in the master archive are not changed. The APPIDX1 and CATALOG1 files will be flat files. |
| ARCSPLIT /sd=19990301 / ed=19990301 / ini=fsiuser.ini /p | Records AA and BB are written to the archive files in the arc1 directory. These records are also deleted from the master archive. |
| ARCSPLIT /dal=deflib\company.dal ini=fsiuser.ini | Records CC and DD are written to the archive files in the arc1 directory. The records in the master archive are not changed. |
| ARCSPLIT /dal=deflib\combined.dal ini=fsiuser.ini | Record CC is written to the archive files in the arc1 directory. The records in the master archive are not changed. |

See also ARCMERGE on page 24

# ARCVIEW

You can use the ARCVIEW utility to view Documaker archive files checked into the Documanage archive system. This utility only runs under 32-bit Windows.

## Program names

| | |
|---|---|
| Windows | ARCVWW32.EXE |

You do not run this utility from the command line. Instead, you simply register this utility as the program you want to use to view Documanage files. To use this utility, follow these steps:

1. Register the Documanage file extension (DPA) in Windows so the operating system will automatically use the ARCVIEW utility to view these files.
2. Set the environment variable FSIPATH to point to the directory where the INI file for the AFEMAIN program is stored. Here is an example:

   ```
   FSIPath = d:\rpex1
   ```

**NOTE:** The AFEMAIN program is the executable file for Documaker Workstation.

3. Place a menu file, similar to the MEN.RES file used by Documaker Workstation, be in the directory specified by the FSIPath option. The name of the menu file should be *ARCVIEW.RES*.

**NOTE:** You can edit this file to remove functionality you do not want to include.

4. Edit the FILETYPES.INI file on the computer where the Documanage server runs. Add the DPA file extension to the list of file types to view with the ARCVIEW.EXE program. This causes the Documanage client to use the viewer registered in Windows instead of the default Documanage viewer.

You can now click on Documaker archive files in Windows Explorer to display them.

# ATPHDR

Use the ATPHDR utility to add missing header information for Xerox fonts if you experience this problem when you insert Xerox fonts into a font cross-reference file (FXR) using the Font Manager.

**NOTE:** Docuview, which is part of Docusave workstation, requires Xerox resources to be padded to fill 512 byte blocks. Some old Xerox resources built with prior versions of the system do not meet this criteria. You can use this utility to read and update those Xerox resources so they can be used by Docuview.

## Program names

| | |
|---|---|
| Windows | ATPHDRW.EXE |

## Syntax

**ATPHDRW /I**

| Parameter | Description |
|---|---|
| /I | Enter the input file name. Omit the extension. |

The input file must have the extension *TMP*. The utility will create an output file with the extension *FNT*.

# BARR2MVS

Use the BARR2MVS utility when you have a Xerox Metacode spool file which is one long record and you want to separate it into separate records for MVS.

**NOTE:** This program is only available on MVS.

Metacode print spools which are created on PCs and which use JES2 format, contain carriage return/line feeds (CR/LF) at the end of each record. However, some Metacode print spools created on a PC may contain binary data which happens to contain a carriage return/line feed. In this case, uploading the spool file with CR/LF translation will produce an invalid Metacode print spool file on MVS.

In this case, create the Metacode print spool using the BARR format, upload the original Metacode print spool file as binary, with no CR/LFs, and use the BARR2MVS utility to convert the BARR formatted print spool file into a Metacode print spool file with separate records on MVS.

## Example

Here is an example to show you how to do this:

First create a PS (Physical Sequential) file on MVS with DCB=U,0,23200. Name this file *WINFILE*. Then, create another PS file with DCB=VB,600,23200 and call it *MVSFILE*. The BARR2MVS utility uses *DD:RSCWIN* as the DD name for the original Metacode print spool file and uses *DD:RSCMVS* as the DD name for the newly-created Metacode print spool file with separate records.

# BARR2VB

Use this utility to convert a Metacode print stream from a BARR record format to a Variable Block (VB) format.

Variable block format Metacode records are used in Docuview LFS, which lets you view a Metacode print stream, and in JES Commander, which lets you upload Metacode print streams to MVS or OS/390.

## Program names

| | |
|---|---|
| Windows | BARR2VB.EXE |

## Syntax

**BARR2VB /I /O /D /P**

| Parameter | Description |
|---|---|
| /I | Enter the name of the input file in BARR format. |
| /O | Enter the name of the output file in VB format. |
| /D | (Optional) Include this parameter to add Docusave comments. |
| /P | (Optional) Use this option to specify the PrtType control group you want the utility to use from the INI file. For example, you could enter *XER*. |

Use the /D parameter to add a dummy Docusave record to the variable block format print stream.

Your FSISYS.INI file should include a printer control group, such as PrtType:XER, which contains the options used to produce the BARR formatted Metacode print stream. This information is necessary for BARR2VB to read the BARR formatted Metacode print stream.

# BARRWRAP

Use this utility to reformat MVS generated Metacode output for submission to a BARR system for Xerox printing on a local area network.

## Program names

| | |
|---|---|
| Windows | BARRWW32.EXE |
| MVS | See the Documaker Server Installation Guide |

## Syntax

**BARRWW32 /I**

| Parameter | Description |
|---|---|
| /I | The Metacode file. The extension must be MET, which you can omit. |

**NOTE:** You can replace the backslash (/) with a dash (-) for any parameter.

The file will be formatted for BARR spool output and the output file name will have the same name with a TXT extension.

This utility converts Metacode output from a JES2 format to a BARR format. The BARR interface attachment for Xerox Metacode printers requires Metacode print stream files to contain BARR specific information. BARRWRAP adds this information to an existing Metacode print stream file, allowing the output file to be printed via the BARR interface. After the utility is run on the Metacode file, "76 1A FF 00" will be added at the beginning of the file which informs BARR that the file is of Metacode type. A byte denoting record length is also added at the beginning and end of each record in the file.

To use this utility, you must set these INI options:

```
Environment=MVS
OutMode=JES2
```

BARRWRAP is useful when testing the GenPrint program on MVS. If the MVS system is not directly channel attached to the Xerox printer, you have to download the print data stream to a Windows system. (use no ASCII translation, but do use CR/LF). Then, using BARRWRAP, it is packaged to pass through BARR/SPOOL successfully.

**NOTE:** Occasionally, binary data contained in the Metacode file has a sequence of hex bytes x'0D0A', which could be misinterpreted as a carriage return / line feed. This is true particularly for charts and other inline graphics. Such data streams should be BARRWRAPped on the MVS platform before being downloaded with no ASCII and no CR/LF options.

# BDF2FDT

Use this utility to load a complete tree, based on the specified BDF file, and then convert it to a pre-version 11.x format FORM.DAT file.

**NOTE:** This utility is typically used by customers who have purchased tools from 3rd party vendors that parse FORM.DAT files. These tools extract a list of the forms in the library as a part of the data exchange mapping process.

## Program names

| | |
|---|---|
| Windows | BDF2FDT.EXE |

## Syntax

**BDF2FDT /I /INI /O /D**

| Parameter | Description |
|---|---|
| /I | (Optional) Enter the name of the BDF file you want to convert. If you omit this parameter, the system looks for the one in the INI file you specify using the INI parameter. |
| /INI | (Optional) Enter the name of the INI file which contains the name of the BDF file you want to convert. The default is FSIUSER.INI. |
| /O | (Optional) Enter the name of the output FORM.DAT file. The default is FORM.DAT. |
| /D | (Optional) Enter the effective date in YYYYMMDD format. The default is the current date. |

**NOTE:** Not all document features and options supported by Documaker Studio can be represented in a FORM.DAT file. You must make sure the forms you develop in Studio convert appropriately.

# CARINTEG

Use the CARINTEG utility to check the integrity of CAR archive files. This utility is built into the base system, so you may not need to run it separately if you are using version 7.5 or later.

## Program names

| | |
|---|---|
| Windows | CARIGW32.EXE |

## Syntax

**CARIGW32 /I**

| Parameter | Description |
|---|---|
| /I | Enter the name of the CAR file. |

A CAR file contains compressed NA and POL information along with a table of offsets to this information. CARINTEG queries the table of offsets to determine the number entries, counts back to the first entry in the table, then verifies that the record referenced by that offset is a valid CAR record.

CARINTEG returns a message which tells you if the CAR file is Ok or if it has errors.

# CARREN

Use the CARREN utility to rename CAR files. Typically, this is done to keep CAR file sizes at a manageable level.

## Program names

| | |
|---|---|
| Windows | CARRNW32.EXE |

## Syntax

**CARRNW32 /I /P**

| Parameter | Description |
|---|---|
| /I | Enter the name of the CAR file. |
| /P | Enter the path in which to place the renamed file. |

You must have the following two files in the working directory, along with the CAR file:

- CATALOG.DBF
- CATALOG.MDX

This utility renames the specified CAR archive file with a new, unique name. The next time archive is run, the system creates a new CAR archive file with the same name as the original CAR file.

**NOTE:** You can automate this process using INI options.

# CFA2FAP

Use the CFA2FAP utility to convert a CFA or CFX file into a FAP or FXR file. CFA files are compiled FAP files. CFX files are compiled FXR (font cross-reference files). This utility is used for debugging purposes.

When you use this utility, the result matches your original FAP file, with these exceptions:

- Version records will not be decompiled
- Some error and backward compatibility issues may have been corrected when you compiled the file so the new, decompiled file will include those changes

**NOTE:** The version of the system you use to compile the FAP and FXR files must be the same version you will use when running Documaker Server. Furthermore, the platforms must also match. For instance, if you compile the FAP and FXR files on version 11.0 for Windows, to use them in Documaker Server (GenTrn, GenData, GenPrint), you must run version 11.0 for Windows of Documaker Server.

## Program names

| | |
|---|---|
| Windows | CFA2FAPW.EXE |

## Syntax

**CFA2FAPW** /I /O /F

| Parameter | Description |
|---|---|
| /I | The input CFA file or CFX file |
| /O | (Optional) The output FAP or FXR file |
| /F | Include this option to create a font cross-reference (FXR) file which contains only fonts |

Make sure the CompiledFAP option is set to *Yes* (the default is No) in the RunMode control group of your INI files before you run the system using precompiled FAP files.

In addition, use the File, Library Setup option (or edit your INI files) to specify the path for the CompLib, which is where the system will look for the compiled files. In your INI files, you will find this setting in the following control group:

```
<MasterResource>
    CompLib=(directory or library the CFA and CFX files are stored in)
```

For MVS systems, you can specify a DD name, such as DD:COMPLIB()

**NOTE:** If you are going to use pre-compiled FAP files, you must also use precompiled FXR files. Also keep in mind that you cannot upload CFA files.

# CPCNV

Use the CPCNV utility to convert the text of a flat file, record by record, from a source codepage to a destination codepage. This utility supports record lengths up to 32k.

**NOTE:** You can port this utility to other platforms.

## Program names

| | |
|---|---|
| Windows | CPCNVW32.EXE |
| MVS | See the Documaker Server Installation Guide |

## Syntax

**CPCNVW32 /S /D /I /O /N /R**

| Parameter | Description |
|---|---|
| /S | The codepage the source file is currently written in. |
| /D | The code page to convert to. |
| /I | The text file to convert (you can use asterisks as wildcards). |
| /O | (Optional) The name of the output file. |
| /N | (Optional) A nontranslatable character value, the default is 255. |
| /R | (Optional) Reads the CR/LFs in the ASCII file and converts the data into multiple records. Use this parameter if you use FTP or similar communications programs to upload files. If the program you use to upload the files converts CR/LFs into separate records, you can omit this parameter. |

Use the CPCNV utility to convert text files written in one codepage to another codepage. The CPCNV utility loads either the FSISYS.INI or FAPCOMP.INI file to find these FMRES control group options:

```
< FMRes >
    DefLib   = ..\MSTRRES\FMRES\DEFLIB\(default shown)
    Codepage = CODEPAGE.INI            (default shown)
```

This utility uses the CODEPAGE.INI file in your \mstrres\fmres\deflib directory. If the file is not there, the utility displays an error message. You can specify a different location by adding an INI option in the FSISYS.INI or FSIUSER.INI files, as shown here:

```
< FMRes >
    DefLib = the path where the codepage.ini is located.
```

The CODEPAGE.INI file contains information about characters in various codepages. Here is an excerpt of the CODEPAGE.INI file:

```
< Codepages >
  Codepage    = 1004,W1
  Codepage    = 863,CF
  Codepage    = 850,PM
  Codepage    = 437,PC
  Codepage    = 37,Z1
< Codepage >
  Char    =  SP010000,              space, , 32, 32, 32, 32, 64
  Char    =  SP020000,             exclam, , 33, 33, 33, 33, 90
  Char    =  SP040000,           quotedbl, , 34, 34, 34, 34,127
(and so on)
  Char    =  LA160000,         Acircumflex, ,194,132,182,  0, 98
  Char    =  LA150000,         acircumflex, ,226,131,131,131, 66
(and so on)
```

The first control group, CODEPAGES, lists the codepages specified by this file. The two character abbreviation, following the codepage number, specifies an internal character set name used by some system font conversion utilities.

The second control group, CODEPAGE, specifies character names and their associated code points in the different codepages listed in the first control group. The first column is the name of the character when printing to an AFP printer. The second column is the name of the character when printing to a PostScript printer. The third column is the name of the character when printing to a TrueType printer (not currently supported). The remaining columns correspond to the code points for the codepages specified in the first control group.

In the example above, there are five codepages specified: 1004, 863, 850, 437, and 37. Therefore, the last five columns represent code points for codepages 1004, 863, 850, 437, and 37 respectively.

For example, the first character defined (the space character) has these attributes:

```
Char    = SP010000,               space, , 32, 32, 32, 32, 64
AFP name = SP010000
PostScript name = space
TrueType name = (blank, not used)
1004 code point = 32
863 code point = 32
850 code point = 32
437 code point = 32
37 code point= 64
```

where as the A-circumflex (Â) character has the following attributes:

```
Char    = LA160000,         Acircumflex, ,194,132,182,  0, 98
AFP name = LA160000
PostScript name = Acircumflex
TrueType name = (blank, not used)
1004 code point = 194
863 code point = 132
850 code point = 182
437 code point = 0 (not defined for this codepage)
37 code point = 98
```

Therefore, if you were to convert a text file built using codepage 863 (Canadian French) to codepage 1004 (ANSI/Skywire Software standard), any space characters would remain unchanged but any A-circumflex (Â) characters would change from a 132 to a 194 code point.

See Using Fonts, in the Documaker Server System Reference for more information about fonts and codepages.

## Understanding the System

Sometimes you need to transfer files between the PC and host (MVS) platforms. There are a number of products that provide this capability. For instance, you can use FTP to transfer files from the PC to the host (MVS). FTP can transfer a file using a *binary* or *text* mode.

- Binary mode means do not translate the characters contained in the file.
- Text mode means translate the characters from ASCII on the PC to EBCDIC on the host.

Text mode will also write a separate record on the host for each carriage return line feed combination found in the original PC file.

Unfortunately, the Text mode ASCII to EBCDIC translation used by FTP does not match our standard ASCII to EBCDIC translation for extended ASCII characters (code points 128 and above contain international characters, some currency and punctuation symbols, and so on). Therefore, if system resource files to be uploaded contain some of these extended ASCII characters, the files must be uploaded as Binary.

If a system resource file, such as a FAP, FXR, or DDT file is uploaded as binary, not only do you need to convert the ASCII characters to EBCDIC, you also need to create records that correspond to each line of text in the PC file.

On the PC, the end of a line is indicated by a couple of control characters, specifically a carriage return character followed by a line feed character (CRLF). Since the PCL file was uploaded as binary, the file still contains the CRLF characters. Use the /R parameter to treat the CRLFs as an end of line indicator when converting the file on MVS.

Some products, including IBM's Personal Communication, provide the ability to upload a file as binary but will translate CRLFs into separate records when uploading the file. Do not use the /R parameter if you use a file transfer program that converts lines ending with CRLFs into separate records when uploading to MVS.

# CSET2FAP

Use the CSET2FAP utility to convert Document Sciences CompuSet scripts into Skywire Software FAP files. This utility does conversions as part of a batch process.

**NOTE:** Using this utility reduces the work necessary to convert a Document Sciences resource base into a Skywire Software resource base. These commands are supported:

| Command | Description | Command | Description |
|---|---|---|---|
| BD | bold | BL | blank line |
| BOX | create a box | CB | change baseline |
| CC | center on column | COLOR | set color |
| CP | center on page | CW | column width |
| DL | dot leader | EL | end underline |
| F | select font | HR | horizontal rule |
| JU | horizontal justify | LT | non bold |
| NL | new line | NP | new page |
| NPR | new page recto | PA | set paragraph indent |
| PD | page depth (height) | PT | point size |
| PW | page width | QL | quad left |
| QR | quad right | SK | skip text (comment) |
| SP | space vertically | T | select tab |
| TABC | set text-centered tab | TABJ | set text-justified tab |
| TABL | set quad left tab | TABR | set quad right tab |
| UL | underline | VR | vertical rule |

Keep in mind the utility does not convert the full set of CompuSet logic.

## Syntax

**CS2FPW32 /I /C /F /S**

| Parameter | Description |
|---|---|
| I | The name of the INI file from which to load the INI options discussed in the configuration topic following this table. The default is the FSISYS.INI file. |
| C | The name of the configuration file that contains font information. If you omit this parameter, the utility uses the value for the Config option in the CompuSet control group. |
| F | The name of the form you want to convert. |
| L | Including this parameter tells the utility to log all of the debug information created when creating FAP files from a CompuSet file. This parameter is turned off by default. |

| Parameter | Description |
|---|---|
| S | The name of the file that contains the master styles, if any. If you omit this parameter, the utility uses the value for the MasterStyles option in the CompuSet control group. |

All parameters are case insensitive and can be preceded by a slash (/) or a dash (-).

## Configuring the INI File

Since Skywire Software applications can use the same font files as the Document Sciences system, first copy the font files into your Skywire Software Metacode resource directory. This is the same place the Metacode fonts licensed by Skywire Software are located. Then use Font Manager to import the fonts. Specify the name of the new font cross-reference (FXR) file in the XRFFile option of the MasterResource control group. The CSET2FAP utility also uses these INI options:

```
< CompuSet >
    MasterStyles   =
    Config         =
    ExtremeLogging =
< MasterResource >
    DefLib         =
< Config:XXX >
    XRFFile        =
< Loaders >
    Loader         =
< Loader:CompuSet >
    Desc           =
    Func           =
    Module         =
```

| Option | Comments |
|---|---|
| CompuSet control group | |
| MasterStyles | (Optional) Enter the global styles definition file name. |
| Config | Enter the name of the file that contains logical and machine font definitions and the font files in which they are located. |
| ExtremeLogging | Enter Yes to view all of the debug information when creating a FAP file from a Document Sciences file. The system defaults to No. |
| MasterResource control group | |
| Deflib | Enter the path to your Skywire Software resources. |
| Config:XXX control group where XXX is the name of the library or workspace | |
| XRFFile | Enter the name of the font cross-reference file (FXR) you want the system to load. |
| LOGO.DAT | Enter the name of the file containing the fonts you want to use as logos. For more information see Using Fonts as Logos on page 59. |

| Option | Comments |
|---|---|
| Loaders control group | |
| Loader | This entry is used to locate image loader INI groups. For instance, you could enter *CompuSet*. |
| Loader:CompuSet control group | |
| Desc | Enter the description you want to appear in the file types list of the File Open window. For instance, you could enter *Compuset files (*.TXT)*. |
| Func | Enter the name of the image loading function, such as *CompuSetImageLoader*. |
| Module | Enter the name of the image loader module, such as *CSLDRW32*. |

**NOTE:** Because the conversion does not handle all CompuSet commands, you may receive some error messages.

## Using Fonts as Logos

If you have fonts you use as logos, you must first convert them to LOG files. Then, enter the root file name for each logo in the LOGO.DAT file. The LOGO.DAT file should reside in the current working directory. The LOGO.DAT file is a semicolon-delimited text file that names the various rotations of logo fonts and it should look similar to this:

```
[file name for 0° rotation];[file name for 90° rotation];[file name
for 180°rotation];[file name for 270° rotation];
```

Any of the rotations can be left out, but you must include the semicolon as a delimiter.

**NOTE:** The logo fonts should not be present in the FXR since the utility uses not finding a font as a trigger to look for a logo.

# CVTFASR

Use the CVTFASR utility to convert a FORMDEF and SETRCPTB pair into BDF (Business unit definition file), GRP (group file), and FOR (form file) files. The existing FORMDEF and SETRCPTB files are not modified or removed.

Syntax: **CVTFASR /FORMDEF /SETRECIP /INI**

| Parameter | Description |
|---|---|
| FORMDEF | Enter the name of the form definition file. |
| SETRECIP | Enter the name of the set recipient file. |
| INI | Enter the name of the INI file. |

The CVTFASR utility reads the specified FORMDEF and SETRECIP files and generates the appropriate BDF, GRP, and FOR files. If you include the INI parameter, the utility opens and reads that INI file. The utility tries to locate these INI options to get the directory locations into which the BDF, GRP and FOR files will be written.

```
< MasterResource >
    BDFLIB = .\bdflib
    GRPLIB = .\grplib
    FORLIB = .\forlib
```

If the INI option for one or more of the generated file types is missing, the utility writes the files for that type into the working directory.

Here is an example of the messages you may see when running the CVTFASR utility on Windows:

```
c:\fap\mstrres\rpex1>cvtfasrw /formdef=.\deflib\form.dat /
setrecip=.\deflib\setrcptb.dat
--- DocuCorp CVTFASR Utility Program (C) ---
--- Convert Formdef And SetRecip Files ---

Number Of Files Generated From Formdef & SetRecip:
Basedef (BDF) : 1
Group (GRP) : 4
Form (FOR) : 24

CVTFASRW was successful.
```

# DALRUN

Use this utility to execute and debug DAL functions.

## Program names

Windows DALRW32.EXE

## Syntax

**DALRW32 /X /INI /D /T**

| Parameter | Description |
|---|---|
| /X | (Optional) Supplies the name of a script to run. If you omit this option, you can use this INI option to provide the name of the script:<br>`< DALRun >`<br>`Script = file name`<br>You can use any extension. The default is *DAL*. |
| /INI | (Optional) Supplies the name of an INI file to load. This INI file supplies additional parameters and options. If an INI file named DALRUN.INI is present, the utility loads it by default.<br>Here are the INI options you can include in the INI file:<br>`< DALRun >`<br>`Title = title string(an override to the window title)`<br>`Script = file name (the script to run)`<br>`< DALFunctions >`<br>`Keyword = DLLMOD->FunctionName`<br>`Keyword2 = DLLMOD->FunctionName2`<br>`(and so on)` |
| /D | The debug switch starts the DAL debugger. When on, the script executes in single step mode and registers this DAL function: *DEBUG("message")*.<br>The DEBUG function breaks execution, displays a message, and invokes the debugger in single step mode. |
| /T | This parameter sends certain text messages to the standard output device. These messages are not visible at runtime, but may be redirected when you run this utility. Here is an example:<br>`DALRW32 /ini=test /d /t > test.txt` |

**NOTE:** Refer to the DAL Reference for information on the DAL functions you can use to create the scripts DALRUN executes.

Debug messages, certain errors, and a dump of the symbol table at the end of the run are examples of output this utility will generate.

# DB2DB

Use this utility to copy data from one database table to another database table. The database tables can be from a different DBMS, such as one from ODBC and the other from a native DB2 database.

## Program names

Windows DB2DBW32.EXE

## Syntax

**DB2DBW32 /ST /SD /SD /TT /TD /TH /?**

| Parameter | Description |
|---|---|
| ST | Enter the name of the source table. |
| SD | Enter the name of the DFD file for the source table. |
| SH | Enter the type of handler for the source table. |
| TT | Enter the name of the target table. |
| TD | Enter the name of the DFD file for the target table. |
| TH | Enter the type of handler for the target table. |
| ? | Print the parameter usage information. |

**NOTE:** The source and the target table names must be different. You can rename target table after the conversion to a new database. For example:

db2dbw32 /st=xdb ..... /tt= xdb ... (incorrect)
db2dbw32 /st=xdb ..... /tt= xdb2 ... (correct)

Here are two examples:

## Example 1

How to convert an XDB dictionary (a CB5 table) into a DB2 table:

```
db2dbw32 /st=xdb /sb=.\deflib\xdb.dfd /sh=CB5  /tt= xdb2 /td=
.\deflib\xdb.dfd /th=db2
```

These parameters tell the utility to copy the source table (XDB), which is associated with the xBase DBHandler called *CB5* and is mapped using the XDB.DFD file in the .\deflib directory, to the target table (xdb2), which is associated with a DB2 DBHandler called *DB2* and is also mapped using the same DFD file. In this example, the actual name of the XDB2 DB2 table is *CP_Xpress*.

In this example, your FSIUSER.INI file contains entries similar to these. The DB2 table, cpxdb2, is contained in the DB2 database, CP_Xpress.

```
< DBHandler:DB2 >
    Database    = CP_Xpress
    BindFile    = c:\dap32103\dll\db2lib.bnd
    Connect     = Yes
    CreateTable= Yes
    UserID      = db2admin
    Debug       = Yes
```

```
    PassWd      = XXX
< DBTable:xdb >
    DBHandler   = CB5
< DBTable:xdb2 >
    DBHandler   = DB2
< DB2_FileConvert >
    xdb2        = cp_xpress_xdb
```

You can use the DB2_FileConvert INI control group to specify longer names for the table in DB2.

| Option | Description |
|---|---|
| CreateTable | Enter Yes so DB2 will create the table if it does not exist. |
| BindFile | Enter the name and path of the DB2LIB.BND file. |

Example 2

How to convert an Access table into a DB2 table:

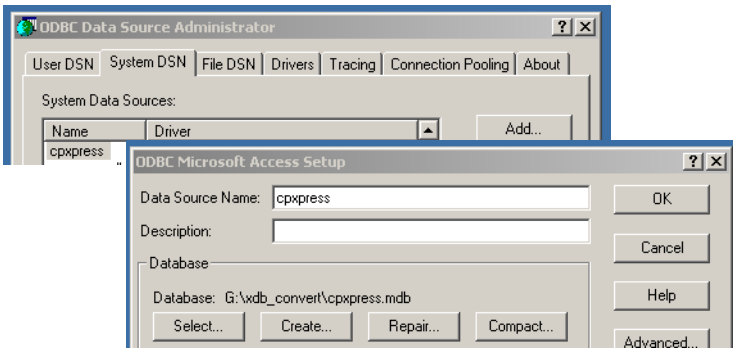
```
db2dbw32 /st=coverage /sd=.\deflib\coverage.dfd /sh=CB5  /tt=
coverage2 /td= .\deflib\coverage2.dfd /th=db2
```

These parameters tell the utility to copy the Microsoft Access source table (coverage), which is associated with a DBHandler called *ODBC* and is mapped using the COVERAGE.DFD file in the .\deflib directory, to the target table (coverage2), which is associated with a DBHandler called *DB2* and is mapped using the COVERAGE2.DFD file in the .\deflib directory.

In this example, your FSIUSER.INI file contains entries similar to these. The DB2 table, coverage2, is contained in the DB2 database, CP_Xpress.

```
< DBHandler:DB2 >
    Database    = CP_Xpress
    BindFile    = c:\dap32103\dll\db2lib.bnd
    Connect     = Yes
    CreateTable= Yes
    UserID      = db2admin
    Debug       = Yes
    PassWd      = MVF
< DBHandler:ODBC <
    Server      = cpxpress
< DBTable:Coverage <
    DBHandler   = ODBC
< DBTable:Coverage2 <
    DBHandler   = DB2
```

These screen samples are an example of the MS Data Sources (ODBC) setup from a conversion:


ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
System Data Sources:
Name
Driver
Add...
cpxpress
ODBC Microsoft Access Setup
Data Source Name:
cpxpress
Description:
Database
Database: G:\xdb_convert\cpxpress.mdb
Select...
Create...
Repair...
Compact...
OK
Cancel
Help
Advanced...

# DCD2FAP

Use this utility to convert a Docucorp Compound Document (DCD) file into a FAP file. DCD files are produced by Skywire Software applications such as Documerge for Windows NT version 3.0 and older versions of Documanage (Printcommander for Windows 95 version 2.0.)

## Program names

| | |
|---|---|
| Windows | DCD2FAPW.EXE |

## Syntax

**DCD2FAPW /I /X**

| Parameter | Description |
|---|---|
| /I | The name of the input DCD file. You can use wildcards. |
| /X | The name of the font cross-reference (FXR) file. Include the extension (.FXR). |

The resulting FAP file will have the same name as the DCD file but with a *FAP* extension.

# DFD2DDL

Use this utility to generate DDL (Data Definition Language) files from your DFD files. DDL files consist of the SQL statements that create your tables and views. This is helpful if you have an RDBMS database where an administrator is required to create tables and indexes and users are not typically granted those privileges.

## Program names

Windows DFD2DDLW.EXE

## Syntax

**DFD2DDLW /I /O /D /T /S /P /L**

| Parameter | Description |
|---|---|
| /I | Enter the name of the DFD file. Include the full or relative path. |
| /O | Enter the name you want the utility to assign to the DDL file it will create. The default is the DDLFile root plus the extension *.SQL*. |
| /D | Enter the name of the target database. You can choose from: MSSQL, DB2, Oracle, MySQL. The default is MSSQL. |
| /T | Enter the name of the target table. The default is the DFD root name. |
| /S | (Optional) Enter the name of the target schema. This is typically used for Oracle schema, DB2 schema, and MSSQL dbowner. |
| /P | (Optional) Enter the name of the primary table index. This allows for unique constraints on key fields. |
| /L | (Optional) Enter the name of the target location. This is used for DB2 (location) and MySQL (database name). |

You should avoid the following ExtTypes because they are not supported as external data types in Documaker's database handlers:

- CHAR
- UCHAR
- DECIMAL
- DOUBLE
- LONG_DOUBLE
- TIMESTAMP
- DATETIME

# FAP2CFA

Use the FAP2CFA utility to convert a FAP file into a compiled FAP file. By pre-compiling your FAP files, you can speed processing.

**NOTE:** The version of the system you use to compile the FAP and FXR files must be the same version you will use when running Documaker Server. Furthermore, the platforms must also match. For instance, if you compile the FAP and FXR files on version 11.0 for Windows, to use them in Documaker Server (GenTrn, GenData, GenPrint), you must run version 11.0 for Windows of Documaker Server.

## Program names

Windows FAP2CFAW.EXE

## Syntax

**FAP2CFAW /I /O /INI /F**

| Parameter | Description |
|---|---|
| /I | The input FAP file or font cross-reference (FXR) file (you can use asterisks as wildcards). |
| /O | The output CFA or CFX file. You can enter an absolute (d:\dap\myinc\mstrres\fap) or relative path (..\mstrres\fap). |
| /INI | Specifies an INI file which contains additional parameters. You can enter an absolute (d:\dap\myinc\mstrres\fap) or relative path (..\mstrres\fap). |
| /F | Include this option to create a compiled font cross-reference (FXR) file which contains only fonts. |

This utility lets you compile FAP files one at a time. To compile all of the FAP files listed in a FORM.DAT file, see FDT2CFA on page 83. To convert a CFA or CFX file back into a FAP or FXR file. see CFA2FAP on page 53.

Make sure the CompiledFAP option is set to *Yes* (the default) in the RunMode control group of your INI files before you run the system using precompiled FAP files. In addition, use the File, Library Setup option (or edit your INI files) to specify the path for CompLib, which is where the system will look for the compiled files. In your INI files, you will find this setting in the following control group:

```
< MasterResource >
    CompLib = (library which contains the CFA and CFX files)
```

For MVS systems, you can specify a DD name, such as DD:COMPLIB()

**NOTE:** If you are using pre-compiled FAP files, you must also use compiled FXR files.

### Example

| If you enter this command... | The result is a compiled FAP file named... |
|---|---|
| FAP2CFAW /I=qaihead | qaihead.cfa |
| FAP2CFAW /I=qaihead / o=qaihead1 | qaihead1.cfa |
| FAP2CFAW /I=cwqa240.fxr /f | cwqa240.cfx |
| FAP2CFAW /I=cwqa240 /f | cwqa240.cfx |
| FAP2CFAW /I=qaihead.old | qaihead.cfa |
| FAP2CFAW /I=..\faps\qaihead | qaihead.cfa, located in the ..\fap subdirectory |

# FAP2DDT

Use the FAP2DDT utility to create or update a DDT file from a FAP file.

**NOTE:** You can also perform this task using the File, Convert, Multiple FAPs option in Docucreate.

## Program names

| | |
|---|---|
| Windows | FAP2DDTW.EXE |

## Syntax

**FAP2DDTW /I /O**

| Parameter | Description |
|---|---|
| /I | Enter the name of the input FAP file. Omit the extension. |
| /O | (Optional) Enter the name of the DDT file. If you omit this file name, the utility uses the FAP file name with the DDT extension. If the DDT file does not exist, the utility creates it. |
| /X | Enables the output of *;X;Y;FontID;* |
| /V | Enter 7 or 8, force output to the new (8) or old (7) format, defaults to the same format as DDT input, or version 8.0 format if you omit this parameter. |
| /S | Synchronizes with the existing DDT file. |

The FAP2DDT utility creates a data definition table (DDT) file from the existing FAP. The DDT file stores image rule assignments. The parameters are optional and case insensitive.

Keep in mind the FAP2DDT utility adds a default SetOrigin rule to the Image Rules section if one does not exist. The SetOrigin rule that is added is shown here:

```
;SetOrigin;Abs+0,Abs+0;
```

For the FAP2DDT utility to add a SetOrigin rule into your DDT file, include this INI option in the FAPCOMP.INI file:

```
< DDTResource >
    AutoIncludeSetOrigin = ;SetOrigin;Abs+0,Abs+0;
```

The DDTResource control group is used for DDT settings, such as WriteCoordinates, MultipleDDTs, and so on.

A SetOrigin rule is only added to a DDT file if there is an AutoIncludeSetOrigin option and these conditions are true:

- The AutoIncludeSetOrigin setting begins with a semicolon (;)
- The AutoIncludeSetOrigin setting contains the string “SetOrigin”
- The DDT file does not already contain a SetOrigin rule (SetOrigin, SetOriginI, SetOriginM, RULSetOrigin2, UTILSetOrigin)

Other than verifying that the rule begins with a semicolon and contains some form of the string “SetOrigin,” no other validation is performed. You must make sure you specify a valid rule with valid parameters.

**NOTE:** Choosing the Convert Multiple FAPs, Update DDT from FAP option in Docucreate is another way to have the default SetOrigin rule added to the DDT file if it does not exist. The same caveats apply.

Example **FAP2DDTW /I=fapfile**

This will create a file named FAPFILE.DDT whose contents look like:

```
<Image Rules>
;SetImageDimensions;0,0,26400,20400,400,600,400,600;

/* By default, this image contains the following fields */
<Image Fields>

<Image Field Rules Override>
;0;0;BLANK FIELD NAME;;;BLANK FIELD NAME;;0;;noopfunc;;;;;;
```

# FAP2FRM

Use the FAP2FRM utility to compile a FAP file into a Xerox FRM Metacode overlay file.

## Program names

| | |
|---|---|
| Windows | FAP2FRMW.EXE |

## Syntax

**FAP2FRMW /I /X**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FAP file. Omit the extension. |
| /X | Enter the name of the font cross-reference (FXR) file. Omit the extension. |
| /P | Enter the INI file PrtType to use, such as *XER*. |
| /D | Enter **SF** for short bind front page duplex. Enter **SB** for short bind back page duplex. |
| /H | (Optional) Add this parameter to use HMI. |
| /C | (Optional) Add this parameter to use color in the FRM file. |

**NOTE:** The name of the FAP file cannot exceed six characters (123456.fap).

The FAP2FRM utility requires these files:

- FSISYS.INI
- Font cross-reference file, such as REL103.FXR

FAP2FRM generates a Xerox form printer resource file (FRM) from a FAP file. In addition to the FAP file name, the FXR font cross-reference file used by the FAP must also be specified. Since FRM file is a printer resident resource, the file name should be no more than six characters long.

FAP2FRM looks for a control group named Printer in the FSISYS.INI. In the Printer control group, FAP2FRM looks for the PrtType option, which determines the type of printer being used, such as AFP, XER, or PCL. Use the /P parameter to specify which PrtType control group the utility should use. The default control group is PrtType:XER.

If the FAP file contains multiple pages, FAP2FRM generates multiple output files with 2-digit numeric suffixes. In this case, file names for these FAP files should not exceed four characters (1234.FAP).

After running FAP2FRM, download the output FRM files to the printer, using the XERDNLD utility. For more information, see XERDNLD on page 214.

Also, in the Form Set Manager, check the Printer Resident field for each such image.

If the FAP file contains logos, the FAP2FRM utility produces a Xerox FRM file using either Xerox fonts or Xerox images, based on the value for the ImageOpt option in your Xerox printer control group. If you set the ImageOpt option to Yes, Xerox images (IMG files) are used in the Xerox FRM file the utility produces. Otherwise, Xerox fonts (.FNT files) are used.

**NOTE:** Image references are supported in FRM files. If you set the ImageOpt entry to Yes, the system generates the required GHO headers and packets in the FRM file so FAP files that contain IMG references can be converted into Xerox FRMs.

### Example

Follow these steps to convert a precompiled MET file into a printer-resident FRM file that will be used in the form set. These instructions assume you are converting an existing MRL that used multiple floating page segments (METs) on some pages:

1. For performance reasons, when faced with a choice of multiple floating page segments on a page, choose the largest one (the one with the most data on it (characters, lines, or shading) to be the FRM so that image can be resident on the printer and does not have to be in the printed output.
2. Change all references to that image in the FORM.DAT file, making that image printer-resident. You can do this using the Form Set Manager.
3. In the Image Editor, edit the FAP file for that image. If it was used as a floating image (precompiled MET), then it is probably not positioned correctly on the page to be a FRM, because FRMs are static and should be thought of as whole-page images, not floaters. Position the image to where it should be on the page, then use the Format, Page Properties option to select *Letter* as the paper type.
4. Next, select Format, Image Properties, and click the Load DDT button. On the Image Rules tab, highlight *SetOrigin*, and look at the Data field. Because the image was probably a floater, it should have relative positioning, such as *REL+0,REL+0*. Change it an absolute value, such as *ABS+0,ABS+0*. Save the image and DDT information. You do not have to change any FAP or DDT files for images that are positioned before where the FRM image is positioned on the page.
5. Next, load the next image that appears directly after the FRM image on that page in the Image Editor. Change its DDT information similarly, except position it below the FRM image, with coordinates larger than zero.

   You can get good coordinates from the standard NAFILE that was used before making these changes. If there are any images that occur after this one on the same page, you should not have to change them, because they should be already set to relative positioning. If the system did not force the image immediately after the FRM to be absolutely positioned on the same page, triggering would tell the GenData program to insert a page break at the end of the FRM, and the next image will incorrectly be printed at the top of the next page. After you do this, you can run the GenData program and make sure the images are positioned correctly.
6. Create the FRM using the FAP2FRM utility, then download the FRM to the printer. You can sample it on the printer to make sure it prints and has the correct positioning on the page. Keep all FRM files on the workstation also in FormLib when the GenPrint program runs.
7. Finally, run the GenPrint program and print.

### Limitations

You cannot have FRM and MET files in the same MRL which have the same file name. For instance, in RPEX1's FORM.DAT, page one has image Q1SNAM listed as printer-resident (an FRM). Page two has the same file name Q1SNAM, only for this page it is a pre-compiled MET. Doing so will cause errors.

# FAP2MET

Use this utility to compile a FAP file into a pre-complied Xerox Metacode overlay. This utility generates a Metacode print file from a FAP file.

## Program names

| | |
|---|---|
| Windows | FAP2METW.EXE |
| MVS | See the Documaker Server Installation Guide |

## Syntax

**FAP2METW /I /X /P /N /VF /SV /D /H /C /LB**

| Parameter | Description |
|---|---|
| /I | Enter then name of the FAP file (you can use asterisks as wildcards). |
| /X | Enter the name of the font cross-reference (FXR) file. |
| /P | Enter the INI file PrtType to use, such as *XER*. |
| /N | Enter the font ID to use to print name of image, such as 11006. |
| /VF | Add this parameter to sample print with variable fields. |
| /SV | Add this parameter to save the MET file for the GenPrint program. Keep in mind you cannot use this parameter if you have the ImageOpt INI option set to Yes for floating images which contain inline graphic logos. |
| /D | (Optional) Enter **SF** for short bind front page duplex, enter **SB** for short bind back page duplex. |
| /H | Add this parameter to use HMI. |
| /C | Add this parameter to use color in the MET file. |
| /LB | Add this parameter to override the default logical bottom for the form being normalized. To specify a different logical bottom value, include the parameter and a value, as shown here:<br>`/LB = value` |

This utility requires these files:

- FSISYS.INI
- Font cross-reference file such as REL103.FXR

Depending on the option flags, the print file may be a print-ready file or a pre-complied file for use with the GenPrint program.

In addition to the FAP file name, you must also specify the name of the font cross-reference file (FXR) used by the FAP file. To print the file, you must install Xerox versions of the fonts used in the FAP file on the printer.

**NOTE:** The INI options for FAP2MET are in XEROXJDL control group in the FSISYS.INI file in version 8.0, and in PrtType:??? control group in versions 8.5 and later.

This table shows the INI options which will work in the FSI environment:

| Option | Setting |
|---|---|
| DJDEIden | A'@@@DJDE' |
| DJDEOffset | 0 |
| DJDESkip | 8 |
| JDLHost | IBMONL |
| JDLData | 0,255 |
| JDLCode | NONE |
| JDLName | DFAULT |
| JDEName | DFLT |
| JDLRStack | 0,10,X'1313131313131313313' |
| JDLROffset | 0,9,X'121212121212121212' |
| JDLRPage | 1,5,X'FFFF26FFFF' |
| ImageOpt | No |
| PrinterInk | Blue |
| Environment | Win |
| OutMode | BARR |

**NOTE:** These settings must match those set in the JDL. The JDL is a compiled JSL installed on the Xerox printer as a resource. The JDL contains the print instructions for Xerox print jobs.

## Running on MVS

If you are running the utility on MVS, set the Environment and OutMode options to:

```
Environment= MVS
OutMode= JES2
```

This is based on definitions in the FSISYS.INI file. For example, in the INI file, you can set up different modes of Xerox printing by setting up different control groups, such as...

```
< PrtType:XER1 >
   DJDEIDEN = E'$$Xerox'
   OUTMODE = JES2
.........
< PrtType:XER2 >
   DJDEIDEN = A'@@@DJDE'
   OUTMODE = BARR
.........
```

When you want to send the FAP2MET output to the printer via JES2, you could provide /P= XER1, or if your output is going through BARR, you could provide /P = XER2 to select proper paragraph. If you omit the /P parameter, it defaults to XER.

The */D* parameter provides short edge binding duplex functions. */D=SB* means the image is to be compiled for short edge binding as a back page. This forces the text to be printed in inverse portrait mode. Your FXR file should have the corresponding font names suitable for inverse portrait mode. */D=SF* means the image is to be compiled for short edge binding as a front page. The default is *SF*.

The */VF* parameter creates Metacode which can be sent to the printer immediately for sample printing, with all field positions indicated with XXXX strings. The name of the form is printed at the bottom right corner, using the font ID you specified with the */N* parameter. Check your FXR file, and make sure the font ID is assigned to a suitable font.

**NOTE:** Do not use this parameter if you are precompiling Metacode files in preparation to run the GenPrint program.

The */SV* parameter creates an intermediate form of Metacode which can not be sent to the printer directly, but which at GenPrint time, gets merged with field data and gets converted into proper Metacode. This combined Metacode is then sent to the printer. Therefore, if you are creating the Metacode files as precompiled Metacode files to be used in GenPrint process, use */SV* parameter. In the absence of either of these flags, a sample print Metacode is generated without the XXXX strings.

If you add an *H* to either the */VF* or */SV* parameters (*/VFH* or */SVH*), the system generates a more efficient Metacode output by combining several Metacode records into one, reducing the output file size. You can also specify this option as a separate standalone parameter (*/H*).

**NOTE:** This feature is available only for portrait mode printing.

Use the */C* parameter when you want to print the compiled instream Metacode on a 4850 or 4890 highlight color printer, and some elements of the form have a color specification. (Be sure to set PrinterInk option to whatever ink is installed on the printer. Also, the SendColor option must be set to *Yes*).

**NOTE:** The PrinterInk option supports red, blue, green, ruby, violet, brown, gray, cardinal, royal, cyan, and magenta.

### Parameters passed via the PARM=‘ ‘field (MVS/JCL)

In earlier versions (before 8.5), these parameters were passed in the PARM=‘ ‘field: fapfilename, *SAVE* or *TEMPLATE*, and *USEHMI*.

The parameters are identical in both environments. The only difference is that an extra backslash (/) must precede the list of parameters. A typical EXEC statement in a FAP2MET run JCL would be:

```
//FAP2MET    EXEC    PGM=FAP2MET,PARM='/  /I=fapfile /P=XER /VFH  /C'
```

## Producing normalized Metacode

You can use the FAP2MET utility to produce normalized Metacode files. To produce a normalized Metacode file, you must add these INI settings to your FSISYS.INI file:

```
< Control >
    Normalize = Norm
< PrtType:Norm >
    ...
```

| Option | Description |
|---|---|
| Normalize | Specify the PrtType control group you want the system to use when it normalizes the files. For instance, if you create a control group called PrtType:Norm to contain the option you want the system to use when it normalizes Metacode files, enter *Norm*.<br>If the control group you specify is missing or does not appear to be a Xerox printer group (Class=XER) you will receive errors. |

**NOTE:** Typically, the Class option is not explicitly set in the INI file. The Class name is normally derived from the first three letters of the Module option. If the Module is not XERW32, and you want the system to think the control group is a Xerox print group, you must set the Class option to XER.

You can also use the /LB parameter to override the default logical bottom for the form being normalized. To specify a different logical bottom value, include the parameter and a value, as shown here:

```
/LB = value
```

**NOTE:** You only use the /LB parameter when normalizing files.

## GenPrint program notes

The INI options required for the GenPrint program are the same as those needed by the FAP2MET utility. In addition, there are other parameters which affect printing on Xerox printers. In the Print control group, or in the PrtType:XER control group,

```
CompileInstream = Yes or No
```

and in the RunMode control group,

```
DownLoadFAP = Yes or No
```

The CompileInStream=Yes and DownLoadFAP=Yes options load all of the FAP files in memory when the GenPrint program runs. This is slow, and is typically used only during testing and development. This mode does not need precompilation of FAP files into intermediate Metacode files.

For production environments, use the CompileInStream=No and DownLoadFAP=No options. This mode requires that you precompile the FAP files using the FAP2MET utility with the /SV parameter.

The precompiled Metacode files must be available to the GenPrint program in the MSTRRES\FORMS directory in Windows, or in a PDS which is assigned to the PMETLIB DD statement on MVS.

## Multi-page FAP files

The FAP2MET utility supports multiple page FAPs. For example, if TEST.FAP contains 10 pages, FAP2MET will build 10 pre-compiled met files. These files will be used by the GenPrint program as each page is printed.

When you run the FAP2MET utility on a multi-page FAP file, the utility creates a FAP file for each page. In the DDT file, you should then make an entry for the FAP file, and add an eject page for each additional page that makes up the form. The system knows by the eject pages to look for additional FAPs for this form.

You can also use the FAP2MET utility to produce a test print version of a FAP file, otherwise known as a print-ready file. This file is only to be sent to the printer. It is not to be used with the GenPrint program. This type of file is produced when you omit the /SV parameter.

# FAP2OVL

Use the FAP2OVL utility to compile a FAP file into an overlay for an AFP printer.

**NOTE:** You can also create overlays using the Image Editor's Tools, Compile option. See the Docucreate User Guide for more information. You can also use the OVLCOMP utility. See OVLCOMP on page 182 for more information.

## Program names

| | |
|---|---|
| Windows | FAP2OVLW.EXE |
| MVS | See the Documaker Server Installation Guide |

## Syntax

**FAP2OVLW /I /X /VF /O /C /R /INI /NO**

| Parameter | Description |
|---|---|
| /I | The FAP file you want to compile. Omit the extension. You can use asterisks as wildcards to select multiple files. |
| /X | The font cross-reference (FXR) file. Omit the extension. You can enter an absolute (d:\dap\myinc\mstrres\rel110) or relative path (..\mstrres\rel110). |
| /VF | Add this parameter to print variable fields. |
| /O | Specifies the output directory. You can enter an absolute d:\dap\my\mst\ovllib) or relative path (..\mstrres\ovl). |
| /C | Add this parameter if the AFP overlay should use color. Enter one of the named AFP colors such as blue, red, magenta, green, cyan, yellow, dark_blue, orange, purple, dark_green, dark_cyan, mustard, gray, or brown. |
| /R | Enter 240, 300, or 600 to indicate the resolution in dots per inch (DPI). The default is 240.<br>Use this parameter to support producing AFP overlays using 240, 300, or 600 DPI for coordinates and bitmap data produced when using inline logos. Image Editor and Documaker Studio can produce AFP output (print streams and normalized AFP files) using either a 240 or 300 DPI coordinate system. |
| /INI | Specifies an INI file which contains additional parameters. You can enter an absolute (d:\dap\myinc\mstrres\fsisys.ini) or relative path (..\mstrres\fsisys). The file name does not have to be *FSISYS.INI*. |
| /NO | Add this parameter if you want the FAP file name at the bottom but no fields. |

**NOTE:** This utility is not case sensitive. You can use dashes (-) instead of slashes (/) to separate parameters.

The FAP2OVL utility generates an AFP OVL overlay printer resource file from a FAP file. In addition to the FAP file name, you must also specify the font cross-reference (FXR) file used by the FAP file.

The /VF parameter prints the FAP file name at the bottom of the page using font ID 3001, and fills all variable fields on the FAP with *X* characters. The /NO option only prints the FAP file name at the bottom of the page using font ID 3001. These options are useful when you are developing and proofing your implementation.

**NOTE:** Do not use the /VF or /NO options when you are creating an overlay (OVL) file for production print use.

After a production quality overlay file is generated, you have to make it available to the PSF2 by copying it to the print server and add it to the PSF2/ Librarian database.

On MVS, the overlay has to be copied into a PDS, which is attached to the OVERLIB DD statement in the PSF PROC, or a PDS which is assigned to the USERLIB= parameter, on the OUTPUT statement in the job JCL. After all the required overlays are installed on PSF2 or PSF, run the GenPrint program with this INI option:

```
SendOverlays = Yes
```

The spool file the GenPrint program creates will include calls to merge those overlays as needed.

## Printing in color

Include the /C parameter if the AFP overlay should print in color. AFP highlight color printing on printers from Xerox and Oce is supported. Before using this feature, make sure the:

- SendColor INI option is set to Yes.
- Objects you want to print in color (text, lines, shades, and so on) are set to print in color. The Print in Color option is on the Color Selection window in the Image Editor. You can display this window by clicking the Color button on the object's Properties window.

The system maps the RGB (red,green,blue) color setting for each object to the closest AFP named color. The default AFP named colors are blue, red, magenta, green, cyan, yellow, dark_blue, orange, purple, dark_green, dark_cyan, mustard, gray, and brown.

Use the NamedColors option to tell the system to use only specific AFP named colors:

```
< PrtType:AFP >
    NamedColors = blue
```

For example, if you wanted all highlight (non-black) colors mapped to blue, you would set the NamedColors option to blue, as shown above.

To allow the mapping of the colors you assigned to the objects in the FAP file to multiple colors, separate each color with a semicolon (;). For example, to use all of the default AFP named colors except brown, set the NamedColors option as shown here:

```
NamedColors = red;blue;magenta;green;cyan;yellow
```

The order of the named colors does not matter.

# FAP2PDF

Use the FAP2PDF utility to convert FAP files into PDF files. To produce a PDF file from a FAP file, you must also supply an FXR file. The FXR file determines what fonts are used in the PDF files the utility produces.

The utility looks in the PDF printer control group in your FSISYS.INI file for options that tell it how to create the PDF file. These options let you include bookmarks, embed fonts, use internal fonts, and so on.

**NOTE:** You can use the /INI parameter to specify the INI file that contains the PDF printer control group and you can use the /P parameter to specify the actual name of the PDF printer control group. For information about the INI options you can use to customize PDF output, see Using the PDF Print Driver.

## Syntax

**FAP2PDF /I /X /INI /P /VF**

| Parameter | Description |
|---|---|
| /I | Enter the names of the FAP files you want to convert into PDF files. You can use wildcard characters (*). You do not have to specify the .FAP extension. |
| /X | Enter the name of the FXR file you want to use. The FXR file determines which fonts are used in the PDF file. You do not have to specify the .FXR extension. |
| /INI | (optional) Enter the name of the INI file that contains PDF settings. The default is FSISYS.INI. |
| /P | (optional) Enter the name of the PDF PrtType control group in the FSISYS.INI file. The default is PDF. |
| /VF | (optional) Include this parameter if you want to template fields. |

If the FXR and INI files specify that you want the fonts embedded, the INI file must contain a FontLib option in the MasterResource control group that specifies the directory of the TrueType or PostScript fonts.

**NOTE:** To see a list of the FAP2PDF parameters, run the utility with no parameters.

# FAP2RTF

Use the FAP2RTF utility to convert a FAP file into an RTF file that can be used, for instance in a word processor such as Microsoft Word.

## Syntax

**fap2rtf /i /x /ini /o**

| Parameter | Description |
|---|---|
| /i | Enter the name of the FAP file you want to convert. |
| /x | Enter the name of the corresponding FXR file. |
| /ini | Enter the name of the INI file which contains the options for this utility. |
| /o | (Optional) Enter the name for the output RTF file. If you omit this option, the system uses the name of the FAP file and adds the RTF extension. |

Here is an example:

```
Fap2rtf /i=q1addr.fap /x=rel103sm.fxr /ini=fapcomp.ini /o=test.rtf
```

This example shows how you can use wild cards:

```
Fap2rtf /i=*.fap /x=rel103sm.fxr /ini=fapcomp.ini
```

Wild cards are not supported if you also use the /o parameter.

## INI File Settings

The following control groups and options are required for the FAP2RTF utility.

```
< Printer >
    PrtType     = RTF
```

| Option | Description |
|---|---|
| PrtType | Enter a name such as RTF. |

```
< Printers >
    PrtType     = RTF
```

| Option | Description |
|---|---|
| PrtType | Enter the name you entered for this option in the Printer control group. |

```
< PrtType:RTF >
    Module          = RTFW32
    PrintFunc       = RTFPrint
    Device          = c:\Frame.RTF
    PageNumbers     = Yes
    AllowInput      = Yes
    EmptyHeaders    = Yes
    EmptyFooters    = Yes
    WriteFrames     = No
    BmSub           = Yes
    BmSubChar       = _
    TemplateFields  = Yes,Enabled
    SendColor       = Yes,Enabled
```

| Option | Description: |
|---|---|
| Module | The DLL name of the RTF driver. |
| PrintFunc | Enter the function name of the print function. |
| Device | Enter the name of output file when printing from Image Editor. |
| PageNumbers | Enter Yes to print page numbers on each page. The default is No. |
| AllowInput | Enter Yes to enable form fields. The default is No. |
| EmptyHeaders | Enter Yes if you want empty headers in the RTF file. The default is No. |
| EmptyFooters | Enter Yes if you want empty footers in the RTF file. The default is No. |
| WriteFrame | Enter No if frames are not required. The default is Yes. |
| BmSub | Enter No if you *do not* want to replace invalid characters. The default is Yes. |
| BmSubChar | Enter character to use when invalid characters are substituted. The default is the underscore ( _ ). |
| TemplateFields | The default, Yes, tells the system to test print Xs in variable fields. If you also include Enabled (Yes,Enabled), the Template Variable Fields field in Image Editor is checked. |
| SendColor | Enter Yes to print in color. The default is No. If you also include Enabled (Yes,Enabled), the Send Color field in Image Editor is checked. |

If you have chosen to allow form fields, you may also need to include the WordTimeFormats and WordDateFormats control groups. You can use these control groups in case you are using a time or date format in Image Editor that has no equivalent in Word. The following groups and options let you map a Skywire Software format to a Word format.

```
< WordTimeFormats >
    hh:mm XM =
< DateTimeFormats >
    bD/bM/YY =
```

To the left of the equals sign, you list the Skywire Software format used on the image. To the right, you list the Word format you want to use.

# FDT2CFA

Use the FDT2CFA utility to convert all of the FAP files listed in a FORM.DAT file into compiled FAP files. By pre-compiling your FAP files, you can speed processing.

**NOTE:** The version of the system you use to compile the FAP and FXR files must be the same version you will use when running Documaker Server. Furthermore, the platforms must also match. For instance, if you compile the FAP and FXR files on version 11.0 for Windows, to use them in Documaker Server (GenTrn, GenData, GenPrint), you must run version 11.0 for Windows of Documaker Server.

## Program names

| | |
|---|---|
| Windows | FDT2CFAW.EXE |

## Syntax

**FDT2CFAW /INI /I /F /L /O**

| Parameter | Description |
|---|---|
| /INI | The input FSIUSER.INI file. |
| /I | The input FORM.DAT file. |
| /F | The input font cross-reference (FXR) file. |
| /L | The input FAP file library. |
| /O | The output CFA file library. |

This utility lets you compile all of the FAP files in a FORM.DAT file at a time. To compile individual FAP files, see FAP2CFA on page 67. To convert a CFA or CFX file back into a FAP or FXR file. see CFA2FAP on page 53.

Make sure the CompiledFAP option is set to *Yes* (the default) in the RunMode control group of your INI files before you run the system using precompiled FAP files.

In addition, use the File, Library Setup option (or edit your INI files) to specify the path for the CompLib, which is where the system will look for the compiled files. In your INI files, you will find this setting in the following control group:

```
< MasterResource >
    CompLib = (directory the CFA and CFX files are stored in)
```

For MVS systems, you can specify a DD name, such as DD:COMPLIB()

**NOTE:** If you are going to use pre-compiled FAP files, you must also use compiled FXR files.

# FDT2DB

Use this utility to create a cross-referenced database of your master resource library (MRL) forms, images, and fields. You can use the resulting database to query this information from within any appropriate database tool. For instance, you can use this information to:

- Build basic reports, such as a field usage report
- Design import/export files for interfacing to other systems
- Create files for use with Transall

The utility uses the FORM.DAT or BDF (Business Definition File) file to extract resource information into a database file, with tables for the library, form group keys, global recipients, form groups, forms, images, linked recipients, and fields.

**NOTE:** Rules are not converted.

The utility reads the entire MRL, but if the resource is not listed in the FORM.DAT or BDF file, the utility does not include it.

## Program names

| | |
|---|---|
| Windows | FDT2DB.EXE |

## Syntax

**FDT2DB /? /INI /I /BDF /Purge**

| Parameter | Description |
|---|---|
| /? | (Optional) Prints to the console all command line options and defaults. No processing occurs when you include this parameter. |
| /INI | (Optional) Tells the utility which INI file to use. The default is the FSIUSER.INI file. The information in the INI file tells the utility where to find the master resource library (MRL). |
| /I | (Optional) The name of the input file. If you omit this parameter, the utility defaults to FORM.DAT.<br>If you enter a file name but omit the extension, the utility defaults to *DAT* — unless you include the /BDF parameter, in which case it defaults to *BDF*.<br>If you omit the extension, include the period at the end of the file name. |
| /BDF | (Optional) Tells the utility to use the Business Definition (BDF) file instead of the FORM.DAT file. You must include this parameter if you want the utility to use a BDF file when it converts resources.<br>Note: If you plan to use the output in the PPS Reporting Tool, do not include this parameter. |
| /Purge | (Optional) Tells the utility to remove the current table data before it repopulates the database with data from the BDF or FORM.DAT file. |

You can run this utility multiple times. Be aware, however, that if you run it multiple times, data may be duplicated and you may have unnecessary records in the database. To remove unnecessary records, include the /Purge parameter. This tells the utility to remove all records from the database before opening the input file. After the purge, only records from the input file are in the database file.

### Specifying the database

Use the following INI options to specify the database you want to use:

```
< DBHandlers >
    DBHandler = ODBC
```

You cannot run this utility without the DBHandler option. The DBHandlers control group defines the ODBC available to the system. With this option, you can name the ODBC anything you want.

```
< DBHandler:ODBC >
    Server = RPEX1
```

The Server option is referenced by the DBHandler option. The utility searches for the DBHandlers:XXXX (XXXX being the selected ODBC) for Server option. If you omit these options and choose to use a blank database, the utility uses the default table DFDs.

## Creating a Database

You can create an ODBC database and use Windows' Control Panel to open the ODBC applet and add a new database connection by following these steps:

**NOTE:** To use the new database, remember to set up the Server option in the INI file.

1. Open Windows' Control Panel. Double-click the ODBC Data Sources icon.
2. On the User DSN page (the first page), click Add. The next page that appears should have the Microsoft Access Driver selected. If not, select it and click Finish.
3. On the next window, enter a name for your database in the Data Source Name field. Then click Create.
4. On the File window, move to the directory where you want the database created. Enter the database file name. This does not have to be the same name as the Data Source Name. When you finish, click Ok. You should now see the database name above the buttons in the Data Source Name window. Click OK again.

   Now you see the list of databases window again and your new database should appear in the list. Click Ok to close the window.

**NOTE:** For Windows 2000, first select Administrative Tools which takes you to another window that contains the ODBC Data Sources icon.

5. Edit the FSIUSER.INI for the new database. Add these lines (assuming its name is *RPEX1*):

   ```
   < DBHandler:ODBC >
       Server = RPEX1
   ```

   The Server option defines the Data Source Name you used to name the database, not the file name of the database.
6. Run the utility in the directory where your INI files reside. When it finishes, you should be able to open the database using Access to see that it is populated.

## Using an Existing Database

If you use an existing database, make sure the tables are in the following formats:

### Library table

| Key | Type | Description |
|---|---|---|
| LibraryName | Text (32) | Short name to identify a library set. This relates a name to a path where the rest of the tables reside. |
| LibraryDescription | Text (50) | |
| LibraryPath | Text (255) | MRL library path |

### FormGroupKeys table

| Key | Type | Description |
|---|---|---|
| KeyType | Text (3) | For example, 001 = key 1 (such as Company), 002 = key 2 (such as Line of business). |
| KeyName | Text (30) | Key name used in the FormGroups table. |

### GlobalRecipients table

| Key | Type | Description |
|---|---|---|
| RecipName | Text (30) | Short recipient name. |
| RecipDescription | Text (50) | Longer description. |
| RecipCode | Text (20) | Used in some print sorting. |
| Recip_ID | Number | Unique ID. Used to link recipients. |

### FormGroups table

| Key | Type | Description |
|---|---|---|
| GroupName1 | Text (30) | Such as a company. Must be an entry in the FormGroupKeys table where KeyType = 001. |
| GroupName2 | Text (30) | Such as a line of business. Must be an entry in the FormGroupKeys table where KeyType = 002. |
| GroupName3 | Text (30) | Such as a recipient. Not yet supported. When supported, this value must be a name in the GlobalRecipients table. |
| Group_ID | Number | Unique ID. Used to group forms within the Forms table |

### Forms table

| Key | Type | Description |
|---|---|---|
| Group_ID | Number | Key from the FormGroups table |
| FormName | Text (30) | |
| FormEffectiveDate | Date/time | |
| FormDescription | Text (50) | |
| FormOptions | Text (20) | |
| FormRevInfo | Text (20) | Reserved for future use. |
| Form_ID | Number | Unique ID. Used to group images in the Images table. |

### Images table

| Key | Type | Description |
|---|---|---|
| Form_ID | Number | Key from the Forms table. |
| FormSequence | Number | The order within the form. |
| ImageName | Text (30) | The image name. |
| ImageEffectiveDate | Date/time | |
| ImageDescription | Text (50) | The image description. |
| ImageOptions | Text (20) | |
| ImageRevInfo | Text (20) | Reserved for future use. |
| Image_ID | Number | Unique ID. Used to group fields in the Fields table. |

### LinkedRecipients table

| Key | Type | Description |
|---|---|---|
| RecipScope | Number | Contains 0=image, 1=form, 2=group |
| Referring_ID | Number | Form_ID, or Image_ID, or Group_ID |
| Recip_ID | Number | The key from the GlobalRecipients table. |
| RecipCopyCount | Number | |

### Fields table

| Key | Type | Description |
|---|---|---|
| FieldScope | Number | 0=image, 1=form, 3=form set |
| Referring_ID | Number | 0=global, else, Form_ID or Image_ID |
| FieldName | Text (32) | |
| FieldType | Text (20) | |
| FieldFormat | Text (20) | |
| FieldLength | Number | |

### INI options

Also, if you use an existing database and the tables are already created, make sure you have these INI options set up:

```
< ODBC_FileConvert >
    FrmGrpKy    = FormGroupKeys
    GlbRec      = GlobalRecipients
    FrmGrps     = FormGroups
    LnkRcps     = LinkedRecipients
```

These are necessary because some types of databases do not like tables with long names.

# FDT2EDL

Use this utility to generate EDL output from a FORM.DAT file. You can use this utility to read a FORM.DAT file and convert an master resource library (MRL) into a Documerge EDL file.

**NOTE:** An EDL file serves the same function in the Documerge system as an FDT file does in the DAP system.

## Program names

Windows FDT2EDL.EXE

## Syntax

**FDT2EDL /INI /O /N**

| Parameter | Description |
|---|---|
| /INI | Enter the path and file name of FSIUSER.INI file. |
| /O | Enter the drive and path of output directory. |
| /N | Enter the name of output file you want to create. |

For each line in the FORM.DAT file, the images are parsed and a group of entries are written out for each image.

```
Output [*<edlname>*] required section header-<edlname> supplied by
command line /N parameter
[ChnDir=<edlpath>\] directory location of chain files, with trailing
backslash
```

The following entries are written for each image:

```
[<membername>]
  1+Info=ED=YYYYMMDD,MD=YYYYMMDD,DTN=0,Desc=36 spaces
   HiRev=n
  1+CHNS=
```

Where...

| Component | Description |
|---|---|
| [*<edlname>*] | the output EDL name supplied via command line parameter |
| [ChnDir=<edlpath>] | the output directory supplied via command line parameter. |
| [<membername>] | the name of the image file (FAP) for the form in the FORM.DAT file |
| ED | YYYYMMDD is the current date |
| MD | YYYYMMDD is the current date |
| DTN | is always zero |

| Component | Description |
|---|---|
| Desc | 36 spaces |
| HiRev | is always 1 |
| 1+CHNS | entry with no value |

The utility returns success or failure. The following errors may occur:

- Output path was not specified
- Output file name was not specified
- Could not open the FSIUSER.INI file
- Could not open the FORM.DAT file
- Could not open the output file

# FIXFNT

Use this utility to test and fix Xerox fonts so you can use them with optimized Metacode. If you notice that text prints in the wrong location on the page when a Metacode print stream is optimized but prints correctly when not optimized, you may be using a font that needs to be corrected.

See METOPT on page 159 for more information about optimizing Metacode print streams.

**NOTE:** Docuview, which is part of Docusave Workstation, requires Xerox resources to be padded to fill 512 byte blocks. This includes Xerox fonts. Some old Xerox fonts built with prior versions of the system do not meet this criteria. You can use this utility to read and update those Xerox fonts so they can be used by Docuview.

## Program names

| | |
|---|---|
| Windows | FIXFNTW.EXE |

## Syntax

**FIXFNT /I /T**

| Parameter | Description |
|---|---|
| /I | The name of the FNT file. You can use asterisks (*) as wildcards. |
| /T | Add this parameter if you only want to test the Xerox font. |

Use the /T option to test any Xerox font to see if it can be used with Metacode optimization.

Once this utility finishes, download the corrected fonts to the printer to eliminate the problem.

## Understanding the System

Before version 10.0, logos converted to Xerox fonts contained font specification table entries that prevent them from being used with Metacode optimization. In most cases, this is not an issue if logos are placed on forms. To cause a problem with Metacode optimization, you must use a text label instead of logo for printing a signature or graphic logo and the Xerox signature font must be inserted into the FXR file.

# FIXFORM

Use the FIXFORM utility to make changes and correct problems in FAP and DDT files. This utility is often used when converting older FAP files into a newer format. You can also use it, for instance, to check DAL script syntax and perform other tasks such as reassigning font IDs, and modifying text rectangle coordinates which define the space reserved for text.

**NOTE:** You can also use Docucreate to perform many of these same tasks. Use the File, Convert, Multiple FAPs option on the main menu.

## Program names

| | |
|---|---|
| Windows | FIXFMW32.EXE |

## Syntax

**FIXFMW32 /I /X /O /SHRINK /SS /SX /G/P**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FAP file. |
| /X | Enter the name of the font cross-reference (FXR) file. |
| /O | Enter the name of the output FAP file if different. Omit the extension. |
| /SHRINK | Include this parameter to shrink the FAP file to a custom size. This parameter tells the utility to remove all of the white space from the bottom, right-hand side of the image.<br>This has the same affect as using the Auto Size button on the Image Editor's Page Properties window. |
| /SS | Include this parameter to check field DAL script syntax. |
| /SX | Include this parameter to check field DAL script syntax and attempt to correct errors. |
| ?G | Include this parameter to load global bitmaps (logos). |
| /P | Enter the name of the INI file. |

You can also include these conversion options:

| Option | Description |
|---|---|
| /A | Convert all colored objects to print in color |
| /C | Combine text elements. The utility defaults to combining text elements if the space between those elements is less than 99% of the width of the space character. You can also specify the percentage. For instance, */C50* tells the utility to combine the text elements into a single text element if the distance between them is less than 50% of the width of the space character. |
| /M | Map alternate fonts |
| /K | Map alternate characters |

| Option | Description |
|---|---|
| /N | Fix negative coordinates |
| /B | Widen text areas as needed to prevent word wrapping that could occur when performing tasks such as re-mapping fonts. Use /B to check the form to see if the text in the text area exceeds the right boundary of the text area. If it does, the FIXFORM utility adjusts the right edge of the text area to be one FAP unit greater than the right edge of the text — as long as that adjustment does not extend the text area beyond the edge of the page. This keeps existing text areas from wrapping differently. This parameter does not tell the utility to contract the right edge, but only to expand it if necessary. |
| /R | Fix rectangle coordinates using font information. Use /RT to preserve top coordinates. Use /RB to preserve bottom coordinates. You would typically use /RB. |
| /E | Examine coordinates for overlap. |
| /L | Adjust x coordinates for 1/4 in. left margin. |
| /T | Adjust y coordinates for 1/6 in. top margin. |
| /S | Convert DAL calcs to DAL script statements. |
| /U | Make field names unique. |
| /Y4 | Force all date field formats to use a four-digit year. |
| /D | Delete all fields. |
| /ROTATE90 | Rotate 90 degrees to left. |
| /NOFIXDIMS | Do not fix dimensions (for custom size images). |

**NOTE:** If you omit an option, the system omits that task, unless the option is turned on in the INI file. The settings in the INI file override command line parameters.

## Setting up the INI file

You can enter some parameters from the command line, while others must be set in an INI file. The /P parameter specifies the name of the INI file, usually FIXFORM.INI.

To use this utility you must have a FIXFORM.INI and an FXR file. The FXR file must be specified in the parameter list to change font information. When you use the File, Convert, Multiple FAPs option, instead of this utility, be aware that any FIXFORM headings in the FAPCOMP.INI file will be used instead of those in the FIXFORM.INI file. Therefore, you should set up the INI settings in only one INI file.

Here is an example of a FIXFORM.INI file. This example shows all of all the options for the FixOptions control group.

```
< FixOptions >
    FixRect = No
    CombText = No
```

```
    MapFonts = Yes
    MapChars = No
    FixLeftMargin = No
    LeftMargin = 0
    FixTopMargin = No
    TopMargin = 0
    DeleteFields = No
    ChkRect = Yes
    Color = No
    FixRectSaveBottom = No
    FixNegative = No
    Rotate90 = No
    FixCalcs = No
    YearDigit4 = Yes
    UniqueFieldNames = No
    FixDALCheck = No
    FixDALSyntax = No
< MasterResource >
    DefLib  = \fap\mstrres\sampco\deflib\
    FormLib = \fap\mstrres\sampco\forms\
    FontLib = \fap\mstrres\sampco\deflib\
    FxrFile = rel103
< MappedFonts >
     1   = 22010
    10   = 22008
    16   = 22008
    1006 = 21006
    1007 = 21007
    1008 = 21008
    1009 = 21009
    1010 = 21010
    1011 = 21011
    1012 = 21012
    1014 = 21014
    1016 = 21016
    1018 = 21018
    1024 = 21024
     (and so on…)
 < MappedChars >
    65 = 18
    69 = 22
    70 = 23
    73 = 26
    74 = 27
    75 = 28
     (and so on…)
```

Example

This example shows how to use the SX option to check DAL script syntax and correct errors. Note the second line of output. This line tells you that an attempt to correct the DAL syntax will be made.

```
fixfmw32 /i=ca2071 /x=rel103sm /p=fixform /sx
```

Here is a sample of the output:

```
** Check field DAL syntax **
 ** Attempt DAL syntax correction **
```

```
Successful load of D:\FixForm - Feature 1371\deflib\rel103sm.FXR
Successful load of D:\FixForm - Feature 1371\formlib\Ca2071.FAP
DAL Syntax error on field <VEHICLE NO.>
    Error 11:Invalid IF statement last token [end] at line 1 column 25
    Orig: return jcenter (@(), 9);end;
    Mods: return jcenter (@(), 9);;
Successful unload of D:\FixForm - Feature 1371\formlib\ca2071.FAP
```

The Mods line tells you how the DAL script syntax was modified.

Here is another example:

```
fixfmw32 /i=ca67a /x=rel103sm /p=fixform /sx
```

Here is a sample of the output:

```
** Check field DAL syntax **
 ** Attempt DAL syntax correction **
Successful load of D:\FixForm - Feature 1371\deflib\rel103sm.FXR
Successful load of D:\FixForm - Feature 1371\formlib\Ca67a.FAP
DAL Syntax error on field <NBR OF AUTOS>
    Error 11:Invalid IF statement last token [(EOF)] at line 1 column
102
    Orig: IF Numeric (@()) then return
(jcenter(format(@(),"n","zzz,zzz,zzz"),11));else return jcenter
(@(),11)
Successful unload of D:\FixForm - Feature 1371\formlib\ca67a.FAP
```

Here, the Mods line does not exist. The utility detected the DAL syntax error but did not know how to correct it. You must correct this error.

Example

This example shows how to use the SS option to check DAL script syntax:

```
fixfmw32 /i=ca2071 /x=rel103sm /p=fixform /ss
```

Here is a sample of the output:

```
** Check field DAL syntax **
Successful load of D:\FixForm - Feature 1371\deflib\rel103sm.FXR
Successful load of D:\FixForm - Feature 1371\formlib\Ca2071.FAP
DAL Syntax error on field <VEHICLE NO.>
    Error 11:Invalid IF statement last token [end] at line 1 column 25
Orig: return jcenter (@(), 9);end;
DAL Syntax error on field <VEHICLE NO. #002>
    Error 11:Invalid IF statement last token [end] at line 1 column 25
    Orig: return jcenter (@(), 9);end;
DAL Syntax error on field <VEHICLE NO. #003>
    Error 11:Invalid IF statement last token [end] at line 1 column 25
    Orig: return jcenter (@(), 9);end;
Successful unload of D:\FixForm - Feature 1371\formlib\ca2071.FAP
```

The second line of output tells you that the utility will only check for DAL syntax errors.

# FIXFXR

Use this utility to renumber or delete font IDs in a font cross-reference (FXR) file.

## Program names

| | |
|---|---|
| Windows | FIXFXR.EXE |

## Syntax

**FIXFXR /I /O /P**

| Parameter | Description |
|---|---|
| /I | Enter the name of the input font cross-reference (FXR) file. |
| /O | Enter the name of the output FXR file if different. Omit the extension. |
| /P | Enter the name of the INI file. The default is FIXFXR.INI. |

You can use this utility to maintain font cross-reference files (FXR) by renumbering or deleting font IDs. This utility takes a FXR file as an input file and then updates or creates a new FXR file based on settings in an INI file.

This utility works similarly to the FIXFORM utility. For more information, see FIXFORM on page 92.

The /P parameter specifies the INI file name. The default is FIXFXR.INI. Here is an example of a FIXFXR.INI file:

```
< FixOptions >
    MapFonts = Y
    DeleteFonts = Y
< MappedFonts >
    1 = 12304
    2 = 12305
    3 = 12306
    4 = 12307
< DeletedFonts >
    11004 = Y
    11005 = Y
    11006 = N
    11007 = Y
    11008 = N
```

In this example, font IDs 1 through 4 are renumbered to 12304-7 and font IDs 11004, 11005, and 11007 are deleted from the font cross-reference file (FXR).

# FIXOFFS

Use this utility to recalculate file offsets when you move files from one platform to another. This utility determines the new offsets within the NAFILE.DAT and POLFILE.DAT files and updates the other output files accordingly. For example, you must use this utility if you are running the GenArc program in an MVS environment using a local area network (LAN) archive of MVS-generated data.

This utility corrects offsets in the NEWTRN.DAT file (path and file name given in INI file) and any number of recipient batches specified in Fix_Batches control group. The location of the NEWTRN.DAT file is specified in the INI files. If the Fix_Batches control group is undefined, this utility updates all of the recipient batches defined in the Print_Batches control group.

**NOTE:** You can also use this utility to fix offsets in a VSAM NAFILE and POLFILE. To indicate that the NAFILE and POLFILE are VSAM files, use the /V parameter.

## Program names

| | |
|---|---|
| MVS | FIXOFFS |
| Windows | FIXOFW32.EXE |

## Syntax

**FIXOFW32 /INI /O /V /L**

| Parameter | Description |
|---|---|
| /? | (Help) Prints to the console all command line options and defaults. No transactions are processes when you include the ? parameter. |
| /INI | Tells the utility which INI file to use. |
| /O | Tells the utility to override the setting for the Outpath option in the FixOffsets control group with the path name you enter. |
| /V | Tells the utility that the NAFILE and POLFILE are VSAM files. If you omit this parameter, the utility checks for the NAFILE=DD:NAVSAM option in the VSAM control group. If the utility finds this INI option, it processes the NAFILE and POLFILE as VSAM files. |
| /L | Activates logging. The name you specify overrides the default log file name. If you include the option without specifying a file name, the utility uses the name specified in the INI file. If omitted from the INI file, the utility defaults to *FIXOFFS.LOG*.<br>When the LOG is activated the utility lists each transaction processed, showing the fields you specified in the LogFields option in the FixOffs control group. |

You can identify the various parameters using dashes (-) or backslashes (/). You can also identify parameters using order.

## Defining Parameters by Order

You can include command line options without flags. The utility assigns the parameters in this order if you omit the flags (- or /).

1 INI file

2 Output path

3 Log file

For example, if the INI file is set to *FSIUSER.INI*, the output path is set to *mypath*, and the log file is set to *mylog.log*, the command would look like this:

```
fixoffs  fsiuser.ini  mypath\  mylog.log
```

## Defining Parameters by Order and Flags

Some command line options may be used with flags while others or not. In this case fixoffs will look through the parameter list assigning specified parameters first. It assumes unspecified parameters to be values that have not been already been set with the specified parameters.

For example, if the log file has been specified but the INI file and output path have not, this works because the utility assigns the log file to *mylog* and then parses the remaining parameters in order, first the INI file, next the output path, and so on.

```
fixoffs fsiuser.ini -Log=mylog.log mypath\
```

## Mixing the Default Parameters

You do not have to specify all of the parameters. The utility defaults to the settings specified in the INI file.

For example, it you specified the OutPath option, the utility assumes the next parameter is in the INI file. Based on this example, the utility would look in the FSIUSER.INI file to find the name of the log file. If you omitted the name there, the utility would use the default, FIXOFFS.LOG.

```
fixoffs -outpath=mypath\  fsiuser.ini
```

## INI Options

This table explains the various options you can set.

| Control Group | Option | Description |
|---|---|---|
| Environment | FSISYSINI | Identifies the name and location of the FSISYS.INI file. |
| MasterResource | Deflib | Defines the path to the default files.<br>Defaults to ..*deflib*\. |
| Data | TrnDFDFile | Required. Identifies the TRNFILE (generated by GenTrn and read by GenData) and the NEWTRN file (written by GenData and read by GenWIP and GenArc).<br>If you omit the path, the utility looks in the \deflib directory. |
| | DataPath | Identifies the default location for the input and output files, such as the NAFILE.DAT, POLFILE.DAT, and recipient batch files.<br>Defaults to ..*data*\. |
| | NewTrn | Identifies the name of the NEWTRN.DAT file. If you omit the path, the utility looks in the directory you defined in the DataPath option. |
| | NAFILE | Required. Identifies the NAFILE.DAT file name and path. If you omit the path, the utility looks in the directory you defined in the DataPath option. |
| | POLFILE | Required. Identifies the POLFILE.DAT file name and path. If you omit the path, the utility looks in the directory you defined in the DataPath option. |
| Print_Batches | * | By default, all print batches identified under this group can be corrected (if located). All names will either contain an explicit path or should default to DATAPATH.<br>If omitted, the utility processes only the NEWTRN.DAT file. |
| DocsetNames | GroupName1<br>GroupName2<br>TransactionID | These options typically correspond to the Key1 (Company), Key2 (line of business), and KeyID (Policy number) fields. The utility uses these fields to get the DFD names for fields as the default for log information. |

| Control Group | Option | Description |
|---|---|---|
| FixOffsets | LogFields | This option overrides the default DocsetNames fields for log information. You can create a comma- or semicolon-delimited list of field names the utility will retrieve and print from each NewTrn record it corrects. The order of the names in your list determines the log output line.<br>Defaults to Company, LOB, PolicyNumber |
| | LogFile | Identifies the file which receives the logging information. Including this option enables the log option. If you omit this option, you can enable logging using the command line option.<br>Defaults to *FIXOFFS.LOG*. |
| | OutPath | The path for the output files the utility creates.<br>Defaults to the path you set in the DATAPATH option. |
| | NewTrn | Name of the new NEWTRN file.<br>Defaults to *FXNEWTRN.DAT*. If you omit the path, the utility, default to the OUTPATH option. |
| | X_OFFSET | Name of field that contains the offsets for the extract file. The NEWTRN.DAT file and the print batches must be sorted by this field.<br>Defaults to *X_OFFSET*. |
| | NA_OFFSET | Name of field that contains the offsets for the NA file.<br>Defaults to *NA_OFFSET*. |
| | LOGTRANSACTIONS | Log messages for each transaction, this option is overrode by command line parameter -LOG<br>Defaults to *No*. |
| | POL_OFFSET | Name of field that will contain the offsets for the POL file.<br>Defaults to *POL_OFFSET*. |
| Fix_Batches | * | This group and option will list all batches that are to be corrected. The option names (on the left of the equal) must match a known batch from Print_Batches. The option value (right side of the equation) will identify the new name and path for the output file. If no path is specified, default to OUTPATH.<br>Defaults to the Print_Batches control group, if you defined the OutPath option. If you defined the OutPath option, the utility assigns *FXBAT00*, *FXBAT01*, and so on. |

The utility displays a warning or an error message for any missing INI value it expected to find in the INI file. Here is an example:

```
WARNING  <FixOffsets> <NA_OFFSET> Not defined in INI file
Default to: NA_OFFSET
```

The second line indicates what the utility used as a default.

## Log File Entries

The utility makes three types of log file entries:

- Errors. The utility terminates when an error occurs. You should consider any data produced by the utility unreliable. Here is an example:

  ERROR  mainLoop End of File Never reached mainLoop .\data\NaFile.Dat

- Warning. This indicates the utility detected an unexpected condition but attempted to continue processing. You must analyze warning messages to determine if a real problem exists. Here is an example:

  ```
  WARNING  <FixOffsets> <NA_OFFSET> Not defined in INI file
  Default to: NA_OFFSET
  ```

- Information Messages. The utility displays information messages when you include the -LOG option on the command line. Informational messages are not preceded with *WARNING* or *ERROR*. Here is an example:

  ```
  This message is logged  for every transaction when the -LOG option is set.
  Processing:  <1234567> <LB1> <SAMPCO>
  Updated Input Batch=NewTrn.Dat  InputBatch=NEWTRN.Fix
  ```

## Warning and Error Messages

Here is a discussion of the warning and error messages you may encounter.

- Command Line Parameter Already Defined

  This warning message tells you a command line parameter has been entered more than once.

- Command Line Parameter Does Not have Value

  This warning message indicates you specified a command line parameter but omitted its value. Follow each command line parameter with an equals sign (=) and a value, as shown here:

  ```
  fixofw32 -ini=fsiuser.ini (correct)
  fixofw32 -ini fsiuser.ini (incorrect)
  ```

- Error in Logging

  This warning message indicates the utility was unable to record an entry to the log file.

- Input and Output Files have the Same Path and File Name

A warning issued if the output and input files you specified in the INI file are identical. For this to occur the Print_Batches control group must be identical to the Fix_Batches control group, and the OutPath option in the FixOffsets control group must be identical to the DataPath option in the Data control group. The utility modifies the offsets of the existing file.

- <FixOffsets><LogFields> Specified But Not Found in Record

  This warning message indicates the LogFields option in the FixOffsets control group defined fields which are not in defined in the TRNDFDFL.DFD or RCBDFDFL.DFD files. Since this affects only the logging of messages, the utility alerts you to the situation and continues processing.

- Unknown Command Line Parameter Ignored

  This warning message indicates you included a command line parameter the utility did not understand and therefore ignored. Use the Help (?) parameter or refer to the syntax discussion for a list of valid command line parameters.

- Not Defined in INI File

  A warning or an error message is issued if the utility cannot find an option it needs in the INI file. If the message says *WARNING*, the next entry indicates the value the utility will use as default.

- Cannot Add to Link List

  This error message occurs if the utility cannot create or add to a link list.

- Could Not Load INI Data

  This error message that indicates either the INI file could not be opened or a required option in the INI file is missing. If a required option is missing, look in the log file for information on the option you need to add.

- End of Docset Without Data

  This error message indicates an end of docset marker was found in the NAFILE.DAT file or POLFILE.DAT file without any preceding data.

- End of File Never Reached

  This error message indicates a file has unprocessed data. All input files must be from the same run of the GenData program.

- Failed to Get Record fgets

  This error message tells you the utility could not read a record from the NAFILE.DAT or POLFILE.DAT files. This message is followed by additional information which tells you which file produced the error.

- Failed to Return File Position ftell

  This error message indicates an ftell function call returned an error. This message is followed in the log file by an entry that indicates which file produced the error and information that describes the system-level error message.

- Failure to Update Record

  This error message indicates the utility could not change or add a record. The following error messages indicate the utility cannot open a file it needs and generate the *Failure to Update Record* error message. These messages indicate why the utility could not change or add a record in a database file and are used for debugging purposes.

  ```
  Could not store NAOFFSET field
  Could not store POLOFFSET field
  DBGotoNthRecord(%p, %p, %lu) - FAILED
  DBAdd(%p,%p,%u) - FAILED
  DBGetFirstRecord(%p,%p,%u) - FAILED
  DBGetCurrRecNum(%p, %p) - FAILED
  DBGetFieldDataPtr(%p, %p, %s) - FAILED
  Could not copy data to new record
  ```

- Fatal Error Cannot Continue

  This error message indicates an error occurred and processing was terminated. All error messages terminate processing, but since some messages may be used as warnings or errors this message accompanies the more serious errors.

- File Could Not be Opened

  This error message tells you the utility could not open a file it needs for processing. The utility includes the name of the file and the file definition, if applicable, as the next entry in the log file. If you encounter this error, make sure the files indicated in the log file are present and have proper permissions.

  The following error messages indicate the utility cannot open a file it needs and generate the *File Could Not be Opened* error message. These messages indicate why a database file was not opened and are used for debugging purposes.

  ```
  Invalid Record size returned by DBAllocateStructMemory
  DBAlloateStructMemory failed(%p, %p)
  DBOpen(%p, %d) failed
  DBInitializeFile(%s, %s, %p) failed
  DBQueryFormatInfo(%s) failed
  DBInitDB((FAPDBINSTALLER)ASCInstallHandler, ASCII) failed
  DBPutFieldData(%p,%p,%s,%p,%u) - FAILED
  ```

- File Not Sorted by X_Offset (file name)

  This error message indicates the utility detected a record that was not sorted by the X_OFFSET field. The NEWTRN.DAT file and all batch files must be sorted by the X_OFFSET field.

- Output File Defined in Fix_Batches Without Matching Entry in Print_Batches

  An error appears if you define a batch option in your Fix_Batches control group without defining an identical option in the Print_Batches control group. For example, based on the settings below, the BATCH2 option causes an error because it is not defined in the Print_Batches control group:

```
< Fix_Batches >
    Batch1 = BAT1.BCH
    Batch2 = BAT2.BCH <<--- Illegal
< Print_Batches >
    Batch1 = BAT1.BCH
```

Example

In this example, the utility updates the NEWTRN.DAT file and the first recipient batch. The output files will have the same name as the input files, but will be stored in the \fixout directory. This example assumes the input files are in the \data directory.

```
< Data >
    DataPath =.\data
    NewTrn  = newtrn.dat
< Print_Batches >
    Batch1 = BATCH1.BCH
    Batch2 = BATCH2.BCH   <- Note: This batch will not be corrected.
< FixOffsets>
    Outpath=.\fixout
    NewTrn = newtrn.dat
< Fix_Batches >
    Batch1=BATCH1.BCH
```

Normally this utility creates new files with the corrected offsets and leaves the original files intact. You can, however, have the utility overwrite the original files by specifying the same file names and paths for input and output. These INI settings force the utility to alter the input file instead of creating a new file:

```
< Data >
    DataPath =.\data
    NewTrn  = newtrn.dat
< Print_Batches >
    Batch1 = BATCH.BCH
    Error = ERROR.BCH
< FixOffsets >
    Outpath=.\data
    NewTrn = newtrn.dat
< Fix_Batches >
    Batch1 = BATCH.BCH
    Error = ERROR1.BCH
```

If the FixOffsets control group has not been set up in the INI file, the utility will use the default file names and will write all output to the \data directory.

```
< Data >
    DataPath =.\data
    NewTrn  = newtrn.dat
< Print_Batches >
    Batch1 = BATCH.BCH
    Error = ERROR.BCH
```

The corrected version of the NEWTRN.DAT file is named *FXNEWTRN.DAT*. The corrected version of Batch1 is named *FXBAT00*. The corrected version of Error is named *FXBAT01*.

# FONTLIST

Use this utilty to create a report which lists the fonts used in the specified FAP files. This report includes information on...

- The font IDs used in each FAP file
- The font IDs used in all specified FAP files
- A list of the font IDs contained in the FXR file but not used in the FAP files
- A list of all FAP files that use a specific font ID (optional)

## Syntax

**FONTLIST /I /X /O /F**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FAP file from which you want to compile a list of fonts. You can use wild cards (*) in the name to specify multiple files. |
| /X | Enter the name of the font cross-reference (FXR) file to search. |
| /O | Enter the name you want to assign to the report. The default report name is FONTLIST.DAT. |
| /F | (Optional) Enter a specific font ID to search for. |

## Example

For example, this command:

```
fontlist /i=forms\c* /x=deflib\rel102sm /o=.\fonts.txt /f=11020
```

would produce a report that looks similar to this one:

```
--- FONTLIST Copyright (C) 1997-2006 Docucorp International
--- Produces a font usage report

FontList Utility

FAP: D:\FAP\mstrres\RPEX1\forms\c*
FXR: D:\FAP\mstrres\RPEX1\deflib\rel102sm

FAP and Font ID Report
----------------------

Fap: CODE128
12012           Courier 12 PT

Fap: Checkbox
11010           Times-Roman 10 PT
11020           Times-Roman 20 PT

Fap: Color
11020           Times-Roman 20 PT

Fap: cuscolor
16016           Univers-Medium 16 PT

Fap Count: 0000000005

Consolidated List of Fonts Report
```

```
--------------------------------
11010           Times-Roman 10 PT
11020           Times-Roman 20 PT
12012           Courier 12 PT
16016           Univers-Medium 16 PT

Fap List for Font ID 11020 Report
--------------------------------
Color
Checkbox

Fonts Included in FXR - Not Found in Fap(s)
-----------------------------------------
11006           Times-Roman 6 PT
11008           Times-Roman 8 PT
11012           Times-Roman 12 PT
11014           Times-Roman 14 PT
< and so on... >
```

# FRM2FAP

Use this utility to convert a version 1 or version 2 Xerox form (FRM) file into a FAP file. This utility is useful if you have existing form files in FRM format and you want to convert those forms into FAP files.

## Program names

| | |
|---|---|
| Windows | FRM2FAPW.EXE |

## Syntax

**FRM2FAPW /I /X**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FRM file. Omit the extension. |
| /X | Enter the name of the font cross-reference (FXR) file. Omit the extension. |
| /F | Enter the default font name to use if the logo is not found. |
| /P | Enter the PrtType control group to use in the INI file, such as *XER*. |

This utility requires these files:

- FSISYS.INI
- Font cross-reference file, such as REL103.FXR

In addition to the FRM file, you must specify the font cross-reference file (FXR) which contains information about the Xerox fonts used in the Xerox FRM file. The FXR file contains information about the fonts which can be used in a FAP file. FRM files also include the names of the Xerox fonts used in the FRM file.

When the utility finds a Xerox font in the FRM file which is not included in the FXR file, it assumes it must be a font which the system will represent as a logo. Logos can be converted to Xerox fonts but this information is not stored in the FXR file. Therefore, the utility looks for the missing Xerox font name in the LOGO.DAT file.

Keep in mind you must convert the Xerox logos (LGO) used in a Xerox FRM file into Documaker logos (LOG files) before you use this utility.

**NOTE:** Logos are used to print pictures, signatures, company logos, and so on. The FRM file may contain these pictures by using the Xerox fonts or logos created for the picture. If so, the names of these Xerox fonts or logos need to be added to the LOGO.DAT file.

If the utility finds the Xerox name in the LOGO.DAT file, it creates a logo record in the FAP file. If it does not find this Xerox name in the LOGO.DAT file, it creates a text label in the FAP file which contains the name of the missing Xerox font file.

To do this, the utility needs a font ID to use for the text label. The /F parameter tells the utility which font ID to use if it has to create text labels for the missing Xerox font files.

**NOTE:** A way to determine missing fonts is to provide the /F parameter with a font ID from the FXR. If you choose a font ID with a large point size, you can easily spot the text labels which contain the names of the missing fonts. If the missing font is a normal Xerox font, simply import the Xerox font into the FXR file. If the missing font is one of these picture fonts, add its name to the LOGO.DAT file. Then rerun FRM2FAP with the improved FXR and/or LOGO.DAT files. When the FXR and LOGO.DAT files contain all of the fonts used in the FRM file, you will have a properly converted FAP file.

If you omit the output file parameter, the utility uses the FRM file name with an FAP extension.

The FRM2FAP utility looks for a control group named PRINTER in the FSISYS.INI file. In the PRINTER control group, the utility looks for the PrtType option, which determines the type of printer being used, such as AFP, XER, or PCL. Use the /P parameter to specify which PrtType control group to use.

# FRMDUMP

Use this utility to create a text file from a version 1 or version 2 Xerox form printer resource (FRM) file.

Similar to the AFPDUMP utility, this utility reads a Xerox Metacode FRM form printer resource file and outputs a formatted text file which contains a list of the fonts and images used as well as positioning information for text, lines, and images.

## Program names

| | |
|---|---|
| Windows | FRMDPW32.EXE |

## Syntax

**FRMDPW32 /I /O**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FRM file. |
| /O | (Optional) Enter the output file name with a TXT extension. |

If you omit the /O parameter, the utility uses the FRM file name with the extension *TXT*.

# FSIVER

Use this utility to generate a report that shows version and patch level information for the Documaker RP or IDS products. Patches are corrections that have been made to the product's program files (executable and DLL files) after the initial release of the product.

The FSIVER utility reads the program files and generates a report showing which patches have been applied to these program files. If you contact Skywire Software Support, they may ask you to run this utility to determine what patch level your system is at.

On Windows, the program files consist of executable files (.exe) and dynamic link library files (.dll). On Unix, the program files consist of executable files (usually no extension) and shared object files (usually a .so extension). On MVS (OS/390, z/OS), the program files are statically linked so there are no DLL files, only executable files.

Generally, two sections are shown in the report. The first section is called the detailed report and it shows version and patch information for each program file. The second section is called the summary report and it includes a listing of the patches that have been applied to the system as a whole.

## Program names

| | |
|---|---|
| Windows | FSIVRW32.EXE |
| UNIX | fsiver |
| OS/390 | FSIVER |

## Syntax

**FSIVRW32 /I= /O= /VO /PO /SO /NV /NP /NS >xxx**

| Parameter | Description |
|---|---|
| I= | (Optional) Enter the name and path of the file or files you want to check. You can include asterisks as wildcards. If you omit this parameter, the utility checks a predefined set of known program files for version and patch information. |
| O= | (Optional) Lets you specify a file to capture the output from this utility. Use this option if you want to send the results to a text file. |
| VO | (Optional) Only produce the version report. |
| PO | (Optional) Only produce the patch report. |
| SO | (Optional) Only produce the summary patch report. |
| NV | (Optional) Do not produce the version report. |
| NP | (Optional) Do not produce the patch report. |
| NS | (Optional) Do not produce the summary patch report. |
| >xxx | (Optional) This parameter lets you specify an alternative method of capturing the output from the utility. For instance,<br>`> version.txt`<br>tells the system to save the output to a file named *version.txt*. |

This utility produces these reports:

- Version report - includes basic version information for each program file
- Patch report - shows the patches (P01, P02, and so on) applied to each program file. The report is sorted in patch number order.
- Summary patch report - lists, in numerical order, all of the patches applied to the program files. The report is sorted by product in patch number order. Missing patches are noted.

In addition to showing patch information for Documaker RP and IDS products, this utility also includes information about patches made to some common, low-level libraries. These program files are called *3rd party libraries* and are used by both products. The 3rd party patches are reported in this format:

```
3RD PATCH 3RD:Pxx
```

where *xx* is the patch number.

**NOTE:** The utility tries to note missing patches. For example, if P03 and P05 are found but not P04, the utility notes that P04 was not detected. Missing patches do not necessarily indicate an error because some patches only apply to one platform and not others.

## Running FSIVER on MVS (OS/390, z/OS)

The JCL for running FSIVER on MVS is provided in member FSIVERX of the JCLLIB dataset. Here is an example:

```
//ZDA JOB (33005),'FSIVER - 110 ',CLASS=T,MSGCLASS=X,
// NOTIFY=&SYSUID
//*
// SET HLQ='FSI.V110' <== SET HIGH LEVEL QUALIFIER
// SET RES='RPEX1' <== SET RESOURCE (E.G. RPEX1, UTEX1)
//*
// JCLLIB ORDER=&HLQ..PROCLIB
//*
//
*****************************************************************
//* PROGRAM : FSIVER
//* PURPOSE : CREATES A REPORT THAT LISTS WHICH PATCHES HAVE BEEN
//* APPLIED TO THE PROGRAMS IN THE LINKLIB REFERENCED BY
//* THE LINKLIB DD STATEMENT.
//*
//* PARMS : /I=PROGRAM (NAME OF MEMBER IN DD:LINKLIB)
//* OR '*' TO LIST PATCH LEVEL OF ALL PROGRAMS IN
//* DD:LINKLIB.
//*
//
*****************************************************************
//FSIVER EXEC PGM=FSIVER,PARM='/ /I=*'
//*
//STEPLIB DD DSN=&HLQ..LINKLIB,DISP=SHR
// DD DSN=SYS1.SCEERUN,DISP=SHR
//LINKLIB DD DSN=&HLQ..LINKLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Keep in mind:

- The LINKLIB DD statement should point to the dataset that contains the program files, such as *FSI.V110.LINKLIB*.
- The report is written to the SYSPRINT DD statement.

## Example

Here is an example of how you can use this utility:

```
FSIVRW32 /i=C:\FAP\DLL\*.* > VERSION.TXT
```

This tells FSIVER to read all the program files in the \FAP\DLL\ directory on the C: drive. The report is written to the file named *version.txt*. Here is an excerpt from the report:

```
--- DocuCorp FSIVER Utility Program (C)  ---
--- Display Version & Patch Level Report ---

Version Report For : A2WBW32.DLL
--------------------------------
C:\RELEASE\REL110\RPS100\SHIPW32\A2WBW32.DLL 40110010 400.110.010
Nov 24 2004
   > 00:50:46

Patch Report   For : C:\RELEASE\REL110\RPS100\SHIPW32\A2WBW32.DLL
-----------------------------------------------------------------
No Patches Detected For :
C:\RELEASE\REL110\RPS100\SHIPW32\A2WBW32.DLL

Version Report For : AFEW32.DLL
-------------------------------
C:\RELEASE\REL110\RPS100\SHIPW32\AFEW32.DLL 40110010 400.110.010 Jul
25 2005
   > 14:01:32

Patch Report   For : C:\RELEASE\REL110\RPS100\SHIPW32\AFEW32.DLL
---------------------------------------------------------------
DAP PATCH 11.0:P03:PCR16419:400.110:..\C\afedform.c:Jun 17
2005:15:16:39:
DAP PATCH 11.0:P03:PCR16335,PCR16378:400.110:..\C\afedpw.c:Jun 30
2005:10:26:51:
DAP PATCH 11.0:P04:PCR16457:400.110:..\C\afedform.c:Jun 17
2005:15:16:39:
DAP PATCH 11.0:P04:PCR15354:400.110:..\C\afedupfm.c:Jun 30
2005:10:14:30:
DAP PATCH 11.0:P04:PCR16457:400.110:..\C\afetrans.c:Mar  4
2005:10:16:07:
DAP PATCH 11.0:P05:PCR16711:400.110:..\C\afedform.c:Jun 17
2005:15:16:39:
DAP PATCH 11.0:P05:PCR16615,PCR16670:400.110:..\C\afedpw.c:Jun 30
2005:10:26:51:
DAP PATCH 11.0:P06:PCR15354:400.110:..\C\afedupfm.c:Jun 30
2005:10:14:30:
DAP PATCH 11.0:P07:PCR16913:400.110:..\C\afeprint.c:Mar  9
2005:12:14:46:
DAP PATCH 11.0:P10:PCR17065:400.110:..\C\afedform.c:Jun 17
2005:15:16:39:
DAP PATCH 11.0:P15:PCR17025:400.110:..\C\afedpw.c:Jun 30
2005:10:26:51:
DAP PATCH 11.0:P18:PCR17557:400.110:..\C\afeversn.c:Jul 25
2005:14:01:32:
```

```
DAP PATCH 11.0:P19:PCR17601:400.110:..\C\afedform.c:Jun 17
2005:15:16:39:
DAP PATCH 11.0:P21:PCR17727:400.110:..\C\afedal.c:Jul 25
2005:14:01:21:
DAP PATCH 11.0:P21:PCR17727:400.110:..\C\afedupfm.c:Jun 30
2005:10:14:30:
DAP PATCH 11.0:P21:PCR17689:400.110:..\C\afeentry.c:Jun 30
2005:15:31:22:
DAP PATCH 11.0:P24:PCR14945:400.110:..\C\afedal.c:Jul 25
2005:14:01:21:

. . .

Version Report For : P417W32.DLL
--------------------------------
*** Version information unavailable for : P417W32.DLL

Patch Report   For : C:\RELEASE\REL110\RPS100\SHIPW32\P417W32.DLL
-----------------------------------------------------------------
No Patches Detected For :
C:\RELEASE\REL110\RPS100\SHIPW32\P417W32.DLL


Summary Patch Report:
---------------------
3RD PATCH 3RD:P01        * Not detected, see explanation below.
3RD PATCH 3RD:P02        * Not detected, see explanation below.
3RD PATCH 3RD:P03        * Not detected, see explanation below.
3RD PATCH 3RD:P04
3RD PATCH 3RD:P05        * Not detected, see explanation below.
3RD PATCH 3RD:P06
3RD PATCH 3RD:P07        * Not detected, see explanation below.
3RD PATCH 3RD:P08
3RD PATCH 3RD:P09        * Not detected, see explanation below.
3RD PATCH 3RD:P10
3RD PATCH 3RD:P11
3RD PATCH 3RD:P12
3RD PATCH 3RD:P13        * Not detected, see explanation below.
3RD PATCH 3RD:P14        * Not detected, see explanation below.
3RD PATCH 3RD:P15
3RD PATCH 3RD:P16        * Not detected, see explanation below.
3RD PATCH 3RD:P17
3RD PATCH 3RD:P18
3RD PATCH 3RD:P19
3RD PATCH 3RD:P20
3RD PATCH 3RD:P21        * Not detected, see explanation below.
3RD PATCH 3RD:P22
3RD PATCH 3RD:P23        * Not detected, see explanation below.
3RD PATCH 3RD:P24
DAP PATCH 11.0:P01
DAP PATCH 11.0:P02
DAP PATCH 11.0:P03
DAP PATCH 11.0:P04
DAP PATCH 11.0:P05
DAP PATCH 11.0:P06
DAP PATCH 11.0:P07
```

```
DAP PATCH 11.0:P08
DAP PATCH 11.0:P09
DAP PATCH 11.0:P10
DAP PATCH 11.0:P11
DAP PATCH 11.0:P12
DAP PATCH 11.0:P13
DAP PATCH 11.0:P14
DAP PATCH 11.0:P15
DAP PATCH 11.0:P16
DAP PATCH 11.0:P17
DAP PATCH 11.0:P18
DAP PATCH 11.0:P19
DAP PATCH 11.0:P20
DAP PATCH 11.0:P21
DAP PATCH 11.0:P22
DAP PATCH 11.0:P23
DAP PATCH 11.0:P24

* When a patch is identified as 'Not detected', it means
  that either the patch is not applicable to your system
  or the patch has been omitted.

--- FSIVER Completed ---
```

# FXLOGREF

Use this utility to change embedded logos in a FAP file into referenced logos. This utility searches a FAP file for embedded logos and either removes...

- Any duplicate logos are changed to reference the first occurrence of the embedded logo.
- All embedded logos and instead references the logos to external logo files it creates.

The resulting FAP file will be identical in appearance, but smaller in size if duplicate embedded logos are found and removed.

You can also use this utility to change your logo files without having to update the FAP files. If a logo is embedded, the only way to remove it is by deleting it and then insert a new reference. If it is an external reference, then you can update the logo file without having to edit each FAP file that might use that logo.

## Program names

Windows FXLOGREF.EXE

## Syntax

**FXLOGREF /F /O /C /L /I**

| Parameter | Description |
|---|---|
| /F | Enter the name of the input FAP file. |
| /O | Enter the name of the output file. |
| /C | Compares the embedded logos to all of the logos in the directory to find duplicates. |
| /L | Include this option to remove all embedded logos and replace them with references to external files (*.LOG) the utility creates. |
| /I | Enter the name of the INI file to use. The default is the FAPCOMP.INI file. |

**NOTE:** This utility can be useful if you are using a newer version of the system to create FAP files for an older version. For instance, if you use the RTF import feature to import a file, it will embed logos for you.

If, however, you then plan to use the FAP file you created in a version that predates version 10.1, you would need to use this utility to remove the embedded logos. (Prior to 10.1 the system did not support embedded logos.)

# FXRCMP

Use this utility to compare two font cross-reference (FXR) files and print the differences to a file you specify. The utility also compares matching font IDs. The output contains the names of the DAP structure for the FXR.

## Program names

Windows FXRCMPW32.C

## Syntax

**FXRCMP32 /1 /2 /INI /O /M /V /C**

| Parameter | Description |
|---|---|
| /1 | Enter the name of the first FXR file. |
| /2 | Enter the name of the second FXR file. |
| /INI | (Optional) Enter the name of the INI file you want the utility to use. The default is the FAPCOMP.INI file. |
| /O | (Optional) Enter the name of the output file in which you want this utility to print its results. The default is TEMP.TXT. |
| /M | (Optional) Include this parameter if you want the utility to only report the differences between the fonts used in both FXRs. Fonts which appear in only one of the FXRs are excluded from the comparison. |
| /V | (Optional) Include this parameter if you want the utility to make an extensive or verbose comparison. |
| /C | (Optional) Include this parameter if you want the utility to compare for case sensitivity |

Here is an example of the results you will see when you use the enter this command:

FXRCMP32 /1=FirstFXR /2=SecondFXR

The output contains the names of the DAP structure for the FXR.

```
Mon May 01 11:27:26 2005
PLEASE NOTE: Results are listed in order by Font ID with
the contents of the first fxr listed and then the contents
of the second fxr listed.
    Font ID: #, <rel103>:<test>

Font ID: 1, FAPFONTHDR->fntName <CO_____.PFB>:<COURIE.TTF>
Font ID: 1, FAPFONTATTR->chrwid:code point<127> width <136>:<80>
Font ID: 1, FAPFONTATTR->chrwid:code point<175> width <64>:<80>
Font ID: 1, FAPFONTATTR->internalLeading <32>:<16>
Font ID: 1, FAPFONTPRT<PS>->typefaceCode <0>:<>
Font ID: 1, FAPFONTPRT<PS>->fontFile <CO_____.PFB>:<>
Font ID: 1, FAPFONTPRT<PS>->charSetID <W1>:<>
Font ID: 1, FAPFONTPRT<PS>->SetupData <Courier>:<>
Font ID: 1, FAPFONTPRT<OTHER>->typefaceCode <>:<0>
Font ID: 1, FAPFONTPRT<OTHER>->fontFile <>:<COURIE.TTF>
Font ID: 1, FAPFONTPRT<OTHER>->charSetID <>:<W1>

Font ID: 2, FAPFONTHDR->fntName <CO_____.PFB>:<COURIE.TTF>
```

```
Font ID: 2, FAPFONTINF->height <184>:<176>
Font ID: 2, FAPFONTATTR->chrwid:code point<32> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<33> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<34> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<35> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<36> width <104>:<96>

May 01 11:27:26 2000
PLEASE NOTE: Results are listed is the following format/order:
       <c:\012600ps.fxr>:<c:\012600tt.fxr>

Font ID: 1, FAPFONTHDR->fntName <CO______.PFB>:<COURIE.TTF>
Font ID: 1, FAPFONTATTR->chrwid:code point<127> width <136>:<80>
Font ID: 1, FAPFONTATTR->chrwid:code point<175> width <64>:<80>
Font ID: 1, FAPFONTATTR->internalLeading <32>:<16>
Font ID: 1, FAPFONTPRT<PS>->typefaceCode <0>:<>
Font ID: 1, FAPFONTPRT<PS>->fontFile <CO______.PFB>:<>
Font ID: 1, FAPFONTPRT<PS>->charSetID <W1>:<>
Font ID: 1, FAPFONTPRT<PS>->SetupData <Courier>:<>
Font ID: 1, FAPFONTPRT<OTHER>->typefaceCode <>:<0>
Font ID: 1, FAPFONTPRT<OTHER>->fontFile <>:<COURIE.TTF>
Font ID: 1, FAPFONTPRT<OTHER>->charSetID <>:<W1>

Font ID: 2, FAPFONTHDR->fntName <CO______.PFB>:<COURIE.TTF>
Font ID: 2, FAPFONTINF->height <184>:<176>
Font ID: 2, FAPFONTATTR->chrwid:code point<32> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<33> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<34> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<35> width <104>:<96>
Font ID: 2, FAPFONTATTR->chrwid:code point<36> width <104>:<96>
```

# FXRVALID

Use this utility to check a font cross-reference (FXR) files for settings which would cause problems. This includes problems affecting Adobe© Acrobat© (PDF) files, such as noting any font ID which would cause problems when during the creation of a PDF files.

## Program names

Windows FXRVALDW.EXE

## Syntax

**FXRVALDW /I /E /G /O /R /D?**

| Parameter | Description |
|---|---|
| /I | The name of the font cross-reference (FXR) file. Omit the extension. |
| /E | (Optional) An error file name. Omit the extension. |
| /G | Turns on the adding of “OTH” entry and the grouping of fonts. You can specify the grouping threshold as an error percentage (see Using grouping on page 120 for more information). The default is zero (0). The default range is 32,127. |
| /O | (Optional) An output file name. The new FXR file contains “OTH” entries and grouping. If you omit the file name, the utility uses the input file name with an *FXX* extension. If you include a file name without an extension, they utility defaults to *FXX*. |
| /R | (Optional) Use this parameter (startchar,endchar) to specify the range of characters in width table to be checked for grouping. You can enter any integer from 0 to 255. The default value for *startchar* is 32 and the default value for *endchar* is 127. If endchar is less than startchar, the value of endchar is set to that for startchar. |
| /D? | Turns on the DownloadFont option, known as the Option field in the “OTH” entry, in every “OTH” entry. The DownloadFont option in every “OTH” entry is turned off if you omit this parameter. |

The FXRVALID utility performs several checks on font IDs in the font cross-reference (FXR) file. The following topics discuss the various checks the utility performs.

## Check typeface

This check makes sure all font IDs contain one of the following PostScript font names in the Font Name field for PostScript printing:

| | |
|---|---|
| Courier | Times-Roman |
| Courier-Bold | Times-Bold |
| Courier-BoldItalic | Times-Italic |
| Courier-Oblique | Times-BoldItalic |
| Courier-BoldOblique | Symbol |
| Courier-Italic | ZapfDingbats |
| Helvetica | Univers-Medium |
| Helvetica-Bold | Univers-Bold |

| | |
|---|---|
| Helvetica-Oblique | Univers-MediumItalic |
| Helvetica-BoldOblique | Univers-BoldItalic |

The FXRVALID utility tells you via an error message if the FXR file contains an invalid PostScript font name or does not contain a PostScript font. The message also tells you whether a fixed or proportional font will be used in place of the invalid typeface. The PDF printer driver will make the font substitution.

**NOTE:** Font IDs have either a fixed pitch or a proportional spacing value. If font substitution is required for fixed pitch fonts, Courier is typically used. If font substitution is required for proportional fonts, Helvetica is typically used. In addition, the stroke weight and style settings of the font ID are checked to see if bold or italic or bold and italic versions of these fonts should be used.

### Check point size

This check compares the font height to the point size for each PostScript font in the FXR. A warning message appears for every font ID whose font height differs from the point size by a factor of 1/3 or greater. A warning also appears if the font height equals zero.

**NOTE:** If the font height and point size do differ by the factor of 1/3, the printer driver will use font height to determine point size. The FXRVALID utility does not determine point size in these situations.

### Check the codepage

A warning appears for any font IDs whose codepage field is not empty or is not set to 1004.

**NOTE:** The PDF printer driver uses the ANSI codepage for text. Codepage 1004 is the IBM codepage which is equivalent to the ANSI codepage. Some older version FXRs, built before the Codepage field was added to the FXR, have a blank in the Codepage field.

### Check spacing

This check makes sure the spacing value (fixed or proportional) of the font ID matches a PostScript font with an equivalent spacing style. If the spacing value does not match, a warning appears.

### Check style

This check makes sure the font style (upright or italic) of the font ID matches a PostScript font with an equivalent font style. If a font ID specifies an italic style, a warning appears if the Font Name field does not contain a PostScript font name containing the word *Italic* or *Oblique*. If a font ID specifies an upright style, a warning appears if the Font Name field contains the word *Italic* or *Oblique*.

### Check weight

This check makes sure the font weight (bold or normal) of the font ID matches a PostScript font with an equivalent font weight. If a font ID specifies a bold style, a message appears if the Font Name field does not contain a PostScript font name which includes the word *Bold* and vice versa.

### Using grouping

When grouping fonts, the utility performs these steps:

1. Creates “OTH” entries for every font record in the FXR file.
2. Groups fonts (set the font index in “OTH” entry the same for each group) if the following conditions are true:
   - Font spacing (proportional or fixed) is the same.
   - Font style (italic or upright) is the same.
   - Font family is the same.
   - Font stroke weight (bold or non-bold) is the same.
   - The percentage difference of the *absolute width* for every character within the user-specified range is not greater than the user-specified grouping threshold.

**NOTE:** Absolute width is defined as: the value in the width table divided by the point size.

3. Sets the Font File field the same for each group. The Font File field (PostScript or TrueType font) that has the smallest absolute width for the *W* character in each group is copied to every font in the same group. This font is called the *base font*.

   If there is no entry in the Font File field in the base font, the FXRVALID utility tries to find a Font File field in the group and use it as base font file. If none is found in the entire group, the Font File field in “OTH” entry for this group is left empty. It is your responsibility to fill in the Font File field in this situation. If you do not, the system may create an invalid PDF file.

### Example

In a Windows environment, you would enter:

```
FXRVALDW /I=rel103
```

In this example, FXRVALDW checks the font cross-reference file named REL103.FXR and creates an error file named REL103.ERR.

# KSDS2SEQ

Use this utility to read the records of a VSAM KSDS and write the records to a sequential file. The utility removes the index before the record is written.

**NOTE:** This utility is only available for MVS systems.

## Example

For example, you could run this utility against a VSAM NAFILE and a VSAM POLFILE to produce sequential copies of those files. You could then download the sequential copies from an MVS system to a Windows Server, where the GenPrint or GenArc programs could use the sequential NAFILE and POLFILE files.

Here is a copy of the JCL. You can find sample JCL for the KSDS2SEQ utility in member KSDS2SQX of JCLLIB.

```
//* COPY JOBCARD HERE
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*      KSDS2SEQ -
//*
//*      PARAMETERS:
//*         /M=MAX   WHERE MAX IS THE MAXIMUM RECORD LENGTH OF THE
//*                  OUTPUT DATASET (DEFAULT = 512).
//*
//*        JOB PERFORMS 2 STEPS :
//*
//*        1.  DELETES / RE-DEFINES SEQUENTIAL OUTFILE.
//*        2.  RUNS KSDS2SEQ PROGRAM TO COPY VSAM KSDS TO SEQUENTIAL
//*            FILE.  CHANGE FILE NAMES APPROPRIATELY.
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*
//DEL     EXEC PGM=IEFBR14
//OUTFILE  DD DSN=&HLQ..&RES..GENDATA.NAFILE,DISP=(MOD,DELETE),
//         UNIT=SYSDA,SPACE=(TRK,0)
//*
//KSDS2SEQ EXEC PGM=KSDS2SEQ,PARM='/ /M=512'
//STEPLIB  DD  DISP=SHR,DSN=&HLQ..LINKLIB
//         DD  DISP=SHR,DSN=SYS1.SCEERUN
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//INFILE   DD DSN=&HLQ..&RES..GENDATA.NAFILE.KSDS,DISP=SHR
//OUTFILE  DD DSN=&HLQ..&RES..GENDATA.NAFILE,DISP=(,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(5,1)),
//         RECFM=FB,LRECL=512,BLKSIZE=23040
```

# LBRYMGR

Use the LBRYMGR utility to create a response file from a FORM.DAT file. Response files define the file references to include in a particular resource library. By running the response file using Library Manager, you can insert or check in numerous file references in a resource library.

**NOTE:** For more information about Library Manager, see the Docucreate User Guide.

When you create a response file from a FORM.DAT file, you extract FORM.DAT information to build a the response file (RSP). You can then edit the response file to add the information not included in the FORM.DAT file.

## Program names

Windows LBRYMGRW.EXE

## Syntax

**LBRYMGRW /LBY /RSP /FORMDEF /REC /S /SP /XP /INI /ALL /FAP /LOGO /DDT /BDF /GRP /FOR /EFF /NOSTATS /NOLOG /R /V /C /DAL /REV /VER /MODE /STATUS /CLASS /PROJECT /LN /EXP /CVTOLD**

| Parameter | Description |
|---|---|
| /LBY | The name of the library file. Include the *LBY* extension |
| /RSP | The name of the response file you want to build. Include the *RSP* extension. The extension is optional but will help you identify the file you created. |
| /FORMDEF | The name and path of your form definition (FORM.DAT) file, such as *\fap\mstrres\rpex1\deflib\form.dat*. |
| /REC | Lets you specify whether the utility creates Add records (REC=A), Extract records (REC=X), or Sync records (REC=S). The default is A. |
| /S | Tells the utility to write synchronization records to the response file. Use the /REC=S parameter instead of this one. |
| /SP | Specifies the path into which the utility unloads Library Manager objects into temporary files (file names are of the pattern F1.SYN, F2.SYN, and so on). |
| /XP | Specifies the path for target file names when creating extract records. |
| /INI | The name and path of the INI file, such as *\fap\dll\fapcomp.ini*. |
| /ALL | If you include just ALL, the utility includes all files in the response file. If you specify a path after the parameter, as shown here:<br>`/ALL=d:\data`<br>the utility builds add records using all files from the given path or builds extract records and uses the path you specified in the target file name. |
| /FAP | If you include just FAP, the utility includes FAP files in the response file. If you specify a path after the parameter, as shown here:<br>`/FAP=d:\data`<br>the utility either builds add response records using FAP files from the given path, or it builds extract response records using the path in the target file name. |

| Parameter | Description |
|---|---|
| /LOGO | If you include just LOGO, the utility includes LOG files in the response file. If you specify a path after the parameter, as shown here:<br>`/LOGO=d:\data`<br>the utility either builds add response records using LOG files from the given path, or it builds extract response records using the path in the target file name. |
| /DDT | If you include just DDT, the utility includes DDT files in the response file. If you specify a path after the parameter, as shown here:<br>`/DDT=d:\data`<br>the utility either builds add response records using DDT files from the given path, or it builds extract response records using the path in the target file name. |
| /BDF | If you include just BDF, the utility includes BDF files in the response file. If you specify a path after the parameter, as shown here:<br>`/BDF=d:\data`<br>the utility either builds add response records using BDF files from the given path, or it builds extract response records using the path in the target file name. |
| /GRP | If you include just GRP, the utility includes GRP files in the response file. If you specify a path after the parameter, as shown here:<br>`/GRP=d:\data`<br>the utility either builds add response records using GRP files from the given path, or it builds extract response records using the path in the target file name. |
| /FOR | If you include just FOR, the utility includes FOR files in the response file. If you specify a path after the parameter, as shown here:<br>`/FOR=d:\data`<br>the utility either builds add response records using FOR files from the given path, or it builds extract response records using the path in the target file name. |
| /EFF | Lets you specify an effective date for the objects you are adding to the library. Enter the effective date in one of these formats: *mm/dd/yy*, *mm/dd/yyyy,* or *yyyymmdd*. If you omit this parameter, the utility uses the current date as the effective date. |
| /NOSTATS | Include this parameter to suppress the statistics report which appears when the utility finishes. |
| /NOLOG | Include this parameter to suppress logging, which otherwise goes into the trace file. |
| /R | Enter LAST if you want the utility to generate sync records for only the latest revision of the resource. Enter E if you want to it to generate records only for expired resources.<br>The default is ALL which generates records for all revisions of the resource. |
| /V | Enter LAST if you want the utility to generate sync records for only the latest version of the resource.<br>The default is ALL which generates records for all versions of the resource. |
| /C | Include this parameter to display a counter which shows the progress of response file processing. |

| Parameter | Description |
|---|---|
| /DAL | If you include this parameter and specify a path, the utility builds Add records using DAL files from the given path. |
| /REV | When generating Add records, this parameter tells the utility what revision to set for the resources added to the library. The default is 00001. |
| /VER | When generating Add records, this parameter tells the utility what version to set for the resources added to the library. The default is 00001. |
| /MODE | This parameter tells the utility to set the mode when creating Add records or, when creating Extract records, to only create records for objects with the given mode. |
| /STATUS | This parameter tells the utility to set the status when creating Add records or, when creating Extract records, to only create records for objects with the given status. |
| /CLASS | This parameter tells the utility to set the class when creating Add records or, when creating Extract records, to only create records for objects with the given class. |
| /PROJECT | This parameter tells the utility to set the project when creating Add records or, when creating Extract records, to only create records for objects with the given project. |
| /LN | The LN (*Long Name*) parameter lets you specify whether the utility applies standard file names or long file names to the library objects it extracts. The default is Yes.<br>A long (or versioned) file name contains the version, revision and effective date in the name. For example, version 1, revision 2 of the Q1ADDR image (FAP), with an effective date of September 1, 2005, would have a long file name of...<br>`Q1ADDR_0000100002_20050901.FAP`<br>When extracting various versions and/or revisions of a library object, it is necessary to extract the objects to long file names to prevent one version/revision of the object from replacing another version/revision of an object with the same name. |
| /EXP | Set this parameter to No to prevent the utility from creating Add or Extract records for expired objects. The default is Yes. |
| /CVTOLD | Use this parameter to convert library files created before version 10.2 into the newer format.<br>Keep in mind you should *always* back up your files *before* you convert a library. |

## Creating Response Files

You tell the utility to create a response file by specifying certain parameters. Generally, if you include the FORMDEF parameter or the BDF, GRP, FOR, FAP, DDT, LOGO, DAL, or ALL parameters, the utility creates a response file. If you omit these parameters, the utility instead processes a response file.

You indicate which type of response file records you want the utility to create using the /REC parameter. The default is to generate Add records in the response file.

## Generating Add Records

You can use the utility to add resources, such as FAP, DDT, LOG, DAL, BDF, GRP, and FOR files, into a library. When you generate Add records to the response file, you can tell the utility to read the contents of your FORM.DAT file and generate Add records for the resources referenced in it. You can also specify paths you want the utility to use to search for files to generate Add records from.

### Example 1 - Using the /FORMDEF parameter

In this example, the command tells the LBRYMGR utility to create a response file named *RESPONSE.RSP* and to read the FORM.DAT file located in the indicated DefLib directory. The utility then generates Add records and writes them into the response file for all referenced resource files.

**NOTE:** Unless you specifically indicate otherwise, the system uses the current date as the effective date for each Add record.

```
C:\fap\dll> LBRYMGR /rsp=response.rsp
/formdef=c:\fap\mstrres\rpex1\deflib\form.dat /fap /ddt /logo

   ---      DocuCorp LBRYMGR Utility Program (C)      ---
   ---      Library Manager Response File Utility     ---

   Attempting to create Response File <response.rsp>
   Effective Date for objects will be:    October 24 00:00 2007
   Successfully created response file <response.rsp>
```

### Example 2 - Using the /FAP, /DDT, and /LOGO parameters

Instead of using the /FORMDEF parameter to indicate which resources to create Add records for, you can also point to specific directories and the LBRYMGR utility will generate Add records for all files in those directories.

This example tells the LBRYMGR utility to create a response file named RESPONSE.RSP. It generates Add records for the files located in the forms directory that have a *.FAP* extension. It also generates Add records for the DDT and LOG files stored in the specified directories.

```
C:\fap\dll> LBRYMGR /rsp=response.rsp
/fap=c:\fap\mstrres\rpex1\forms\*.fap
/ddt=c:\fap\mstrres\rpex1\deflib\*.ddt
/logo=c:\fap\mstrres\rpex1\forms\*.log

   ---      DocuCorp LBRYMGR Utility Program (C)      ---
   ---      Library Manager Response File Utility     ---

   Attempting to create Response File <response.rsp>
   Successfully created response file <response.rsp>
```

## Generating Extract Records

You can also create a response file that contains Extract records. You can then process this response file. The utility will extract the indicated resources from the library and copy them to the location you specified. To generate Extract records, use the /REC=X parameter.

### Example 1 - Creating extract records for all resources

This example tells the LBRYMGR utility to create a response file named RESPONSE.RSP. It generates Extract records for all resources in the library. The response file contains the target path and file name for each resource that is extracted. The /ALL parameter tells the system to extract all resource types and to create target file names using the indicated directory.

```
C:\fap\dll>lbrymgrw /rsp=response.rsp /rec=x
/lby=.\deflib\master.lby /all=.\ext\

---      DocuCorp LBRYMGR Utility Program (C)      ---
---      Library Manager Response File Utility     ---

Successfully created response file <response.rsp>
```

### Example 2 - Creating extract records for the latest version and revision

This example tells the LBRYMGR utility to create a response file named RESPONSE.RSP. It generates Extract records for the latest revision of the latest version of each FAP file in the library. The file name for each resource placed in the response file will be the short name instead of the long or versioned name, which is the default.

```
C:\fap\dll>lbrymgrw /rsp=response.rsp /rec=x
/lby=.\deflib\master.lby /fap=.\ext\ /r=l /ln=no /v=l

---      DocuCorp LBRYMGR Utility Program (C)      ---
---      Library Manager Response File Utility     ---

Successfully created response file <response.rsp>
```

## Generating Sync Records

You use Sync records to update or promote resources. For instance, you could generate Sync records to promote resources from a development library into a testing library. Include the /REC=S parameter to create a response file that contains Sync records.

**NOTE:** To promote resources from one library to another or to synchronize libraries, use the LBYSYNC utility.

### Example - Creating sync records

This example tells the LBRYMGR utility to create a response file named RESPONSE.RSP. It generates Sync records for all resources in the library. Part of the process of generating the response file involves extracting the resources from the library, assigning them a temporary name, and writing these files to the directory indicated with the /SP parameter. When you later process the response file, these resources are copied into the target library, provided they are newer than the existing resources in the target library.

```
C:\fap\dll>lbrymgrw /rsp=response.rsp /rec=s
/lby=.\deflib\master.lby /sp=.\sync\

---      DocuCorp LBRYMGR Utility Program (C)      ---
---      Library Manager Response File Utility     ---

Will create Sync records for all Revisions of all Versions.
Attempting to create Sync Response File <response.rsp>
Successfully created response file <response.rsp>
```

## Response File Format

Generally you create a response file by running the LBRYMGR utility. Though it is best to let the LBRYMGR utility create the response file, there may be situations in which you need to later edit the resulting response file.

Response files contain one or more records. Each record contains the information necessary information for an action (Add, Extract, Sync) to be performed on a single resource. Each record consists of several semi-colon delimited fields. The fields are ordered as shown here:

```
;Action;FileType;FileSubType;FullName;Name;Resource;Description;Eff
ectiveDate;UserLevel;Password;UserID;Version;Revision;ModifyTime;Mo
de;Status;Class;Project;
```

| Field | Description |
|---|---|
| Action | Specifies what action to perform when this response record is processed. You can enter<br>- A (adds a file)<br>- D (deletes a file)<br>- R (replaces the reference by deleting old reference and adding a new one)<br>- U (updates record data but does not update the file<br>- X (extracts file)<br>- S (synchronizes or promotes the resource to another library). |
| FileType | Indicates the type of resource that is being processed. You can enter<br>- FAP (image)<br>- DDT (data definition table)<br>- LOG (Logo)<br>- DAL (Document Automation Language)<br>- BDF (business unit definition)<br>- GRP (business unit group)<br>- FOR (form). |
| FileSubType | Indicates the subtype of the resource that is being processed. You can enter<br>- FAP (image)<br>- DDT (data definition table)<br>- LOG (logo)<br>- DAL (Document Automation Language)<br>- BDF (business unit definition)<br>- GRP (business unit group)<br>- FOR (form)<br>Use the same value here as is used for the FileType field. |
| FullName | Indicates the fully or partially qualified name of the resource being processed. |

| Field | Description |
|---|---|
| Name | Specifies the name of the resource.<br>When processing an Add record, the name field indicates the value the name field in the library index will be set to as the resource is added to the library.<br>When processing an Extract record, the name field indicates the name of the resource to extract from the library. |
| Resource | This is a legacy field used to indicate the general location the resource came from, such as FORMS or DEFLIB. This field is no longer used. |
| Description | (optional) You can use this field to include a comment that will be placed in the library index Description field. You can enter up to 100 characters. |
| Effectivedate | (optional) You can use this field to specify a date at which the resource will become effective. Enter the date in the YYYYMMDD format. If you omit this date, the system uses the date on which the response file is created. |
| UserLevel | (optional) You can use this field to specify a two-digit key for access rights. The default is 99, which allows access to all users. |
| Password | (optional) You can use this field to associate a password with the resource. |
| UserID | (optional) You can use this field to specify the user ID you want to associate with the resource in the library index. You can enter up to 64 characters. The default is *docucorp*. |
| Version | (optional) When processing an Add record, this field specifies the value the Version field in the library index will be set to as the resource is added to the library.<br>When processing an Extract record, this field specifies the version of the resource to extract.<br>The Version field should consist of five digits padded with zeros, such as 00001 or 00015. The default is 00001. |
| Revision | (optional) When processing an Add record, this field specifies the value the Revision field in the library index will be set to as that resource is added to the library.<br>When processing an Extract record, this field specifies the revision of the resource to extract from the library.<br>The Revision field should consist of five digits padded with zeros, such as 00001 or 00015. The default is 00001. |
| ModifyTime | (optional) When processing an Add or Sync record, this field specifies the value that the ModifyTime field in the library index will be set to as that resource is added to the library. If omitted, the ModifyTime field in the library index is set to the time at which the resource is added to the library. This value is in a hexadecimal format and should generally not be manually edited. |
| Mode | (optional) When processing an Add record, you can use this field to specify the value the Mode field in the library index will be set to as the resource is added to the library. If omitted, the Mode field in the library index is set to blank. |

| Field | Description |
|---|---|
| Status | (optional) When processing an Add record, you can use this field to specify the value the Status field in the library index will be set to as the resource is added to the library. If omitted, the Status field in the library index is set to blank. |
| Class | (optional) When processing an Add record, you can use this field to specify the value the Class field in the library index will be set to as the resource is added to the library. If omitted, the Class field in the library index is set to blank. |
| Project | (optional) When processing an Add record, you can use this field to specify the value the Project field in the library index will be set to as the resource is added to the library. If omitted, the Project field in the library index is set to blank. |

Here is an example of an Add record:

```
;A;FAP;FAP;.\forms\q1snam.fap;q1snam;FORMS;sample
description;20040603;99;;DOCUCORP;00001;00001;;DEV;TEST;GA;P001;
```

## Processing Response Files

You process a response file by specifying a combination of parameters. Generally, you only need to specify a response file name and a library name. The utility then processes the response file and, depending on the contents of the response file, adds to, extracts from, or synchronizes resources in the library.

### Processing Add Records

If you have created a response file that contains Add records, you can run this response file against a library. The utility then adds the resources listed in the response file to the library.

Example - Adding resources to the library

In this example, the LBRYMGR utility reads the response file and processes the records against the library called *master.lby*. They are all add records. Including the /C parameter tells the utility to display a counter to show its progress.

```
C:\fap\dll> LBRYMGR /rsp=response.rsp /
lby=c:\fap\mstrres\rpex1\deflib\master.lby /c

   ---      DocuCorp LBRYMGR Utility Program (C)      ---
   ---      Library Manager Response File Utility     ---

   Attempting to read and process Response File <response.rsp>
   Successfully processed response file <response.rsp>

   Add (A) Record Statistics:
   Records Read                 :     125
   Objects updated index        :       0
   Objects added, Ver/Rev ok    :       0
   Objects added as new         :     125
   Objects added Ver/Rev incr   :       0
   Objects Ver/Rev error        :       0
   Objects add error            :       0
```

## Processing Extract Records

If you have created a response file that contains Extract records, you can run this response file against a library and the resources will be extracted from the library and saved to the location you specified in the response file.

### Example - Extracting resources from the library

In this example, the LBRYMGR utility reads the response file and processes the records against the library named MASTER.LBY. They are all extract records. Including the /C parameter tells the utility to display a counter to show its progress.

```
C:\fap\dll>lbrymgrw /rsp=response.rsp /lby=.\deflib\master.lby /c

---      DocuCorp LBRYMGR Utility Program (C)      ---
---      Library Manager Response File Utility     ---

Attempting to read and process Response File <response.rsp>
Successfully processed response file <response.rsp>

Extract (X) Record Statistics:
Records Read                 :     142
Objects not found in Library :       0
Objects extracted            :     142
Objects extract error        :       0
```

## Processing Sync Records

If you have created a response file that contains Sync records, you can run this response file against a target library and the resources listed in the response file will be copied into the target library, provided the last modified date and time for the resource in the response file is newer than the last modified date and time of the resource in the target library.

### Example - Synchronizing libraries

In this example, the LBRYMGR utility reads the response file and processes the Sync, records, against the library named MASTER.LBY. The /C parameter tells the utility to display a counter to show its progress.

```
C:\fap\dll>lbrymgrw /rsp=response.rsp /lby=.\deflib\prod.lby /c

---      DocuCorp LBRYMGR Utility Program (C)      ---
---      Library Manager Response File Utility     ---

Attempting to read and process Response File <response.rsp>
Successfully processed response file <response.rsp>

Sync (S) Record Statistics:
Records Read                 :     142
Objects found in Library     :       0
Objects not found in Library :     142
Objects older or same        :       0
Objects updated index        :       0
Objects added, Ver/Rev ok    :      18
Objects added as new         :     124
Objects added Ver/Rev incr   :       0
Objects Ver/Rev error        :       0
Objects add error            :       0
```

## Converting Libraries

If a library is in an older, pre-version 10.2 format, use the LBRYMGR utility to convert the library to the newer format. The format of the library changed in version 10.2 to increase the lengths of some fields and to add these fields, used for project management:

- Mode
- Status
- Class
- Project

### Example - Converting a library

In this example, you use the /CVTOLD parameter to tell the LBRYMGR utility to convert the library named *MASTER.LBY* from an older format into the newer format. Before converting the library, the utility backs up the library using file names prefixed with a dollar sign ($). For instance, for the MASTER.LBY library, this table shows the back up files:

| This file | Is backed up to |
|---|---|
| MASTER.DBF | $MASTER.DBF |
| MASTER.MDX | $MASTER.MDX |
| MASTER.LBY | $MASTER.LBY |

Once the library is backed up, the utility converts it to the newer format.

```
C:\fap\dll> LBRYMGR /lby=c:\fap\mstrres\rpex1\deflib\master.lby /
cvtold

   ---      DocuCorp LBRYMGR Utility Program (C)      ---
   ---      Library Manager Response File Utility     ---

Library <c:\fap\mstrres\rpex1\deflib\master.lby> was successfully
backed up.
Library <c:\fap\mstrres\rpex1\deflib\master.lby> was successfully
converted.
```

### Syntax

# LBYPROC

Use the LBYPROC utility to process library scripts. Library scripts are XML-based files that let you perform actions on a resource library. You can use these scripts to...

- Add resources to a library
- Delete resources from a library
- Update a resource's library index record
- Extract resources from a library (writing the contents to a disk file)
- Promote resources from one library to another
- Roll back resources that have been promoted
- Search a library for specific elements
- Produce a list of resources that match a designated set of filter values

The scripts are designed to perform the indicated action on multiple resources. For example, a promote script can promote many resources from one library to another and an extract script can extract many resources from the library and write them to disk.

## Program names

| | |
|---|---|
| Windows | lbyproc.exe |
| UNIX/Linux | lbyproc |
| z/OS (MVS) | LBYPROC |

## Syntax

**LBYPROC /I= [/INI=] [/TEST] [/T] [/L] [/NOLOG]**

| Parameter | Description |
|---|---|
| /I | Specifies the name of the file that contains the library scripts you want the utility to process. |
| /INI | Specifies the name and path of the INI file |
| /TEST | Include this parameter to run the utility without actually updating anything. The utility will report what actions it would perform if it were actually run. |
| /T | Turns on tracing, which may be useful during problem analysis. |
| /L | Causes the INI options to be written to the trace file, which may be useful during problem analysis. |
| /NOLOG | Suppresses logging of the library actions to the library log file. |

## Return codes

This utility returns these codes upon execution:

| Code | Description |
|---|---|
| 0 | SUCCESS |
| 4 | WARNING |
| 8 | ERROR |

## Examples

Here are some examples:

### Example 1

Promoting all resources from a development library into a test library.

In this example you run the LBYPROC utility with a Promote script that specifies the source library name (MASTER.LBY) and the target library name (TEST.LBY). In this example, both libraries are in xBase format.

If you run the utility with these parameters:

```
c:\fap\dll\rpex1dm>lbyproc /i=deflib\pro1.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2006 Docucorp International
--- Documaker library script processor


Found <1> Library Scripts
PROMOTE Successful. Name<RPEX1DM> Type<BDF> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<ADDCOM> Type<DAL> Ver<00001> Rev<00001>
Note<Normal promotion>
 . . .
--- LBYPROC Complete ---
```

Here are the contents of the *pro1.lsc* file, which contains the Promote script:

```
<LBYSCRIPT>
<PROMOTE>
<LIBRARY SRC="DEFLIB\MASTER.LBY" TGT="DEFLIB\TEST.LBY"/>
<NAME SRC=""/>
<TYPE SRC=""/>
<VER SRC=""/>
<REV SRC=""/>
<USERID SRC=""/>
<EFFDATE SRC="" />
<TEMPNAME SRC="" />
<MODE SRC="" TGT=""/>
<STATUS SRC="" TGT="*"/>
<CLASS SRC="" TGT="*"/>
<PROJECT SRC="" TGT="*"/>
</PROMOTE>
</LBYSCRIPT>
```

Example 2 Using the /TEST parameter to preview what the Promote script will do.

In this example the Promote script promotes all resources that have a status of *Passed* from the development library (MASTER.LBY) into the test library (TEST.LBY).

You first run the LBYPROC utility using the /TEST parameter so you can see a preview of what will be promoted. After running it with the /TEST parameter, remove this parameter and run the utility again to actually promote resources.

Resources in the source library (MASTER.LBY) that have a status of *Passed* are promoted to the target library (TEST.LBY). In the target library, these newly promoted resources are assigned a status of *Test*. Upon successful promotion, the resources in the source library that were promoted are assigned a status of *Promoted*.

If you run the utility with these parameters:

```
c:\fap\dll\rpex1dm>lbyproc /i=deflib\pro2.lsc /test
```

You will get this output from the utility when running in Test mode:

```
--- LBYPROC Copyright (C) 1997-2006 Docucorp International
--- Documaker library script processor

Found <1> Library Scripts
(Preview)PROMOTE Successful. Name<SETRCPTB> Type<DAL> Ver<00001>
Rev<00001> Note<Normal promotion>
(Preview)PROMOTE Successful. Name<Q1ADDR> Type<FAP> Ver<00001>
Rev<00001> Note<Normal promotion>
(Preview)PROMOTE Successful. Name<Q1AFLG> Type<FAP> Ver<00001>
Rev<00001> Note<Normal promotion>
(Preview)PROMOTE Successful. Name<Q1B302> Type<FAP> Ver<00001>
Rev<00001> Note<Normal promotion>
(Preview)PROMOTE Successful. Name<CGDEC> Type<FOR> Ver<00001>
Rev<00001> Note<Normal promotion>
(Preview)PROMOTE Successful. Name<Q1DLOG> Type<LOG> Ver<00001>
Rev<00001> Note<Normal promotion>

--- LBYPROC Complete ---
```

Use a command similar to this one to actually promote the resources:

```
c:\fap\dll\rpex1dm>lbyproc /i=deflib\pro2.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2006 Docucorp International
--- Documaker library script processor

Found <1> Library Scripts
PROMOTE Successful. Name<SETRCPTB> Type<DAL> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1ADDR> Type<FAP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1AFLG> Type<FAP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1B302> Type<FAP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<CGDEC> Type<FOR> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1DLOG> Type<LOG> Ver<00001> Rev<00001>
Note<Normal promotion>

--- LBYPROC Complete ---
```

Here are the contents of the *pro2.lsc* file, which contains the Promote script:

```
<LBYSCRIPT>
<PROMOTE>
<LIBRARY SRC="DEFLIB\MASTER.LBY" TGT="DEFLIB\TEST.LBY"/>
<NAME SRC=""/>
<TYPE SRC=""/>
<VER SRC=""/>
<REV SRC=""/>
<USERID SRC=""/>
<EFFDATE SRC="" />
<TEMPNAME SRC="" />
<MODE SRC="" TGT=""/>
<STATUS SRC="PASSED" TGT="TEST" FINAL="PROMOTED"/>
<CLASS SRC="" TGT="*"/>
<PROJECT SRC="" TGT="*"/>
</PROMOTE>
</LBYSCRIPT>
```

### Example 3

Promoting resources from a development library (in SQL Server) to a test library (in SQL Server).

In this example you run the LBYPROC utility with a Promote script that specifies the source library name (LBYDEV) and the target library name (LBYTEST). Each library is stored in an SQL Server database.

Resources in the source library (LBYDEV) that have a status of *Passed* are promoted to the target library (LBYTEST). In the target library, these newly promoted resources are assigned a status of *Test*. Upon successful promotion, the resources in the source library that were promoted are assigned a status of *Promoted*.

If you run the utility with these parameters:

```
c:\fap\dll\rpex1dm>lbyproc /i=deflib\pro3.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2006 Docucorp International
--- Documaker library script processor

Found <1> Library Scripts

PROMOTE Successful. Name<Q1ADDR> Type<FAP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1AFLG> Type<FAP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1B302> Type<FAP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1BA32> Type<FAP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<SUPPLEMENT> Type<FOR> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<FSI_CPP> Type<GRP> Ver<00001> Rev<00001>
Note<Normal promotion>
PROMOTE Successful. Name<Q1DLOG> Type<LOG> Ver<00001> Rev<00001>
Note<Normal promotion>

--- LBYPROC Complete ---
```

Here are the contents of the *pro3.lsc* file, which contains the Promote script:

```
<LBYSCRIPT>
<PROMOTE>
<LIBRARY SRC="LBYDEV" TGT="LBYTEST"/>
<NAME SRC=""/>
<TYPE SRC=""/>
<VER SRC=""/>
<REV SRC=""/>
<USERID SRC=""/>
<EFFDATE SRC=""/>
<MODE SRC="" TGT="" FINAL=""/>
<STATUS SRC="PASSED" TGT="TEST" FINAL="PROMOTED"/>
<CLASS SRC="" TGT="" FINAL=""/>
<PROJECT SRC="" TGT="" FINAL=""/>
</PROMOTE>
</LBYSCRIPT>
```

Here are the INI options that relate to the ODBC database handlers and library tables used for this example:

```
< DBHandler:LBYSETUP >
    Class              = ODBC
    CreateTable        = Yes
    Debug              = No
    Description        = Original ODBC handler in SQL Server
    Passwd             = pw
    Server             = SQLDEV    (ODBC Data Source Name)
    UserID             = userid
< DBHandler:LBYTEST >
    Class              = ODBC
    CreateTable        = Yes
    Debug              = No
    Description        = ODBC handler for Test database in SQL Server
    Passwd             = pw
    Server             = SQLTEST   (ODBC Data Source Name)
    UserID             = userid
< DBTable:CATALOG >
    DBHandler          = LBYSETUP
< DBTable:LBYDEV >
    DBHandler          = LBYSETUP
< DBTable:LBYDEVD >
    DBHandler          = LBYSETUP
< DBTable:LBYTEST >
    DBHandler          = LBYTEST
< DBTable:LBYTESTD >
    DBHandler          = LBYTEST
< Library:LBYDEV >
    DBTable            = LBYDEVD
    Description        = Development library in SQL Server
< Library:LBYTEST >
    DBTable            = LBYTESTD
    Description        = Test library in SQL Server
< ODBC_FileConvert >
    CATALOG            = dbo.DMKR_CATALOG
    LBYDEV             = dbo.DMKR_LBYDEV
    LBYDEVD            = dbo.DMKR_LBYDEVD
```

```
LBYTEST           = dbo.DMKR_LBYTEST
LBYTESTD          = dbo.DMKR_LBYTESTD
```

Example 4 Filtering the development library for all FAP files that contain a Text element with the value *POLICY*. (On Windows)

In this example you run the LBYPROC utility with a Filter script that indicates what to search for. You specify the source library name (MASTER.LBY) and the target library name (TEST.LBY). In this example, both libraries are in xBase format.

If you run the utility with these parameters:

```
c:\fap\dll\rpex1dm>lbyproc /i=deflib\filter1.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2006 Docucorp International
--- Documaker library script processor

Found <1> Library Scripts

Search parameters follow:

Library         : DEFLIB\MASTER.LBY
Type            : FAP
Object Type     : TEXT
Object Name     :
Object Text     : POLICY
Object TextCase :

Found match: <Q1AFLG> <FAP> <00001> <00001> <Initial check in>
Found match: <Q1BA32> <FAP> <00001> <00001> <Initial check in>
Found match: <Q1BA36> <FAP> <00001> <00001> <Initial check in>
. . .
SEARCH Successful. Found <23> matching resources.

--- LBYPROC Complete ---
```

Here are the contents of the *filter1.lsc* file, which contains the Filter script:

```
<LBYSCRIPT>
<FILTER>
<LIBRARY VALUE="DEFLIB\MASTER.LBY"/>
<NAME VALUE=""/>
<DESC VALUE=""/>
<TYPE VALUE="FAP"/>
<VER VALUE=""/>
<REV VALUE=""/>
<USERID VALUE=""/>
<EFFDATE VALUE=""/>
<LOCKED VALUE=""/>
<MODE VALUE=""/>
<STATUS VALUE=""/>
<CLASS VALUE=""/>
<PROJECT VALUE=""/>
<OBJECTTYPE VALUE="TEXT"/>
<OBJECTNAME VALUE=""/>
<OBJECTTEXT VALUE="POLICY"/>
<OBJECTTEXTCASE VALUE=""/>
</FILTER>
</LBYSCRIPT>
```

Example 5 Extracting all resources from the development library and writing them to disk. (On Windows)

In this example you run the LBYPROC utility with an Extract script that tells the utility to extract all resources from the library (MASTER.LBY) and write the resources into the ".\EXT\" directory. Specify the target directory using the XML tag *ALLLIB*.

Set the XML tag *LongFileName* to Yes to tell the utility to write the resources to disk using the versioned or long name. The long name of the resource consists of the resource name followed by an underscore, followed by the version and revision of the resource followed by an underscore, followed by the effective date of the resource, followed by a period and the resource type. For example, the long name of version 1 revision 3 of the RPEX1DM BDF resource is called:

```
RPEX1DM_0000100003_19800101.bdf
```

Setting the LongFileName option to Yes allows multiple versions/revisions of the same resource to be written to disk and identified uniquely.

If you run the utility with these parameters:

```
c:\fap\dll\rpex1dm>lbyproc /i=deflib\ext1.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2006 Docucorp International
--- Documaker library script processor

Found <1> Library Scripts

EXTRACT Successful. Filename<.\ext\RPEX1DM_0000100001_19800101.bdf>
Name<RPEX1DM> Type<BDF> Ver<00001> Rev<00001> Note<File did not exist
yet>
EXTRACT Successful. Filename<.\ext\RPEX1DM_0000100002_19800101.bdf>
Name<RPEX1DM> Type<BDF> Ver<00001> Rev<00002> Note<File did not exist
yet>
EXTRACT Successful. Filename<.\ext\RPEX1DM_0000100003_19800101.bdf>
Name<RPEX1DM> Type<BDF> Ver<00001> Rev<00003> Note<File did not exist
yet>
EXTRACT Successful. Filename<.\ext\RPEX1DM_0000200001_20060901.bdf>
Name<RPEX1DM> Type<BDF> Ver<00002> Rev<00001> Note<File did not exist
yet>
EXTRACT Successful. Filename<.\ext\ADDCOM_0000100001_19800101.dal>
Name<ADDCOM> Type<DAL> Ver<00001> Rev<00001> Note<File did not exist
yet>
EXTRACT Successful. Filename<.\ext\ADDCOM_0000100002_19800101.dal>
Name<ADDCOM> Type<DAL> Ver<00001> Rev<00002> Note<File did not exist
yet>
. . .
EXTRACT Successful. Filename<.\ext\SYMBOL_0000100001_19800101.XDD>
Name<SYMBOL> Type<XDD> Ver<00001> Rev<00001> Note<File did not exist
yet>

--- LBYPROC Complete ---
```

Here are the contents of the *ext1.lsc* file, which contains the Extract script:

```
<LBYSCRIPT>
<EXTRACT>
<LIBRARY VALUE="DEFLIB\MASTER.LBY"/>
<FILENAME VALUE=""/>
<NAME VALUE=""/>
<TYPE VALUE=""/>
<DESC VALUE=""/>
<VER VALUE=""/>
<REV VALUE=""/>
<USERID VALUE=""/>
<EFFDATE VALUE=""/>
<MODE VALUE=""/>
<STATUS VALUE=""/>
<CLASS VALUE=""/>
<PROJECT VALUE=""/>
<ALLLIB VALUE=".\ext\"/>
<BDFLIB VALUE=""/>
<GRPLIB VALUE=""/>
<FORLIB VALUE=""/>
<FAPLIB VALUE=""/>
<DDTLIB VALUE=""/>
<LOGLIB VALUE=""/>
<DALLIB VALUE=""/>
<DEFLIB VALUE=""/>
<LONGFILENAME VALUE="Yes"/>
</EXTRACT>
</LBYSCRIPT>
```

Example 6 Extracting the last version and revision of all FAP resources from the development library and writing them to disk. (On Windows)

In this example you run the LBYPROC utility with an extract script that tells the utility to extract resources of type *FAP* from the library (MASTER.LBY) and write the resources to the ".\EXT\" directory. Specify the target directory using the XML tag *ALLLIB*.

Set the XML tag of *VER* to *(last)* to indicate that you only want to extract the latest version of each resource to disk. Also set the XML tag of *REV* to *(last)* to indicate you only want to extract the latest revision of each resource. This combination of VER=(last) and REV=(last) means that, for any given resource, the utility will only extract the latest revision of the latest version of that resource.

Since you are asking for only the latest version and revision of each FAP resource, set the LongFileName option to No. So instead of the resource Q1ADDR.FAP being written to disk with a long name of *Q1ADDR_0000100001_19800101.fap*, it is simply written to disk with a name of *Q1ADDR.FAP*.

If you run the utility with these parameters:

```
c:\fap\dll\rpex1dm>lbyproc /i=deflib\ext2.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2006 Docucorp International
--- Documaker library script processor

Found <1> Library Scripts

EXTRACT Successful. Filename<.\ext\Q1ADDR.fap> Name<Q1ADDR>
Type<FAP> Ver<00001> Rev<00001> Note<File did not exist yet>
EXTRACT Successful. Filename<.\ext\Q1AFLG.fap> Name<Q1AFLG>
Type<FAP> Ver<00001> Rev<00001> Note<File did not exist yet>
EXTRACT Successful. Filename<.\ext\Q1B302.fap> Name<Q1B302>
Type<FAP> Ver<00001> Rev<00001> Note<File did not exist yet>
EXTRACT Successful. Filename<.\ext\Q1BA32.fap> Name<Q1BA32>
Type<FAP> Ver<00001> Rev<00001> Note<File did not exist yet>
. . .
EXTRACT Successful. Filename<.\ext\Q1VRFL.fap> Name<Q1VRFL>
Type<FAP> Ver<00001> Rev<00001> Note<File did not exist yet>

--- LBYPROC Complete ---
```

Here are the contents of the *ext2.lsc* file, which contains the EXTRACT script:

```
<LBYSCRIPT>
<EXTRACT>
<LIBRARY VALUE="DEFLIB\MASTER.LBY"/>
<FILENAME VALUE=""/>
<NAME VALUE=""/>
<TYPE VALUE="FAP"/>
<DESC VALUE=""/>
<VER VALUE="(last)"/>
<REV VALUE="(last)"/>
<USERID VALUE=""/>
<EFFDATE VALUE=""/>
<MODE VALUE=""/>
<STATUS VALUE=""/>
<CLASS VALUE=""/>
<PROJECT VALUE=""/>
```

```
<ALLLIB VALUE=".\ext\"/>
<BDFLIB VALUE=""/>
<GRPLIB VALUE=""/>
<FORLIB VALUE=""/>
<FAPLIB VALUE=""/>
<DDTLIB VALUE=""/>
<LOGLIB VALUE=""/>
<DALLIB VALUE=""/>
<DEFLIB VALUE=""/>
<LONGFILENAME VALUE="No"/>
</EXTRACT>
</LBYSCRIPT>
```

### Example 7 Adding a resource to a development library

In this example, an ADD script adds a FAP file stored on the disk into a development library. The FAP file is assigned a name of Q1ADDR, a type of FAP, a version of 00001, and a revision of 00001.

If you run the utility with these parameters:

```
C:\fap\dll\rpex1dm>lbyproc /i=deflib\add1.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2008 Skywire Software
--- Documaker library script processor

ADD Successful.  File<.\forms\Q1ADDR.fap>

Add performed.  The following number of objects were
added to the library.

LIBRARY:    DEFLIB\MASTER.LBY

BDFs :      0
GRPs :      0
FORs :      0
FAPs :      1
DDTs :      0
LOGs :      0
DALs :      0
XDDs :      0
-------------
Total:      1


--- LBYPROC Complete ---
```

Here are the contents of the *add1.lsc* file, which contains the ADD script:

```
<LBYSCRIPT>
     <ADD>
     <LIBRARY VALUE="DEFLIB\MASTER.LBY"/>
     <FILENAME VALUE=".\forms\Q1ADDR.fap"/>
     <NAME VALUE="Q1ADDR"/>
     <TYPE VALUE="FAP"/>
     <VER VALUE="00001"/>
     <REV VALUE="00001"/>
     <MODE VALUE=""/>
     <STATUS VALUE=""/>
     <CLASS VALUE=""/>
     <PROJECT VALUE=""/>
     </ADD>
</LBYSCRIPT>
```

Example 8 Updating resources in a development library.

In this example, an UPDATE script updates all FAP resources in the development library (master.lby) that have a status of *Test* and changes that status to *Failed*.

If you run the utility with these parameters:

```
C:\fap\dll\rpex1dm>lbyproc /i=deflib\upd1.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2008 Skywire Software
--- Documaker library script processor

UPDATE Successful. Name<Q1ADDR> Type<FAP> Ver<00001> Rev<00001>
Note<Resource updated>
UPDATE Successful. Name<Q1AFLG> Type<FAP> Ver<00001> Rev<00001>
Note<Resource updated>
UPDATE Successful. Name<Q1B302> Type<FAP> Ver<00001> Rev<00001>
Note<Resource updated>
. . .
UPDATE Successful. Name<Q1VRFL> Type<FAP> Ver<00001> Rev<00001>
Note<Resource updated>

Update performed.  The following number of objects were
updated in the library.

LIBRARY:    DEFLIB\MASTER.LBY

BDFs :      0
GRPs :      0
FORs :      0
FAPs :     81
DDTs :      0
LOGs :      0
DALs :      0
XDDs :      0
-------------
Total:     81


--- LBYPROC Complete ---
```

Here are the contents of the *upd1.lsc* file, which contains the UPDATE script:

```
<LBYSCRIPT>
     <UPDATE>
     <LIBRARY VALUE="DEFLIB\MASTER.LBY"/>
     <NAME VALUE=""/>
     <TYPE VALUE=""/>
     <VER VALUE=""/>
     <REV VALUE=""/>
     <MODE VALUE=""/>
     <STATUS VALUE="TEST" NEWVALUE="FAILED"/>
     <CLASS VALUE=""/>
     <PROJECT VALUE=""/>
     </UPDATE>
</LBYSCRIPT>
```

Example 9 Deleting resources from a development library.

In this example, a DELETE script removes all FAP resources in the development library (master.lby) with a status of *Failed*.

If you run the utility with these parameters:

```
C:\fap\dll\rpex1dm>lbyproc /i=deflib\del1.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2008 Skywire Software
--- Documaker library script processor

DELETE Successful. Name<Q1ADDR> Type<FAP> Ver<00001> Rev<00001>
Note<Normal Deletion>
DELETE Successful. Name<Q1AFLG> Type<FAP> Ver<00001> Rev<00001>
Note<Normal Deletion>
DELETE Successful. Name<Q1B302> Type<FAP> Ver<00001> Rev<00001>
Note<Normal Deletion>
. . .
DELETE Successful. Name<Q1TILE> Type<FAP> Ver<00001> Rev<00001>
Note<Normal Deletion>
DELETE Successful. Name<Q1VRFL> Type<FAP> Ver<00001> Rev<00001>
Note<Normal Deletion>

Delete performed.  The following number of objects were
deleted from the library.

LIBRARY:    DEFLIB\MASTER.LBY

BDFs :      0
GRPs :      0
FORs :      0
FAPs :     81
DDTs :      0
LOGs :      0
DALs :      0
XDDs :      0
-------------
Total:     81

--- LBYPROC Complete ---
```

Here are the contents of the *del1.lsc* file, which contains the DELETE script:

```
<LBYSCRIPT>
     <DELETE>
     <LIBRARY VALUE="DEFLIB\MASTER.LBY"/>
     <NAME VALUE=""/>
     <TYPE VALUE=""/>
     <VER VALUE=""/>
     <REV VALUE=""/>
     <MODE VALUE=""/>
     <STATUS VALUE="FAILED"/>
     <CLASS VALUE=""/>
     <PROJECT VALUE=""/>
     </DELETE>
</LBYSCRIPT>
```

Example 10

Rolling back resources from a test library (in SQL Server) to a development library (in SQL Server).

This example reverses the PROMOTE performed in Example 3. The LBYPROC utility is run with a single ROLLBACK script. The ROLLBACK script specifies the source library name (LBYTEST) and the target library name (LBYDEV). Each library is stored in an SQL server database. Note that the source library is the library you are rolling back *from* and the target library is the library you are rolling back *to*.

**NOTE:** A ROLLBACK does not copy any resources from the source library to the target library. A ROLLBACK removes a resource from the source library and, optionally, updates the Mode, Status, Class, and/or Project fields of that resource in the target library to a value you assign, like *ROLLEDBACK*, to indicate that it has been rolled back from a *higher* library.

In this example, resources in the source library (LBYTEST) that have a status of *TEST* are removed from this library. In the target library (LBYDEV), these resources are then assigned a status of *ROLLEDBACK*.

If you run the utility with these parameters:

```
C:\fap\dll\rpex1dm>lbyproc /i=deflib\roll3.lsc
```

You will get this output from the utility:

```
--- LBYPROC Copyright (C) 1997-2007 Docucorp International
--- Documaker library script processor

ROLLBACK Successful. Name<Q1ADDR> Type<FAP> Ver<00001> Rev<00001>
Note<Normal rollback>
ROLLBACK Successful. Name<Q1AFLG> Type<FAP> Ver<00001> Rev<00001>
Note<Normal rollback>
ROLLBACK Successful. Name<Q1B302> Type<FAP> Ver<00001> Rev<00001>
Note<Normal rollback>
ROLLBACK Successful. Name<Q1BA32> Type<FAP> Ver<00001> Rev<00001>
Note<Normal rollback>
ROLLBACK Successful. Name<SUPPLEMENT> Type<FOR> Ver<00001>
Rev<00001> Note<Normal rollback>
ROLLBACK Successful. Name<FSI_CPP> Type<GRP> Ver<00001> Rev<00001>
Note<Normal rollback>
ROLLBACK Successful. Name<Q1DLOG> Type<LOG> Ver<00001> Rev<00001>
Note<Normal rollback>

--- LBYPROC Complete ---
```

Here are the contents of the *pro3.lsc* file, which contains the ROLLBACK script:

```
<LBYSCRIPT>
     <ROLLBACK>
     <LIBRARY SRC="LBYTEST" TGT="LBYDEV"/>
     <NAME SRC=""/>
     <TYPE SRC=""/>
     <VER SRC=""/>
     <REV SRC=""/>
     <USERID SRC=""/>
     <EFFDATE SRC=""/>
     <MODE SRC="" TGT=""/>
     <STATUS SRC="TEST" TGT="ROLLEDBACK"/>
     <CLASS SRC="" TGT=""/>
     <PROJECT SRC="" TGT=""/>
     </ROLLBACK>
</LBYSCRIPT>
```

# LBYSYNC

Use the LBYSYNC utility to synchronize libraries. To use this utility, simply designate a library to sync from and a library to sync to. You can also designate synchronization criteria, which lets the LBYSYNC restrict library objects synchronized to those with a specific MODE, STATUS, and CLASS.

## Program names

Windows LBYSYNC.EXE

## Syntax

**LBYSYNC /FROMLBY /TOLBY /INI /TEST /CRIT /C /R /V**

| Parameter | Description |
|---|---|
| /FROMLBY | The name of the library to synchronize from. |
| /TOLBY | The name of the library to synchronize to. |
| /INI | The INI file you want the utility to use. The default is the FSIUSER.INI file. |
| /TEST | Include this parameter if you want to simulate a synchronization without actually copying any files to the target library. This lets you see what the utility would do without changing anything. |
| /CRIT | Specifies the synchronization criteria to use. This criteria determines which resources in the source library are eligible to be synchronized to the target library and what values to set in the source and target library for each resource upon a successful promotion. The criteria string consists of three sets of three parameters.<br>The first set of parameters specifies the values for the Mode, Status, and Class fields of the resources in the source library eligible for promotion.<br>The second set of parameters specifies the values to set in the target library for the Mode, Status, and Class fields if the resource is successfully promoted.<br>The third set of parameters specifies the values to set in the source library for the Mode, Status, and Class fields if the resource is successfully promoted to the target library.<br>The fields are semi-colon delimited. Specify the criteria like this:<br>`;SMODE;SSTATUS;SCLASS;TMODE;TSTATUS;TCLASS;FMODE;FSTATUS;FCLASS;`<br>If you omit the /CRIT parameter, it defaults as shown here:<br>`;*;*;*;*;*;*;*;*;*;`<br>This tells the utility:<br>- All resources in the source library are eligible to be promoted to the target library<br>- Mode, Status, and Class fields of the resource in the target library will be set to the value of those fields from the source library.<br>- Mode, Status, and Class fields of the resource in the source library will remain unchanged upon a successful promotion. |

| Parameter | Description |
|---|---|
| /C | Causes a counter to show the progress of the utility as it processes library entries. |
| /R | Specifies which revisions of the resource to synchronize. Enter LAST if you want the utility to synchronize only the latest revision of the resource. The default is ALL which synchronizes all revisions of the resource. |
| /V | Specifies which version of the resource to synchronize. Enter LAST if you want the utility to synchronize only the latest version of the resource. The default is ALL which synchronizes all versions of the resource. |

In this example:

```
;SMODE;SSTATUS;SCLASS;TMODE;TSTATUS;TCLASS;FMODE;FSTATUS;FCLASS;
```

- SMODE, SSTATUS and SCLASS are the mode, status, and class of the resources in the source library you want to copy
- TMODE, TSTATUS and TCLASS are the mode, status, and class you want to assign to the resource entry in the target library as it is stored in the target library
- FMODE, FSTATUS, and FCLASS are used to set the final mode, status, and class of the resource entry in the source library, once the resource has been successfully promoted to the target library

You can enter an asterisk (*) for the SMODE, SSTATUS, or SCLASS fields to indicate *any value*.

You specify the synchronization criteria using the /CRIT parameter or in the INI file with the SyncCriteria option. If you use the /CRIT parameter, you can only enter a single record of synchronization criteria. You can enter multiple records of synchronization criteria using the SyncCriteria option.

### Example 1 - Synchronizing with no parameters

In this example, resources in the library named MASTER.LBY are synchronized to the library named PROD.LBY. No synchronization criteria is specified, so the default of

```
;*;*;*;*;*;*;*;*;*;*;
```

is used, which means any source library resource whose index record contains a newer modification date than the newest modification date for a resource of that same name, type, version, and revision in the target library, will be copied.

```
E:\fap\dll> lbysync /fromlby=c:\fap\mstrres\rpex1\deflib\master.lby
/tolby=c:\fap\mstrres\rpex1\deflib\prod.lby /c

   ---      DocuCorp LBYSync Utility Program (C)      ---
   ---              Synchronize Library               ---

   Inserting default LBYSync criteria  ;*;*;*;*;*;*;*;*;*;*;
   Synchronization performed. The following number of objects were
   added to the target library.

   SOURCE LIBRARY:     c:\fap\mstrres\rpex1\deflib\master.lby
   TARGET LIBRARY:     c:\fap\mstrres\rpex1\deflib\prod.lby

   FAPs :      61
   DDTs :      61
```

```
LOGOs:       5
DALs :       0
Total:     127
```

### Example 2 - Synchronizing using the /CRIT parameter

In the example below, resources in the library named MASTER.LBY are synchronized to the library named PROD.LBY.

The synchronization criteria is supplied using the /CRIT parameter. This tells the utility to copy resources from the MASTER.LBY source library that have a mode of *DEV*, a status of *PASSED*, and are in any class, into the PROD.LBY target library.

As the resources are copied into the PROD.LBY library, the utility assigns a new mode of *PROD*. The status and class do not change. Each resource in the source library that is successfully promoted is assigned a new status of *PROMOTED*.

```
E:\fap\dll>lbysync /fromlby=c:\fap\mstrres\rpex1\deflib\master.lby
/tolby=c:\fap\mstrres\rpex1\deflib\prod.lby /
crit=;DEV;PASSED;*;PROD;*;*;*;PROMOTED;*;

   ---      DocuCorp LBYSync Utility Program (C)      ---
   ---               Synchronize Library              ---

   Synchronization performed. The following number of objects were
   added to the target library.

   SOURCE LIBRARY:     c:\fap\mstrres\rpex1\deflib\master.lby
   TARGET LIBRARY:     c:\fap\mstrres\rpex1\deflib\prod.lby

   FAPs :       1
   DDTs :       1
   LOGOs:       0
   DALs :       0
   Total:       2
```

# LOG2IMG

Use the LOG2IMG utility to convert a logo (bitmap) file into a Xerox graphic image (IMG) file. An IMG file is a Xerox printer resource that contains bitmap graphics information.

## Program names

| | |
|---|---|
| Windows | LOG2IMGW.EXE |

## Syntax

**LOG2IMGW /A /I /O /R /M /C**

| Parameter | Description |
|---|---|
| /A | Include this parameter to create all four rotations. |
| /I | Enter the name of the LOG file. |
| /O | (Optional) Enter the name you want to assign to the IMG file. |
| /R | Enter the rotation of image (0, 90, 180, 270). The default is zero (0). |
| /M | Enter the mode. You can select from ENC, LIN, or HTN compression modes or select UNC for uncompressed image mode. |
| /C | (Optional) Include to create a red, green, blue, ruby, violet, brown, gray, cardinal, royal, cyan, or magenta image. |

Parameters /A and /R are mutually exclusive. If you include the /A parameter, you need to specify four names in the LOGOED window.

This utility also automatically adds tape header information at the beginning of the Xerox IMG file as some products expect a 128-byte block tape header at the beginning of the file.

# LOG2JPG

Use the LOG2JPG utility to convert a logo (bitmap) file into a JPEG graphic file. This utility can convert any color, gray, or 2-bit single color images into compressed JPEG files. During the conversion, the utility tells you which files are converted.

**NOTE:** You can also use the File, Save As option in the Logo Manager or the File, Convert, Logos option in Docucreate to convert logos to JPEG files.

## Program names

Windows LOG2JPGW.EXE

## Syntax

**LOG2JPGW /I /O /R /C**

| Parameter | Description |
|---|---|
| /I | The logo file for input with or without the extension (.log). When converting a large number of logo files, simply enter /i=*. |
| /O | The name of the JPEG file you want to create. Enter the name with or without the extension (.jpg). If omitted, the utility uses the logo's file name and the JPG extension. |
| /R | Specifies the orientation or rotation. If omitted or specified incorrectly, no rotation takes place.<br>1 - 0 degrees (portrait)<br>2 - 90 degrees<br>3 - 180 degrees<br>4 - 270 degrees (landscape)<br>5 - the rotation specified in the rotation names in the logo header.<br>The settings /r=5 and /o=ofile are mutually exclusive. When you specify /r=5, the output file uses the rotation name in the logo header. If there is no rotation name, the output file takes input file as a default. |
| /C | Tells the utility to convert a color logo to a single color. Specify the color using these values:<br>1-black 9-dark blue<br>2-blue 10-dark cyan<br>3-cyan 11-dark green<br>4-green 12-dark magenta<br>5-magenta 13-dark red<br>6-red 14-dark yellow<br>7-yellow 15-dark gray<br>8-white 16-light gray |

# LOG2LOB

Use the LOG2LOB utility to convert a LOG (logo) file into a DOS entry LOB file.

**NOTE:** You can also use Docucreate's File, Convert, LOG to LOB option to perform this task.

## Program names

| | |
|---|---|
| Windows | LOG2LB32.EXE |

## Syntax

**LOG2LB32 /I /O**

| Parameter | Description |
|---|---|
| /I | Enter the name of the LOG file. |
| /O | Enter the name of the LOB file. |

**NOTE:** The LOB extension defines a FormEntry file format.

# LOG2PSEG

Use the LOG2PSEG utility to compile a LOG (logo) file into an AFP page segment. Use this utility to apply rotations to logos before downloading them to an AFP printer. You can also enlarge the logo using the scale option.

## Program names

| | |
|---|---|
| Windows | LOG2PG32.EXE |
| MVS | See the Documaker Server Installation Guide |

## Syntax

**LOG2PG32 /I**

| Parameter | Description |
|---|---|
| /I | Enter the name of the file which contains records in this format:<br>`;name;logofile;outfile;scale;orientation;`<br>The default extension is *DAT*. (you can use asterisks as wildcards) |

or

**LOG2PG32 /I /O /C /S /R**

| Parameter | Description |
|---|---|
| /I | Enter the name of the logo file. The default extension is *LOG*. |
| /O | Enter the name of the output file. The default extension is *SEG*. |
| /C | Add this parameter if the AFP overlay should use color. Enter one of the named AFP colors such as blue, red, magenta, green, cyan, yellow, dark_blue, orange, purple, dark_green, dark_cyan, mustard, gray, or brown. |
| /S | Enter 1 or 2. The default is 1 (normal), 2 indicates enlarge. |
| /R | Enter 1 (0 degrees-portrait), 2 (90 degrees), 3 (180 degrees), 4 (landscape 270 degrees), or 5 (all rotations). |

**NOTE:** The utility ignores the /O option if you include /R=5. The utility will instead use data from the LOG file.

## Printing in color

Include the /C parameter if the AFP overlay should print in color. AFP highlight color printing on printers from Xerox and Oce is supported. Before using this feature, make sure the:

- SendColor INI option is set to Yes.
- Objects you want to print in color (text, lines, shades, and so on) are set to print in color. The Print in Color option is on the Color Selection window in the Image Editor. You can display this window by clicking the Color button on the object's Properties window.

The system maps the RGB (red,green,blue) color setting for each object to the closest AFP named color. The default AFP named colors are: blue, red, magenta, green, cyan, yellow, dark_blue, orange, purple, dark_green, dark_cyan, mustard, gray, and brown.

Use the NamedColors option to tell the system to use only specific AFP named colors:

```
< PrtType:AFP >
    NamedColors = blue
```

For example, if you wanted all highlight (non-black) colors mapped to blue, you would set the NamedColors option to blue, as shown above.

To allow the mapping of the colors you assigned to the objects in the FAP file to multiple colors, separate each color with a semicolon (;). For example, to use all of the default AFP named colors except brown, set the NamedColors option as shown here:

```
NamedColors = red;blue;magenta;green;cyan;yellow
```

The order of the named colors does not matter.

# LOG2TIF

Use the LOG2TIF utility to convert any logo into a TIFF file. During the conversion, the utility tells you whether or not each file was converted and then provides a summary when finished.

## Program names

| | |
|---|---|
| Windows | LOG2TIFW.EXE |

## Syntax

LOG2TIFW /I=logofile [/O=tiffile] [/C]

| Parameter | Description |
|---|---|
| /I | The name of the logo file. The extension defaults to .LOG. Enter an asterisk (*) if you want the utility to convert all the logo files in the current directory. |
| /O | (Optional) Enter the name you want the utility to assign to the resulting TIFF file. The extension defaults to .TIF. |
| /C | (Optional) Include this parameter to convert the logo into a monocolor TIFF image. |

## Example

Here are some examples:

```
LOG2TIFW /I=BEAST004.LOG /O=BEAST.TIF /C
LOG2TIFW /I=*
```

# LOG2VIPP

Use the LOG2VIPP utility to convert multi-color logos into JPG files and monocolor logos into TIFF files for use as VIPP printer resources.

## Program names

| | |
|---|---|
| Windows | LOG2VIPW.EXE |

## Syntax

```
LOG2VIPW /I=logofile [/C]
```

| Parameter | Description |
|---|---|
| /I | Enter the name of the logo file. The utility assume the default extension .LOG. To process all logo files in the current directory, enter an asterisk (*). |
| /C | (Optional) Include this parameter to convert to a monocolor TIFF image. This is useful when you have the SendColor INI option set to No for producing VIPP print streams. |

**NOTE:** If you include the /C parameter, the utility creates TIFF files for all logos.

# LOG2XFNT

Use the LOG2XFNT utility to compile a logo (LOG) file into a Xerox Metacode font file.

**NOTE:** You can also use the Logo Manager to perform this task.

## Program names

| | |
|---|---|
| Windows | LOG2XF32.EXE |

## Syntax

**LOG2XF32 /I**

| Parameter | Description |
|---|---|
| /I | Enter the name of the LOG file. |

This utility converts a LOG file into a Xerox font file (FNT). This allows a bitmap graphic to be treated as a font on the printer. There will be a font file created for each rotation (0, 90, 180, and 270 degrees) and the output files will have FNT extensions.

**NOTE:** There is a limit to the size of a Xerox font. Because of this limit, you cannot convert logos larger than 128k.

# MET2FAP

Use the MET2FAP utility to convert a Xerox Metacode file created by the system into a FAP file. You can then view and edit the FAP file using the Image Editor.

**NOTE:** This utility supports DJDE FORMAT commands contained in the Metacode print stream you are converting.

## Program names

Windows MET2FAPW.EXE

## Syntax

**MET2FAPW /I /X /P**

| Parameter | Description |
|---|---|
| /I | Enter the name of the MET file. Omit the extension. |
| /X | Enter the name of the font cross-reference (FXR) file. Omit the extension. |
| /P | Enter the PrtType control group in the INI file to use, such as XER. |

This utility requires the following files:

- FSISYS.INI
- *.FXR
- LOGO.DAT file should be included, but is not required, in the following format: logo0;logo90;logo180;logo270;

```
FLOGO;FLOGO9;FLOGO1;FLOGO2;
```

This utility reads a Metacode print file created using the system and creates a FAP file.

In addition to a Metacode print file (MET), you must also specify a font cross-reference (FXR) file which contains the Xerox fonts used.

**NOTE:** The MET2FAP utility only works with Metacode print files created by Documaker Server. To convert a Metacode print file which was created by the Documerge system, use the MRG2FAP utility. For more information, see MRG2FAP on page 163.

# METOPT

Use this utility to optimize Metacode print streams before they are sent to the printer. This utility combines Metacode print records into larger and fewer records. Reducing the number of records that must be transmitted reduces the amount of time needed to spool the Metacode print stream to the printer.

The METOPT utility detects errors in the Metacode information caused by the data passed or by the compiled resources used. This helps you detect errors before printing. This utility also notes comment (or pure text) records that are loaded and will output a copy of the most recent comment record when it detects errors in a Metacode print stream.

The utility also lets you use common font lists at the beginning of a Metacode print stream. A common font list names all of the Xerox fonts that will be used by the print job. This helps some Metacode printers print jobs at their highest rated speed.

## Program names

| | |
|---|---|
| Windows | METOPT32.EXE |
| MVS | see below |

## Syntax

**METOPT32 /I /O /N /X**

| Parameter | Description |
|---|---|
| /I | The name and location of the input file to be optimized |
| /O | The name and location of the output optimized file to be created by the MetOpt utility. |
| /N | (Optional) The name and location of an INI file to use. The default value is *FSISYS.INI* for the PC and *DD:FSISYS* for MVS. The INI file used for running this utility should be the same INI file used to produce the print stream to be optimized. |
| /X | (Optional) The name and location of an FXR file to use other than the default REL103.FXR on the PC and DD:FXRFILE on MVS. The FXR used for running METOPT should be the same FXR used to produce the print stream to be optimized. |

You can also enter a question mark (?) after the command to see a list of all parameters.

Keep in mind this utility does not work with precompiled MET files.

## Running METOPT on MVS

The METOPT utility is currently supported for use using BARR and JES2 output modes (PrtType:XER control group, OutMode option).

Consider the sample JCL below for running METOPT on MVS:

```
//MRCC  JOB   (33005),'RUN METOPT    ',CLASS=T,MSGCLASS=X, NOTIFY=MRC
//*
//          SET HLQ='MRC.V100'     <== SET HIGH LEVEL QUALIFIER
//          SET RES='DAPDATA'      <== SET RESOURCE
//*
//          JCLLIB  ORDER=&HLQ..PROCLIB
//*
//METOPTD EXEC PGM=IEFBR14
//DD1     DD DSN=&HLQ..&RES..OUTPUT.OPT,DISP=(MOD,DELETE),
//        SPACE=(TRK,(1,1)),UNIT=SYSDA
//*
//METOPT  EXEC PGM=METOPT
//STEPLIB  DD DSN=&HLQ..LINKLIB,DISP=SHR
//         DD DSN=SYS1.SCEERUN,DISP=SHR
//FSISYS   DD DSN=&HLQ..&RES..DEFLIB(FSISYS),DISP=SHR
//PFRMLIB  DD DSN=&HLQ..&RES..PFRMLIB, DISP=SHR
//FXRFILE  DD DSN=&HLQ..&RES..DEFLIB(FONTFILE),DISP=SHR
//INFILE   DD DSN=&HLQ..&RES..INPUT,DISP=SHR
//OUTFILE  DD DSN=&HLQ..&RES..OUTPUT.OPT,DISP=(,CATLG),
//         SPACE=(CYL,(20,5)),LIKE=&HLQ..&RES..INPUT
//SYSPRINT DD SYSOUT=*
```

Note that the input and output DD statements (*INFILE* and *OUTFILE*) refer to physical sequential datasets. The INI member FSISYS and the FXR member FONTFILE in this example are both contained in a partitioned dataset called *DefLib*. Also note that the DD statements, FSISYS, PFRMLIB, and FXRFILE, reference the INI and FXR files instead of command line parameters.

## FRM support

For this utility to correctly optimize print streams that reference printer-resident FRM files, you must make available a copy of each FRM file.

In a Windows environment, make sure the FRM files are in the same directory as this utility.

On MVS systems, the METOPT utility uses the standard naming convention PFRMLIB for the PDS that contains the pre-compiled FRM members the utility will need.

## Using a common font list

The METOPT utility lets you use common font lists at the beginning of a Metacode print stream. A common font list names all of the Xerox fonts that will be used by the print job.

By knowing all of the fonts up front, the Metacode driver can issue a single DJDE FONTS command once at the beginning of the job and avoid issuing DJDE FONTS commands on subsequent pages. This helps some Metacode printers print jobs at their highest rated speed.

In the CommonFonts control group, you will see a list of options similar to these:

```
< CommonFonts >
    Names=28
    Name1=FORMSX
    Name2=FXUNBD
    Name3=FXUNN6
```

```
Name4=FXCON6
Name5=FXUNN8
Name6=FXUNN0
Name7=FXUNBH
...
Name28=FXUNI0
```

The first option, Names, defines the number of font name entries that follow. The following options specify the Xerox fonts which will be used in the print job.

**NOTE:** The format used for the CommonFonts control group is the same as that used by Documerge. Therefore, if you used this in Documerge, you can copy that INI control group into your Documaker INI file.

## Errors

If this utility encounters critical errors, such as the inability to find or open a file, while it is running, it will notify you and stop immediately. Common critical errors include:

- Cannot create the output file
- Cannot read the output file
- Cannot open the FRM file

The METOPT utility can also report actual or potential non-critical problems it encounters while it runs. For instance, if the utility finds Metacode records that may prevent the file from printing, it can warn you. These non-critical errors usually fall into two categories:

- One of the METOPT utility's required steps was unable to be completed, such as building the font symbol table.
- Invalid input data of some kind was encountered, so the METOPT utility's output data may be invalid, too. This could mean that some data in the optimized file will not be printed, will not print correctly, or will generate printer errors.

To have this utility notify you if it spots potential problems, add the following option to your PrtType:XER control group:

```
< PrtType:XER >
    ValidLevel =
```

| Option | Description |
|---|---|
| 0 | Enter zero (0) to tell the utility not to report non-critical problems. This is the default. |
| 1 | Enter one (1) to tell the utility to report warnings for non-critical problems, but continue optimizing. |
| 2 | Enter two (2) to tell the utility to report warnings for non-critical problems and attempt to fix the problems |
| 3 | Enter three (3) to tell the utility to report warnings for non-critical problems and exit immediately |

Using the ValidLevel option gives you a degree of control over how certain errors are handled. Keep in mind that, although classified as non-critical, many of these errors can produce invalid output. So, many of the more severe non-critical errors will be promoted automatically by METOPT to a ValidLevel of 3. METOPT will then display the error and stop executing. Such instances usually fall into the first category described above.

Usually, the non-critical errors that do not force METOPT to stop processing involve one corrupt record, such as a record that contains an invalid Metacode sequence. In most cases, if you set the ValidLevel option to 1, a warning appears and the record displays, but the utility continues to process the record.

For the same record, if you set ValidLevel to 2, the same warnings appear, but this time the record is not processed with the others. So, since that record is left alone, the printer will treat it as it would have the identical record in the input file.

**NOTE:** When you set the ValidLevel option to a value other than zero and the problem involves a particular Metacode record, the METOPT utility typically displays the record in hexadecimal to help you solve the problem.

You can also use the following INI option to help track down and resolve errors:

```
< PrtType:XER >
    SaveComment =
```

The SaveComment option lets you save the last Metacode comment record processed and display it when an error is encountered. This can be a helpful placeholder when you are trying to isolate the location of an error record in a large file. If the Metacode file contains occasional or frequent comment records, then the SaveComment option can be a useful troubleshooting tool.

The possible values for the SaveComment option are:

| Option | Description |
|---|---|
| (blank) | Do not save comments. This is the default. |
| S | Display the last comment as-is, with no conversion. |
| A | Convert the last comment to ASCII, then display. |
| E | Convert the last comment to EBCDIC, then display. |
| H | Display the last comment in hexadecimal. |

# MRG2FAP

Use the MRG2FAP utility to convert a Documerge AFP or Metacode file into a FAP file. This utility also converts AFP print files created by the system into FAP files. You can then view and edit the FAP file using the Image Editor. In addition to creating a FAP file, you can also create a PDF file.

The MRG2FAP utility lets you load Xerox FRM files and IMG files that are referenced in the Metacode print stream being converted. It can also produce a BPSD/Field cross-reference listing.

The system looks for the FRM and IMG files in the directory specified by the FormLib option in the MasterResource control group. If you omit this option, the system looks in the current directory.

**NOTE:** This utility supports DJDE FORMAT commands contained in the Metacode print stream you are converting.

## Program names

Windows MRG2FAPW.EXE

## Syntax

**MRG2FAPW /I /T /X /P /L /A**

| Parameter | Description |
|---|---|
| /I | Enter the name of the AFP file (include the extension) or Metacode file (omit the extension). |
| /T | Enter the file type: AFP or MET. |
| /X | Enter the name of the font cross-reference (FXR) file. Omit the extension. |
| /P | (Optional) Enter the PrtType control group in the INI file to use, such as XER. |
| /L | (Optional) Include to create a listing of the BPSD tag field names from the input file and the subsequently created FAP field names. You must specify the file name in the BPSDReport option in the MasterResource control group. |
| /A | (Optional) Include to also create a PDF file. The system uses the PDF settings in the PrtType:PDF control group. |

The MRG2FAP utility requires the following files:

- FSISYS.INI
- *.FXR
- IBMXFER.TBL cross-reference table which correlates the coded font names with character sets and codepages (only for Documerge AFP output)
- LOGO.DAT file should be included, but is not required. The LOGO.DAT file is a text file that specifies the names and rotations of logo fonts. If there are any logo fonts in the Metacode file, MRG2FAP uses a LOGO.DAT file in the FormLib directory so it can convert logo fonts for all rotations into a single DAP/RP logo.

The LOGO.DAT file, which is a semicolon-delimited file, should look similar to this:

```
[file name for 0° rotation];[file name for 90° rotation];[file name
for 180° rotation];[file name for 270° rotation];
```

Here is an example:

```
logo0;logo90;logo180;logo270;
```

If you do not have a LOGO.DAT file, you will see a warning message.

- In addition to an AFP or Metacode print file, you must also specify a font cross-reference (FXR) file which contains the AFP or Xerox fonts used.

**NOTE:** This utility only works with Metacode print files created by Documerge. To convert a Metacode print file which was created by Documaker Server, use the MET2FAP utility. You can use the MRG2FAP utility to convert AFP print files created by either Documerge or Documaker Server into FAP files.

### Handling overlays and page segments

If the AFP print file contains references to overlays or page segments, copy the overlay or page segment files into the directory in which the AFP print file resides. Add the following INI options in the PrtType:AFP control group in the FSISYS.INI file to specify the file extension for overlay and page segment files.

```
< PrtType:AFP >
    OverlayExt=
    PageSegExt=
```

### Keeping blank pages

Use the KeepBlankPages option when you are converting AFP and Xerox Documerge files into FAP files to retain blank pages. Here is an example:

```
< PrtType:AFP > or < PrtType:XER >
    KeepBlankPages = Yes
```

Normally blank pages are removed because the system assumes they are duplex back pages that are not needed. If, however, you want to retain these pages, add this option and set it to Yes. The default is No which indicates you do want to remove blank pages during a conversion.

### Importing logos

You can select a logo to be imported into a FAP file when you are converting from an AFP file. This logo is placed on the page based on a medium map tray selection. The logo is imported so it can mimic paper in a particular tray that has a preprinted logo.

The options for the tray logo will be placed within the print driver control group. The system supports up to nine trays so there are up to nine available INI options for the bitmaps. You must include the name of the logo file, followed by the top and left coordinates specified in FAP units (2400 per inch).

Here is an example:

```
TrayLogo1 = logoa.log,300,4800
TrayLogo3 = logob.log,300,6500
TrayLogo9 = logoc,200,2000
```

Keep in mind that you can only import logos.

### Suppressing LOG files when converting AFP to FAP or PDF

You can use the new SuppressLogoUnload option when converting AFP files into FAP or PDF format to avoid possible conversion errors which can arise from converting the same AFP file multiple times.

```
< PrtType:AFP >
    SuppressLogoUnload = Yes
```

| Option | Description |
|---|---|
| SuppressLogoUnload | Enter Yes to suppress the unloading of logo (LOG) files during a conversion of AFP files to FAP or PDF format. You should always set this option to yes when creating PDF files. The default is No. |

When you enter Yes, AFP to PDF conversions, whether run from the MRG2FAP utility or IDS, do not result in new LOG files being unloaded from AFP page segments. The LOG files were unloaded for use by FAP files and are not needed for the conversion to PDF format.

Unless you need to look at or use the FAP files, set this option to Yes. If you do need the FAP files, you must also delete all LOG files before you run the MRG2FAP utility to make sure you are getting the correct results.

**NOTE:** This option does not delete any LOG files created by a prior conversion, so you should delete all LOG files before you use the SuppressLogoUnload option.

## Specifying Code Pages

You can use font definition files to specify the code pages you want to use in a document. The system loads these files as it loads the AFP print stream. For instance, you can use...

- The default EBCDIC code page for every font used in a document (just as before)
- The default EBCDIC code page for some fonts and a specified code page for other fonts
- Specified code pages for all of the fonts in the document

**NOTE:** Prior to version 11.1, Documaker Server handled traditional AFP font resources (character sets, code pages, and coded fonts), but only allowed you to use a single, default code page for each document. The AFP loader translated input AFP text from EBCDIC to ASCII using a default EBCDIC code page.

Here is a brief summary of the font definition files you can use to specify code pages:

| File | Description |
|---|---|
| CODED.FNT | The coded font definitions. This file specifies which AFP code page and AFP font character set make up the coded font. |
| CPDEF.FNT | The code page definitions. This file maps each AFP code page to a Windows character set. |
| CPGID.CP | The code page map file. This file contains the character identifiers (and associated EBCDIC hexadecimal code points) for an IBM code page and maps them to character identifiers (and associated ASCII code points) for a Windows ANSI or SYMBOL character set. |

Here are the general syntax rules for the font definition files:

- A semicolon (;) in the first column of any of these files will cause the line to be treated as a comment statement and ignored.
- Section headers within files are enclosed either in brackets (< > or []) with *no* spaces and must *not* be removed or changed.
- All values are case insensitive.
- If a parameter value is invalid and a default value exists, it will be substituted.
- All parameters are positional.
- Blanks are allowed between parameter values.
- The question mark (?) is used in some areas as a single wildcard character.
- If the resource file exists in DEFLIB directory and contains valid data conforming to these specifications, it will be loaded and used.
- If bad data is encountered in the file, either the offending record is ignored or a warning is issued. If the file is considered corrupt or invalid enough, it may not be used at all.

### CODED.FNT file

This file specifies which AFP code page and AFP font character set make up the coded font. The CODED.FNT file is necessary for basic multiple code page support. When creating this file, keep these rules in mind:

- The coded font name and both parameters are required.
- A question mark (?) can be used as the wild-card character only for the second character in the coded font name and for any character of the character set name. This allows all the character rotations of the coded fonts to be handled with one entry for searching.
- After the coded font name, the character set name must be listed first, followed by the code page name.
- The character set and code page must be separated by a comma.

Here is an example of this file:

```
X?COL8=C?420080,T1000850
X?COL7=C?420070,T1000850
;Core
X?H210AC=C?H200A0,T1V10500
```

```
X?H210FC=C?H200F0,T1V10500
;FormMaker Fonts
X?FA????=C?FA????,T100ASC4
X?DA????=C?FA????,T1DOC037
X0P09X12=C0P09X12,T1DOC037
X0P12X16=C0P12X16,T1DOC037
```

### CPDEF.FNT file

This file maps each AFP code page name to its code page global identifier (CPGID) and to a Windows character set. If you do not have at least one valid entry in this file for each code page you want to use, the system uses the default code page. When creating this file, keep these rules in mind:

- Parameters must be separated by a comma.
- AFP code page name and code page identifier are required.
- If you create your own code page, you must assign it a unique code page identifier. Leading zeros are invalid.
- Code Page Global Identifier (CPGID) attribute's possible values: IBM-defined CPGID or your own defined CPGID between 65280 and 65534, inclusively. This value matches the name of a code page map file.
- For each CPDEF.FNT entry, you must have a corresponding code page map file with the same name as the CPGID.
- Windows character set attribute's possible values: ANSI or SYMBOL.

Here is an example of this file:

```
<CODEPG>
;codepage = cpgid,wincp
;*****Put User-defined/Custom code pages Here *****
T100ASC4=361,ANSI
T1DOC037=37,ANSI
T1OMR=5280,ANSI
T1POSTBC=5280,ANSI
;******* End User-defined/Custom code pages ******
T1000259=259,SYMBOL
T1000290=290,ANSI
T1000293=293,ANSI
T1000310=310,ANSI
DEFAULT=361,ANSI
```

### *CPGID*.CP (code page map file)

You must have a separate *CPGID*.CP file for each AFP code page entry in the CPDEF.FNT file. Each code page map file contains the character identifiers (and associated EBCDIC hexadecimal code points) for an IBM code page and maps them to character identifiers (and associated ASCII code points) for a Windows ANSI or SYMBOL character set. Code page map files are necessary for basic multiple code page support.

**NOTE:** The actual file name is not CPGID.CP, but rather the *CPGID* value from the CPDEF.FNT file with an extension of *CP*. For instance, in the CPDEF.FNT example, the first two lines are:

```
T100ASC4=361,ANSI
```

```
T1DOC037=37,ANSI
```

So, since those two entries are in the CPDEF.FNT file, that means that there must be code page map files with named *361.CP* and *37.CP*.

Also, if these two entries are in the CPDEF.FNT file, but the corresponding 361.CP and 37.CP code page map files are not in DEFLIB, the translations for those fonts will not be correct.

---

When creating this file, keep these rules in mind:

- Parameters must be separated by blanks.
- All four parameters are required.
- *NOMATCH* means there is not a matching character in the Windows character set.

Here is an example of this file: (395.cp for the T1000395 code page mapped to the Windows ANSI character set):

```
;T1000395 to ANSI
SP010000 40 SP010000 20
LA150000 42 LA150000 E2
LA170000 43 La170000 E4
LA130000 44 LA130000 E0
SP180000 8B SP180000 BB
SM560000 8C SM560000 89
SA000000 8D SP100000 2D
LI510000 8E NOMATCH 00
LI570000 8F NOMATCH 00
SM190000 90 SM190000 B0
LJ010000 91 LJ010000 6A
LF510000 A0 NOMATCH 00
;;;;;;;; ;  SD150000 5E
;;;;;;;; ;  SD130000 60
```

## Building Metacode Resources

The FSISYS.INI and FAPCOMP.INI files are system initialization files used by various Skywire Software programs, including this utility. You must add a PrtType:XER control group to these INI files. This control group contains the Xerox Metacode options used for the archived Metacode print streams.

### PrtType Control Group

Below is an example of the PrtType:XER control group, which contains these options:

```
< PrtType:XER >
    DJDEIden   = A'@@@DJDE'
    DJDEOffset = 0
    DJDESkip   = 8
    OutMode    = BARR
    ImageOpt   = No
    JDEName    = DFLT
    JDLCode    = NONE
    JDLData    = 0,255
    JDLHost    = IBMONL
    JDLName    = CBA
    JDLRStack  = 0,10,EQ,X'13131313131313131313' (optional)
    JDLRPage   = 1,5,EQ,X'FFFF26FFFF'            (optional)
    PrinterInk = Blue
    PaperSize  = 0
    DefaultFont= 11010
```

Several of these INI settings are based on comparable options and values in the settings of the printer's JSL. A JSL may contain many JDLs from which to choose, or there may be multiple JSLs compiled into multiple JDLs.

An excerpt of a JDL follows, along with an explanation of each of the PrtType:XER control group options.

#### JDL Example

Here is an excerpt of a JDL. This excerpt is referenced in the control group options discussion.

```
CBA:    JDL;

T1:     TABLE    CONSTANT=X'121212121212121212';
T2:     TABLE    CONSTANT=X'13131313131313131313';
T3:     TABLE    CONSTANT=X'FFFF26FFFF';
C1:     CRITERIA CONSTANT=(0,9,EQ,T1);
C2:     CRITERIA CONSTANT=(0,10,EQ,T2);
C3:     CRITERIA CONSTANT=(1,5,EQ,T3);
VOLUME            HOST=IBMONL;
LINE              DATA=(0,255);
IDEN              PRE=A'@@@DJDE',
                  OFF=0,
                  SKIP=8;
ROFFSET           TEST=C1;
RSTACK            TEST=C2,DELIMITER=YES,PRINT=NONE;
RPAGETEST=C3,SIDE=NUFRONT;

/* 8.5 x 11 job */
USA1: JDE;       /* JOB can be used in place of JDE  */
```

```
OUTPUT              PAPERSIZE=USLETTER;

/* 8.5 x 14 job */
META: JOB;
VOLUME              CODE=NONE

/* Default job */
DFLT: JDE;
VOLUME              CODE=EBCDIC

END;
```

### DJDEIden, DJDEOffset, and DJDESkip

These options represent the IDEN statement of the JDL. The value of the DJDEIden setting is a string constant. The types of supported string constants are ASCII (A'string'), EBCDIC (E'string'), Character ('string'), and Hex (X'string').

These types of strings are not supported: Octal, H2, and H6. Strings containing repeat counts, embedded hex values, and upper/lower case toggles are not supported. Using the JDL sample listed earlier, the INI options should be:

```
DJDEIden      = A'@@@DJDE'
DJDEOffset    = 0
DJDESkip      = 8
```

### OutMode

This option indicates the output format for the Metacode data stream generated by your Documerge system. You have these options:

Enter **BARR**, if you generate output using a Windows system and then transmit that output to a Xerox printer using BARR SPOOL hardware and software. If you choose BARR, a length byte is placed at the start and end of each Metacode record.

Enter **BARRWORD** only if records longer than 255 characters can be handled by your Xerox printer.

Enter **ELIXIR** to convert Elixir-formatted Metacode print files into FAP files.

For normalized Metacode, the system supports the standard Documerge 4-byte ISI format and the 2-byte variable (ISI 2-byte) format. Enter **MRG4** to use the Documerge 4-byte ISI format. Enter **MRG2** to indicate you want to use the 2-byte variable (ISI 2-byte) format.

Enter **PCO** if you generate output using a Windows system and then transmit that output to a Xerox printer using PCO hardware and software (from Prism). When you select PCO, a 4-byte length field is placed at the start of each Metacode record.

**NOTE:** Skywire Software has not completely tested the PCO interface.

Enter **JES2** for MVS environments. If you will upload output generated on a Windows system to an MVS system and then transmit the output to your printer via JES2, use OutMode = JES2.

Enter **ENTIRE** if you will transmit output generated by a Windows or UNIX system to a Xerox printer via a Sun workstation using ENTIRE/FIBER GATEWAY hardware and software (from Entire, Inc.). When you choose ENTIRE, a 2-byte length field is placed at the start of each Metacode record.

Enter **LAN4235**, if you generate output for a Xerox 4235 printer attached to a network.

Here is an example of this INI option:

```
OutMode        = BARR
```

**NOTE:** This version assumes Metacode output produced by Documerge which does not correspond to any of the outmodes listed above. You must, however, still choose an outmode from those options listed above.

### ImageOpt

Use this option to specify if the logos are saved on the Xerox printer as IMG files or as FNT files. To use IMG files, your printer must have GVG or GHO hardware installed. Also, in the JSL, you must set the Graphics option to Yes.

If you are using IMG files, set this option to Yes; otherwise set it to No. Metacode printers have a limit of 16 images on a page. Here is an example of this option:

```
ImageOpt       = No
```

### JDEName

Use this option to represent the name of the job. A JDL may contain many jobs (JDEs) from which to choose. Using the JDL sample listed earlier, the Metacode job is selected using this INI setting: (This JDE must contain VOLUME CODE=NONE)

```
JDEName        = META
```

### JDLCode

Use this option to represent the type of input format expected by the Xerox printer during normal operation (that is, the JDL/JDE setting used to start the printer). Character translation is performed as necessary.

The system supports EBCDIC, ASCII, or NONE, which is the same as ASCII. These formats are not supported: BCD, H2BCD, H6BCD, IBMBCD, PEBCDIC, and user-defined code translation.

Referring to the sample JSL, if the printer is normally started with STA DLFT,CBA then the JDLCode parameter must be set to CODE = EBCDIC. The INI setting must contain the value of the CODE= statement for the printer's normal operation. Here is an example of this INI option:

```
JDLCode        = EBCDIC
```

### JDLData

Use this option to represent the starting position and length of the print line data within an input data record. The LINE statement contains a DATA entry which holds these values. An example of this INI setting is as follows:

```
JDLData        = 0,255
```

### JDLHost

Use this option to tell the system whether the printer is normally on-line or off-line. You can choose from **IBMONL** (on-line) and **IBMOS** (off-line). Using the JDL example listed earlier, this INI option should be set to:

```
JDLHost        = IBMONL
```

### Additional settings for Xerox printers

For a Xerox print driver, you must also specify these functions in the PrtType:XER control group:

```
OutputFunc     = XEROutput
OutMetFunc     = XEROutMet
InitFunc       = XERInit
TermFunc       = XERTerm
```

```
Module          = XERW32
```

**JDLName** Use this option to represent the name of the JDL to use. Using the JDL sample listed earlier, this option should be set to:

```
JDLName         = CBA
```

**JDLRStack** Use this optional INI option to represent criteria which tells the system to send an end of report condition to the printer. In the JDL example, the RSTACK statement performed a criteria test named C2. The C2 test checks a specific part of each input line against the string named T2. If the string T2 matches an input data record at position 0 for length of 10 bytes, an end of report condition is signaled. Only CONSTANT criteria using an EQ operator is supported.

**NOTE:** If the printer is alternately used for Metacode and text file print jobs, you must include the JDLRStack option. We recommend always using JDLRStack.

Using the JDL sample listed earlier, this option should be set to:

```
JDLRStack       = 0,10,EQ,X'13131313131313131313'
```

**JDLRPage** Use this optional INI option to represent the criteria which signals a jump to the front side of a new sheet to the printer. In the JDL sample listed earlier, the RPAGE statement performed a criteria test named C3. The C3 test checks a specific part of each input line against the string named T3. If the string T3 matches an input data record at position zero (0) for a length of 5 bytes, a *jump to new sheet* condition is signaled because of the SIDE=NUFRONT setting. Only CONSTANT criteria using an EQ operator is supported. The SIDE=NUFRONT setting in the JSL is required for JDLRPage to work properly.

**NOTE:** If the print job is likely to contain duplex pages alternating with simplex (one sided) pages, JDLRPAGE provides a way to leave the back sides of certain pages blank.

Using the JDL sample listed earlier, this option should be set to:

```
JDLRPage        = 1,5,EQ,X'FFFF26FFFF'
```

**PrinterInk** Use this option to specify the color of ink loaded on a Xerox highlight color printer. You can set the PrinterInk option to either Blue, Red, or Green (blue is the default). This option is used with the SendColor INI option. If you set the SendColor option to Yes, you should also set the PrinterInk option. Here is an example of this INI option:

```
PrinterInk      = Blue
```

PaperSize Use this option to specify the size of the paper. Here are your options:

| For | Enter |
|---|---|
| US letter | zero (o). This is the default. |
| US legal | 1 |
| ISO A4 | 2 |
| US executive | 3 |
| US ledger | 4 |
| US tabloid | 5 |
| US statement | 6 |
| US folio | 7 |
| US fanfold | 8 |
| ISO A0 | 20 |
| ISO A1 | 21 |
| ISO A2 | 22 |
| ISO A3 | 23 |
| ISO A5 | 25 |
| ISO A6 | 26 |
| ISO A7 | 27 |
| ISO A8 | 28 |
| ISO A9 | 29 |
| ISO A10 | 30 |
| ISO 2A | 32 |
| ISO 4A | 34 |
| ISO B0 | 40 |
| ISO B1 | 41 |
| ISO B2 | 42 |
| ISO B3 | 43 |
| ISO B4 | 44 |
| ISO B5 | 45 |

| For | Enter |
|---|---|
| ISO B6 | 46 |
| ISO B7 | 47 |
| ISO B8 | 48 |
| ISO B9 | 49 |
| ISO B10 | 50 |
| ISO 2B | 52 |
| ISO 4B | 54 |
| ISO C0 | 60 |
| ISO C1 | 61 |
| ISO C2 | 62 |
| ISO C3 | 63 |
| ISO C4 | 64 |
| ISO C5 | 65 |
| ISO C6 | 66 |
| ISO C7 | 67 |
| ISO C8 | 68 |
| ISO C9 | 69 |
| ISO C10 | 70 |
| ISO DL | 71 |
| JIS B0 | 80 |
| JIS B1 | 81 |
| JIS B2 | 82 |
| JIS B3 | 83 |
| JIS B4 | 84 |
| JIS B5 | 85 |
| JIS B6 | 86 |
| JIS B7 | 87 |
| JIS B8 | 88 |

| For | Enter |
|---|---|
| JIS B | 89 |
| JIS B10 | 90 |
| custom | 98 |

DefaultFont

Use this option when displaying the names of fonts which are not found in the font cross reference (FXR) or LOGO.DAT files. The value for the DefaultFont option is a font ID which is contained in the font cross reference (FXR) file being used.

```
< PrtType:XER >
    DefaultFont = 11010
```

## Error Messages

The following information describes how to handle error messages you may encounter while using the MRG2FAP utility on AFP print streams.

- Character set xxxxxxxx not found...

  If you receive this message, the AFP print stream uses a character set and codepage file name instead of a coded font file name to specify the AFP font to be used. In this case, you must create a file called *IBMXREF.TBL* to provide additional AFP font information. *IBMXREF.TBL* is a text file that contains pairs of coded font names and character set names. Place this file in the directory with the AFP print stream.

  What you are doing is specifying the coded font file name to use when a reference to the character set file is found in the AFP print stream. The system searches in the FXR file for the coded font file name to determine font information it needs during the PDF conversion.

  When entering the coded font and character set names in IBMXREF.TBL, do not use the first two letters (Xo, X1, Co, C1, and so on). The coded font and character set names need to be written in *UPPER CASE* and separated by at least one space. Each pair of coded font and character set names should be written on separate lines. For example, if you receive an error stating...

  ```
  Character set C0AR111 not found…
  ```

  Add a line of coded font and character set names to the IBMXREF.TBL. If a coded font file named *XoAR11P* contained a reference to the character set file *CoAR111*, you would add the following line to IBMXREF.TBL:

  ```
  AR11P AR111
  ```

  Notice the first two letters of *XoAR11P* and *CoAR111* are omitted from the line added to IBMXREF.TBL. You should have inserted the coded font file, XoAR11P, into the FXR file previously.

If you have character set files but do not have any coded font files, you can insert a character file into the FXR. However, you must edit fonts inserted into the FXR and specify a coded font file name on the Printers page. In this case, simply use the character set name as the coded font name (remembering to change the first letter from C to X). In this case, the pair of names stored in the IBMXREF.TBL will be the same.

If you have a character set file used by more than one codepage file, you may want to map each character set/codepage file combination to a coded font file named in the FXR. To do this, add a third column to the IBMXREF.TBL. The third column will contain the name of codepage file. For example, to map the coded font file, *XoAR11P*, to the character set and codepage files, *CoAR111*, and *T1ISI121*, you would add the following line to IBMXREF.TBL:

```
AR11P AR111 T1ISI121
```

Notice the first two letters of *XoAR11P* and *CoAR111* are omitted from the line added to IBMXREF.TBL but the full name of the codepage file, *T1ISI121*, is used.

- Error opening overlay: xxxxxxxx

  This message indicates the AFP print stream uses an overlay the system could not find. Note the path and file extension of the overlay specified in the error message. Make sure your AFP overlay is stored in the directory with the AFP print stream. The overlay extension is specified in the OVERLAYEXT option of the printer control group. If no OVERLAYEXT option setting is found, the system looks for the default extension (OVL) for the AFP overlays.

- Error opening page segment: xxxxxxxx

  This message indicates the AFP print stream uses a page segment the system could not find. Note the path and file extension of the page segment specified in the error message. Make sure your AFP page segment is stored in the proper directory and contains the expected file extension. Page segments are expected to be in the same directory as the AFP print stream. The page segment extension is specified in the PAGESEGEXT option of the printer control group. If no PAGESEGEXT option setting is found, the system looks for the default extension (PSG) for the AFP overlays.

- Error opening logo: xxxxxxxx

  If you receive this message, it is likely the AFP print stream uses a page segment the system could not find. If so, you would have received an error message for the missing page segment as well. Correct the problem with the missing page segment and this error will not occur.

# MRGADD

Use this utility to search the text file produced by the MRGCHK utility and add the fonts listed there to the designated FXR file so they can be used in the library.

## Syntax

```
mrgaddw.exe /I=InputFile /X=FXRFile /N=INIFile
```

| Parameter | Description |
|---|---|
| /I=InputFile | This is the file created by the MRGCHK utility. |
| /X=FXRFile | This is the FXR file that contains the fonts you want to add. |
| /N=INIFile | (Optional) Use this parameter to specify an INI file to use. The utility uses the master resource information for the directories so it can find the resources. |

Here is an example:

```
mrgaddw.exe /I=resource.txt /X=rel103sm.fxr /N=fapcomp.ini
```

This would use the directories from the FAPCOMP.INI file to search for the fonts and resources that are in the RESOURCE.TXT file. It would add the fonts found to the REL103SM.FXR file. MRGADD would then output the RESOURCE.TXT file with all the resources that were still not found.

## See also

MRGCHK on page 178

# MRGCHK

Use this utility to check all the print files you designate and output the resources used in those files into a text file you specify. Use this utility and the MRGADD utility to get information about Documerge Metacode and AFP print streams.

This utility can detect Metacode files that have improper formats, such as if the files are not 4-byte variable blocked (MRG4 format). The utility issues an error message if such a file is detected.

## Syntax

**mrgchkw.exe /I=PrintFile /L=FileName /P=PrtType /N=INIFile /X=FXRFile /A /R**

| Parameter | Description |
|---|---|
| /I=PrintFile | Use this parameter to specify the print files you want to scan for resources. You can use wildcard, such as **.met*. |
| /L=FileName | Enter the name of the output file where you want the utility to put the resource list. The utility returns the information in a comma-delimited text file. The file is placed in the directory from which you are running the MRGCHK utility. |
| /P=PrtType | This is the printer type to be used from the INI file. MRGCHK automatically uses the FSISYS.INI so you must designate which printer information to use. |
| /N=INIFile | (Optional) You can use this parameter to specify an INI file. The default is the FSISYS.INI file. |
| /X=FXRFile | (Optional) Use this parameter to check for font names instead of just checking the font directory from the INI file. |
| /A | (Optional) Use this parameter to tell the utility to load an existing output file and append new resources to it; otherwise, the output file is overwritten. Existing resources are not duplicated so using this option gives you a growing list of resources. |
| /R | (Optional) This parameter outputs all resources regardless of whether they are found within the setup. Use this option only if you want to have a concise list of resources and are not using the output file with the MRGADD utility.<br>Omitting this parameter means only the resources that were not found in the fonts directory or in the FXR are included in the output file. |

Here is an example:

```
mrgchkw.exe /I=*.met /L=resource.txt /P=XER
```

This example will search for all MET files in the forms library using the XER PrtType from the FSISYS.INI file and will list the resources that were not found in the file named RESOURCE.TXT.

Here is another example:

```
mrgchkw.exe /I=*.met /L=resource.txt /P=XER /N=fapcomp.ini /
X=rel102sm.fxr /A /R
```

This example would find all the resources used by all the MET files in the forms directory using the XER printer found in the FAPCOMP.INI file. It would append those resources to an existing file named RESOURCE.TXT.

See also MRGADD on page 177

# OPENUSER

Use the OPENUSER utility to reset user IDs.

**NOTE:** This utility only works with xBase files. It does not work on SQL, Oracle, or DB2 databases.

## Program names

| | |
|---|---|
| Windows | OPENUW32.EXE |

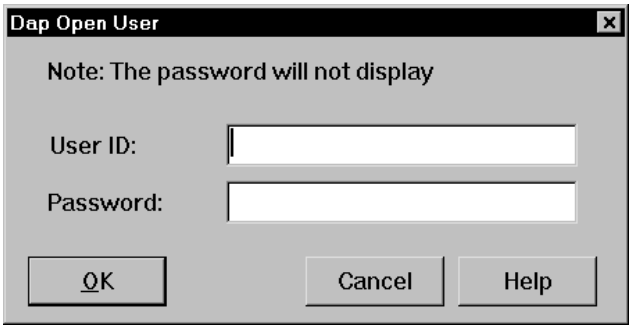
## Syntax

**OPENUW32**

## Example

Only one user can log in at a time. This prevents people on different workstations from clashing over the same resources. If the InUse flag is inadvertently left on, usually through a crash or power off, a system supervisor can run this utility to *unlock* that user's ID. Then the user can log on as usual.

This window appears when you run this utility:




The OPENUSER utility requires the following files to be in the working directory:

- FSIUSER.INI
- USERINFO.DBF
- FAPCOMP.INI (if you are using Docucreate)

# OVL2FAP

Use the OVL2FAP utility to convert an OVL (IBM AFP overlay) file into a FAP file.

## Program names

| | |
|---|---|
| Windows | OVL2FAPW.EXE |

## Syntax

**OVL2FAPW /I /X /T /O**

| Parameter | Description |
|---|---|
| /I | Enter the name of the OVL file. |
| /X | Enter the name of the font cross-reference (FXR) file. |
| /T | Enter the path where the utility should look for the IBMXREF.TBL file. |
| /O | Enter the name of the FAP file if different than input file name. |

The OVL2FAP utility requires the following files be present:

- ?.FXR
- IBMXREF.TBL (This file contains pairs of coded font names and character set names, without prefixes. It is not included with Skywire Software applications.)

The OVL2FAP utility is the opposite of the FAP2OVL utility. The OVL2FAP utility converts an AFP overlay printer resource file into a FAP file. In addition to the OVL AFP overlay file, you must also specify the font cross-reference (FXR) file which contains the AFP fonts used.

## Example

**OVL2FAP /I=AFPBAT1 /X=UFSTSM**

This command creates a file named AFPBAT1.FAP.

**NOTE:** Be sure the OVL file used is one used for AFP and not PCL. To be sure that the correct file type is being used, open the OVL file in an editor that supports hex mode. If the first byte is “1B” then the OVL is for a PCL printer. If the first byte is 5A, the correct OVL is being used.

## See also

FAP2OVL on page 78

# OVLCOMP

Use the OVLCOMP utility to create precompiled overlays for the GenPrint program.

**NOTE:** You can also perform these tasks using the Tools, Compile option in the Image Editor.

## Program names

Windows OVLCOMPW.EXE

## Syntax

**OVLCOMPW /I /X /O /L /F /U /T /W /N /P /H /VF /C /LB**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FAP file. Omit the extension. |
| /X | Enter the name of the FXR file. Omit the extension. The utility creates a FNT file if you omit the /I parameter. |
| /O | (Optional) Enter the name of the output OVL file. Omit the extension. |
| /L | Enter the name of the DLL. PCLW32 is the default. |
| /F | Enter the name of the function. PCLPrint is the default. |
| /U | Enter the type of printer. PCL is the default. |
| /T | Include to perform a print test to a file with a TST extension. |
| /W | Include to download fonts in the test, if requested. |
| /N | Include if you do not want to be notified of errors. |
| /P | Include to incorporate permanent fonts, f downloaded. |
| /H | Include to use HMI. |
| /VF | Include to get a sample print with variable fields. |
| /C | Include to use color. |
| /LB | Add this parameter to override the default logical bottom for the form being normalized. To specify a different logical bottom value, include the parameter and a value, as shown here:<br>`/LB = value` |

**NOTE:** All switches are optional and all except those noted below are case-insensitive.

Please note that:

- The /P and /W parameters are for PCL only.
- The /P parameter must be used with the /W parameter.

- The parameters used with the /F parameter are case sensitive.
- The /L parameter tells the utility which DLL to use, such as AFPPRT, PCLW32, XERW32, or PSTW32.
- The /F parameter tells the utility which print function to use, such as AFPPrint, PCLPrint, or XERPrint. These print function names are case sensitive.
- The /U parameter specifies the type of printer. This is the same as the PrtType:XXX control group in the FSISYS.INI file. The XXX represents the printer type. For example, XER for Xerox, AFP for IBM's AFP printers, PCL for the many different PCL type printers such as HP, Optra, Lexmark, and PST for PostScript printers.
- The /T parameter creates a print ready file to run a test print.
- The /W parameter downloads PCL fonts that are used in the PCL test file being created.
- The /N parameter suppresses error notification.
- The /P parameter, used with the /W parameter, restores printer resident fonts.
- The /H parameter combines many small Metacode records into one Metacode record to conserve space. This is often referred to as HMI.
- The /VF parameter creates a print-ready sample file with variable fields filled with *Xs* to indicate the length of the field. For example, a variable field eight spaces long would appears as XXXXXXXX when printed.
- The /C parameter turns on the color flag. For Xerox printers, set the PrinterInk option in the PrtType:XER control group in the FSISYS.INI file to the color you want all colored areas to be.

**NOTE:** Red, blue, or green are the only colors supported by Xerox. Blue is the default for Xerox. AFP printers do not support color. PCL printers use the colors specified in the FAP file (blue is blue, red is red, green is green, and so on). The PrinterInk option supports red, blue, green, ruby, violet, brown, gray, cardinal, royal, cyan, and magenta.

The OVLCOMP utility requires the following files:

- FSISYS.INI
- ?.FXR (you must specify the FXR as one of the parameters)

## Producing normalized files

The OVLCOMP utility can create normalized AFP and Metacode files in addition to AFP overlays or Metacode pre-compiled MET files. To create a normalized file, you must start the OVLCOMP utility with the /VF parameter to specify a print-ready file.

Modify your INI file as shown here:

```
< Control >
    Normalize = Norm
< PrtType:Norm >
    ...
```

| Option | Description |
|---|---|
| Normalize | Specify the PrtType control group you want the system to use when it normalizes the files. For instance, if you create a control group called PrtType:Norm to contain the option you want the system to use when it normalizes Metacode files, enter **Norm**.<br>If the control group you specify is missing or does not appear to be a Xerox or an AFP printer group (Class=XER, Class=AFP) you will receive errors. |

The OVLCOMP utility also now includes the /LB parameter which you can use to override the default logical bottom for the form being normalized. To specify a different logical bottom value, include the parameter and a value, as shown here:

```
/LB = value
```

**NOTE:** You can only use the /LB parameter when normalizing files.

## Example

Here is an example:

```
ovlcomp /i=chtovp3 /x=ufstsm
```

This command creates a PCL overlay file named CHTOVP3.OVL.

Here is another example, which shows how you can use this utility to compile a FAP file into a PST overlay.

1. Copy the FXR and INI files (such as REL103.FXR and FSISYS.INI) into the directory where the FAP files are stored.
2. Enter this command:

```
ovlcompw /I=my.fap /L=PSTW32 /F=PSTPrint /U=PST /X=REL103 /H /C
```

# PCL2AFP

Use the PCL2AFP utility to convert HP PCL fonts into IBM AFP fonts.

**NOTE:** You can also use the Font Manager to perform this task.

## Program names

Windows PCL2AFPW.EXE

## Syntax

**PCL2AFPW /I /C /S /D /O**

| Parameter | Description |
|---|---|
| /I | Enter the name of the PCL file. Omit the extension. |
| /C | Enter the code page, such as 437, 850, 1004, or T file name (/C=1004 is the standard in most cases.) |
| /S | Enter the dots per inch, such as 240 or 300 PCL font. The default is 300. |
| /D | Enter the dots per inch, such as 240 or 300 AFP font. |
| /O | (Optional) Enter the output file extension, such as *240* or *300*. |

You must specify the /I, /C, /D, /S parameters. The output file will default to the same name as the PCL file with either a *300* or *240* extension. This utility creates an AFP character set file.

**NOTE:** This utility uses the FSISYS.INI file to retrieve code page information.

# PCL2FAP

Use the PCL2FAP utility to convert a PCL overlay file into a FAP file.

## Program names

Windows PCL2FAPW.EXE

## Syntax

**PCL2FAPW /I /X /R /O /F /S /5**

| Parameter | Description |
|---|---|
| /I | Enter the name of the OVL file. |
| /X | Enter the name of the XRF file. |
| /R | Enter the name of the font cross-reference (FXR) file. |
| /O | Enter the name of the output FAP file and/or font cross-reference (FXR) file name. Omit the extension. |
| /F | Add to include font records in the FAP file, if created. |
| /S | Add to strip sequence numbers from field names. |
| /5 | Add to fix the date format (**5** makes it date format 1). |

**NOTE:** All switches are optional and case-insensitive.

Be sure to specify the font cross-reference (FXR) file.

# PCL2XFNT

Use the PCL2XFNT utility to convert a PCL bitmap font file into a Xerox Metacode font file.

## Program names

Windows PCL2XFTW.EXE

## Syntax

**PCL2XFTW /I /O /M /R**

| Parameter | Description |
|---|---|
| /I | Enter the name of the PCL file. |
| /O | Enter the name of the FNT file. |
| /M | Enter the maximum character number, greater than 64. |
| /R | Enter the rotation of the font (90, 180, 270, or all), no rotation is the default. |

This utility creates a Xerox Metacode font file with an FNT extension from a PCL font file.

# PNG2LOG

Use the PNG2LOG utility to convert a PNG (Portable Network Graphic) file into a logo.

**NOTE:** You can use the PNG2LOG utility or the File, Open option in Logo Manager or Image Editor to convert the PNG files. When you use the File, Open option in Image Editor, the system converts the file as it inserts the graphic onto the image.

Not all PNG formats are supported. Specifically, PNG supports a *transparency* attribute that is not supported. The system only supports opaque (non-transparent) bitmaps.

PNG also supports a variety of color bit patterns, such as - 1, 2, 4, 8, 16, 24, and 32 bits per pixel. The system does not support all of these formats, but a 3rd-party PNG library included with the system will convert the bitmaps into a pixel format the system does support.

## Program names

| | |
|---|---|
| Windows | PNG2LOG.EXE |

## Syntax

**PNG2LOG /I /O**

| Parameter | Description |
|---|---|
| /I | Enter the name of the PNG file you want to convert, with or without the extension (PNG). When converting a large number of files, simply enter /i=*. |
| /O | The name of the logo you want to create. Enter the name with or without the extension (LOG). If omitted, the utility uses the PNG file's name and the LOG extension. |

Keep in mind...

- The system converts PNG files into bitmaps when printing. It handles monochrome (1 bit), 16-color (4-bit), 256-color (8-bit), and full color (24-bit) bitmaps. However, not all printers can support these, so make sure you use bitmaps appropriate for your printer.
- PDF only supports 1-bit and 24-bit bitmaps. So, some types of PNG files may look different when you create PDF files.

# PS2PCL

Use the PS2PCL utility to convert a PostScript font into a PCL bitmap font file.

**NOTE:** You can also use Font Manager . See also TT2PCL on page 200.

## Program names

Windows PS2PCL32.EXE

## Syntax

**PS2PCL32 /I /P /D /S /O /T**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FNT file (PFB file.) |
| /P | Enter the point size, such as 6, 10, and so on. |
| /D | (Optional) Enter the dots per inch. The default is 300. |

| Parameter | Description |
|---|---|
| /S | Enter the symbol set. You can choose from:<br>DN for ISO 60: Danish/Norwegian<br>DT for desktop<br>E1 for ISO 8859/1 Latin 1<br>E2 for ISO 8859/1 Latin 2<br>E5 for ISO 8859/1 Latin 5<br>FR for ISO 69: French<br>GR for ISO 21: German<br>IT for ISO 15: Italian<br>LG for legal<br>M8 for Math-8<br>MC for Macintosh<br>MS for PS Math<br>PB for Microsoft Publishing<br>PC for PC-8 code page 437<br>PD for PC-8 D/N, code page 437N<br>PE for PC-852 Latin 2<br>PI for PI font<br>PM for PC-850 Multilingual<br>PT for PC-8 TK, code page 437T<br>R8 for Roman-8<br>SP for ISO 17: Spanish<br>SW for ISO 11: Swedish<br>TS for PS text<br>UK for ISO 4: United Kingdom<br>US for ISO 6: ASCII<br>VI for Ventura International<br>VM for Ventura Math<br>VU for Ventura US<br>W1 for Windows 3.1 Latin 1<br>WD for symbol fonts<br>WE for Windows 3.1 Latin 2<br>WO for Windows 3.0 Latin 1<br>WT for Windows 3.1 Latin 5 |
| /O | Enter the name of the output PCL file to be created. |
| /T | Enter the typeface family, such as 3=Courier, 4=Helv, or 5=TmsRmn. |

In most cases, Skywire Software recommends setting the /S parameter to W1. The W1 symbol set matches the characters used by Windows for English and several Western European languages.

For importing PostScript symbol fonts, such as Euro Sans and ITD Zapf Dingbats, which contain characters that do not adhere to a standard Windows code page, set the /S parameter to *WD*. This tells the utility that these PostScript fonts that contain characters that do not adhere to a standard Windows code page.

The PS2PCL utility converts a PostScript PFB font file into a PCL font. Because PostScript fonts are scalable and PCL fonts are bitmaps, you must specify a point size.

The PS2PCL utility loads the FSISYS.INI file and looks for the DefLib option in the FMRES control group. The path you specify in the DefLib option should indicate where a CODEPAGE.INI file can be found (usually ..*fap**mstrres**fmres**deflib*), however, the CODEPAGE.INI file is not required.

If no FMRES control group or DefLib option exists in the FSISYS.INI file, the utility uses the default path (..*mstrres**fmres**deflib*).

To run PS2PCL, you must have these additional files:

| File | Description |
|---|---|
| IFW32.DLL | Intellifont Runtime DLL for Windows in the FAP/DLL directory |
| The following files should be located in the path specified by the DefLib option in the FMRES control group in your FSISYS.INI file. | |
| IF.CFG | Intellifont Configuration file (not required with version 9.5 and later) |
| IF.FNT | Intellifont Typeface Index file |
| UIF.SS | Intellifont Symbol Set file |
| UMT.SS | MicroType Symbol Set file |
| UTT.SS | TrueType Symbol Set file |
| PLUGIN.TYQ | Intellifont Plugin and Typeface Library file |
| PLUGIN.TTF | TrueType Plugin and Typeface Library file |

# PSEG2LOG

Use the PSEG2LOG utility to convert an AFP page segment resource which contains a bitmap graphic into a LOG (logo) file.

**NOTE:** You can also perform this task using the Logo Manager. For more information on the Logo Manager, see the Docucreate User Guide.

## Program names

| | |
|---|---|
| Windows | PSEG2LGW.EXE |

## Syntax

**PSEG2LGW /I**

| Parameter | Description |
|---|---|
| /I | Enter the name of the SEG file. Omit the extension. |

# PSRESET

Use the PSRESET utility to reset a PostScript printer.

## Program names

| | |
|---|---|
| Windows | PSRESETW.EXE |

## Syntax

**PSRESETW/I /P**

| Parameter | Description |
|---|---|
| /I | Enter the printer in this format: LPTx (where x is 1, 2, 3, or 4). |
| /P | (Optional) Enter the printer definition file (.PPD). |

This utility resets a PostScript printer attached to the port specified by the /I parameter. If you do not specify a printer definition file, the utility sends a generic PostScript reset command to the printer.

# REINDEX

Use the REINDEX utility to reindex a dBase file.

## Program names

Windows REINDEXW.EXE

## Syntax

**REINDEXW /I /T /X**

| Parameter | Description |
|---|---|
| /I | Enter the name of the dBase file. |
| /T | Enter the tag name. |
| /X | Enter the index definition. |

**NOTE:** DBF files are dBase data files and MDX files are dBase index files.

Tags define *sort orders* in the index file. Tags contain information about the different ways to sort records in the data file. A tag name is a label for a tag. Reindexing is comparable to packing the file. You recalculate the sort for each tag.

If you do not have the corresponding MDX file, you must provide the tag name information if the file you are reindexing does not have a DFD file.

When reindexing a file, you have to specify all the tags and components of those tags in the correct order. Although shown on different lines for clarity, all the tags for a given file must be specified on the command line. For example...

**reindex /T=tag1 /X=field /T=tag1 /X=field1+field2**

| Type of file | Tag name | |
|---|---|---|
| WIP.DBF | /T=DOCTAG<br>/T=KEY2TAG<br>/T=KEYIDTAG<br>/T=USERTAG | /X=KEY1+KEY2+KEYID+RECTYPE<br>/X=KEY2<br>/X=KEYID<br>/X=CURRUSER |
| Help DBF files | /T=HELP_KEY | /X=HELP_KEY |
| Table DBF files | /T=TS_KEY_FIELD | /X=TABLE_NAME,ENTRY_NAME |
| XDB.DBF files | /T=BYFILE_NAME<br>/T=BYFILE_OFFSET<br>/T=BYNAME<br>/T=BYFIELD_NAME | /X=SrcFile+Parent+SrcName/<br>/X=SrcFile+Parent+STR(SrcOffset,5,0)<br>/X=SrcName<br>/X=Name |
| FDB.DBF | /T=BYNAME | /X=NAME |
| Library Manager DBF files | /T=FILEINDX<br>/T=UNIQUE_ID | /X=FILETYPE+FILESTYP+FILENAME<br>/X=UNIQUE_ID |

Example **REINDEX /I=RCPMSTR /T=POLRCPKEY /X=POLICY+RECIPCD**

input RCPMSTR.DBF

output RCPMSTR.MDX

# RENFORM

Use the RENFORM utility to rename an image (FAP file).

**NOTE:** You can also use the File, Convert, Multiple FAPs option in Docucreate to rename images or the File, Save As option in the Image Editor.

## Program names

| | |
|---|---|
| Windows | RENFORMW.EXE |

## Syntax

**RENFORMW /I /U /O /INI /D**

| Parameter | Description |
|---|---|
| /I | Enter the name of the image you want to rename (FAP file). |
| /U | Enter your user ID (required.) |
| /O | Enter the new name for the image (FAP file). |
| /INI | Enter the name of the INI file to refer to. |
| /D | Enter whether you want to delete the original image. The default is No. |

Use the /P parameter to use path information in the MasterResource control group in an INI file, such as the FSISYS.INI file. The RENFORM utility will look in the directory you specify in the FormLib entry.

## Example

**FormLib = D:\FAP\MSTRRES\FORMS**

In addition to copying the first FAP file to the second FAP file with a different name, this utility also changes the file name in the header. It also adds a comment line noting who renamed it and when it was renamed. In the comment, the utility notes the FAP file's original name.

# RTF2FAP

Use this utility to convert Rich Text Format (RTF) files into FAP files.

## Syntax

**RTF2FAP /I /X /L /F /O /M /S**

| Parameter | Description |
|---|---|
| /I | Enter the name of the RTF file you want to convert. |
| /X | Enter the name of the FXR file you want the utility to use during the conversion. |
| /L | (Optional) Include this parameter to create LOG files for all bitmap images in the input file. |
| /F | (Optional) Include this parameter and a file name to create a FOR file for use with Documaker Studio. If you omit the file name, the utility appends the FOR extension to the input file name. |
| /O | (Optional) Enter the name you want the utility to assign to the FAP file. |
| /M | Include this parameter if you want the utility to create a FAP file for each page of the RTF file. |
| /S | Use this parameter to specify how you want the utility to number pages. For instance,<br>`/S=_`<br>results in:<br>`image.fap, image_2.fap, image_3.fap`<br>If you include the /M parameter but omit the /S parameter or its mask, the utility numbers the pages in this manner:<br>`image.fap, image2.fap, image3.fap` |

When using this utility, keep in mind...

- You can import polyline or vector drawings.
- Unicode support includes paragraphs containing mixed single-byte and multi-byte characters.
- Hyphenation settings are retained.
- The utility converts bitmaps in the original document into external Documaker logo (LOG) references.

Also keep in mind these limitations:

- If an bitmap image does not specify a resolution, the utility defaults to 300 DPI. This may result in logos being sized incorrectly. The utility tries to calculate a resolution based on the output size specified by the RTF file, but this may not work in every instance.
- The *linkstyles* control word causes Microsoft Word to alter a document (fonts) according to user-specific styles held in the Normal.dot template. Since the utility cannot interpret the Normal.dot file — as it would vary according to the user — the *linkstyles* control word cannot be supported. If the imported FAP file has fonts that are significantly different from those in the original RTF file, this may be the cause.

- The utility does not support z-ordering, the layering of objects one over the other. Documents which rely on z-ordering appear differently when imported.
- The *charscalex* and *charscaley* control words cause individual characters to be scaled horizontally or vertically, respectively. The utility does not support such functionality, so these control words are not supported.
- Document layout is heavily dependent on choice of fonts. The utility tries to match the fonts in the source document, but substitution can affect the layout. The best way to avoid layout changes is to include all used fonts in your font cross-reference (FXR) file. Be aware of font copyright restrictions when doing so.
- When you use the RTF import feature to convert rich text pasted into a FAP text area, the system has no information about the margins in the FAP file. This can cause some objects to shift.
- Skywire Software applications do not include a table object, so RTF tables are converted to columnar data. If, however, the table is embedded in a column, this results in nested columns (columns within a column), which is not supported.
- Merged table cells are not supported.

# SEQ2KSDS

Use the SEQ2KSDS utility to convert a non-VSAM NAFILE or POLFILE dataset into a VSAM copy of that dataset.

## Program names

| | |
|---|---|
| MVS | see below |

## Syntax

**SEQ2KSDS**

SEQ2KSDS takes no parameters. A 4-byte key is prefixed to each record of the VSAM dataset as it is created.

You can find sample JCL for running the SEQ2KSDS utility in the SEQ2KSDX member of JCLLIB. The SEQ2KSDX member is also shown below.

```
//USERIDA  JOB   (33005),'SEQ2KSDS      ',CLASS=T,MSGCLASS=X,
//         NOTIFY=USERID
//*
//         SET HLQ='FSI.V100'    <== SET HIGH LEVEL QUALIFIER
//         SET RES='RPEX1'      <== SET RESOURCE (E.G. RPEX1, UTEX1)
//*
//         JCLLIB  ORDER=&HLQ..PROCLIB
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*        JOB PERFORMS 2 STEPS :
//*
//*        1.  DELETES / RE-DEFINES OUTFILE KSDS.
//*        2.  RUNS SEQ2KSDS PROGRAM TO COPY SEQUENTIAL FILE TO
//*            VSAM KSDS.
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*
//IDCAMS  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD  *

  DELETE FSI.V100.RPEX1.GENDATA.NAFILE.KSDS CLUSTER

  DEFINE CLUSTER(NAME(FSI.V100.RPEX1.GENDATA.NAFILE.KSDS) +
           CYL(5 1)      +
           KEY(4 0)      +
           REUSE )       +
         DATA(NAME(FSI.V100.RPEX1.GENDATA.NAFILE.KSDS.DATA) +
           RECORDSIZE(2048 2048) )                        +
         INDEX(NAME(FSI.V100.RPEX1.GENDATA.NAFILE.KSDS.INDEX))

  IF LASTCC = 00 THEN SET MAXCC = 00

//*
//SEQ2KSDS EXEC PGM=SEQ2KSDS
//STEPLIB  DD  DISP=SHR,DSN=&HLQ..LINKLIB
//         DD  DISP=SHR,DSN=SYS1.SCEERUN
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//INFILE   DD DSN=&HLQ..&RES..GENDATA.NAFILE,DISP=SHR
//OUTFILE  DD DSN=&HLQ..&RES..GENDATA.NAFILE.KSDS,DISP=SHR
```

# TT2PCL

Use the TT2PCL utility to convert TrueType fonts into PCL bitmap fonts.

**NOTE:** See also PS2PCL on page 189.

## Program names

| | |
|---|---|
| Windows | TT2PCL32.EXE |

## Syntax

**TT2PCL32 /I /P /S /D /O /T**

| Parameter | Description |
|---|---|
| /I | Enter the name of the TTF file (True Type font). |
| /P | Enter the font point size, such as 6, 10, and so on. |
| /D | (Optional) Enter the dots per inch, 240 or 300. The default is 300. |

| Parameter | Description |
|---|---|
| /S | Enter the symbol set. Choose from these options:<br>DN for ISO 60: Danish/Norwegian<br>DT for desktop<br>E1 for ISO 8859/1 Latin 1<br>E2 for ISO 8859/1 Latin 2<br>E5 for ISO 8859/1 Latin 5<br>FR for ISO 69: French<br>GR for ISO 21: German<br>IT for ISO 15: Italian<br>LG for legal<br>MC for Macintosh<br>PC for PC-8 Codepage 437<br>PD for PC-8 D/N, Codepage 437N<br>PE for PC-852 Latin 2<br>PM for PC-850 Multilingual<br>PT for PC-8 TK, Codepage 437T<br>R8 for Roman-8<br>SP for ISO 17: Spanish<br>SW for ISO 11: Swedish<br>TS for PS text<br>UK for ISO 4: United Kingdom<br>US for ISO 6: ASCII<br>VI for Ventura International<br>VU for Ventura US<br>W1 for Windows 3.1 Latin 1<br>WE for Windows 3.1 Latin 2<br>WO for Windows 3.0 Latin 1<br>WT for Windows 3.1 Latin 5 |
| /O | Enter the name of the PCL file to be created. |
| /T | Enter the typeface family (3=Courier, 4=Helv, 5=Times Roman, and so on). |

**NOTE:** In most cases, Skywire Software recommends setting the /S parameter to W1. The W1 symbol set matches the characters used by Windows for English and several Western European languages.

This utility loads the FSISYS.INI file and looks for the DefLib option in the FMRES control group. The path you specify in the DefLib option tells the utility where to find the CODEPAGE.INI file (usually ..*fap**mstrres**fmres**deflib*), however, the utility does not require the CODEPAGE.INI file.

If there is no FMRES control group or no DefLib option, the utility uses the default path (..*mstrres**fmres**deflib*).

To run TT2PCL, you must have these additional files:

| File name | Description |
|---|---|
| IF.CFG | Intellifont Configuration file (not required with version 9.5 and later) |
| IF.FNT | Intellifont Typeface Index file |
| UIF.SS | Intellifont Symbol Set file |
| UMT.SS | MicroType Symbol Set file |
| UTT.SS | TrueType Symbol Set file |
| PLUGIN.TYQ | Intellifont Plugin and Typeface Library file |
| PLUGIN.TTF | TrueType Plugin and Typeface Library file |

When you install the system, these files part of the TrueType font installation in the ..\mstrres\fmres\deflib directory. These files should be in the path specified by the DefLib option of the FMRES control group in your FSISYS.INI file.

# TRANSLAT

Use the TRANSLAT utility to create ERRFILE.DAT and LOGFILE.DAT files from the MSGFILE.DAT file. This utility uses the TRANSLAT.INI file as a resource.

Normally, the system creates the ERRFILE.DAT and LOGFILE.DAT files for you during the normal GenTrn, GenData, and GenPrint processing steps. If, however, you have used the ImmediateTranslate option in the Control INI file control group to turn off the automatic conversion of the MSGFILE.DAT file, you must use the TRANSLAT utility to convert the MSGFILE.DAT file into the ERRFILE.DAT and LOGFILE.DAT files.

**NOTE:** Typically, you would turn off the creation of the ERRFILE.DAT file to maximize performance. You can also use this utility to translate error message into different languages or to substitute your own custom messages.

## Program names

| | |
|---|---|
| MVS | TRANSLAT |
| Windows | TRANSW32.EXE |

## Syntax

**TRANSLAT /INI /MSG /TINI**

| Parameter | Description |
|---|---|
| /INI | Enter the name of the INI file you want the utility to read. The default is FSIUSER.INI. |
| /MSG | Enter the name of the message file (MSGFILE.DAT) you want the utility to use. |
| /TINI | Enter the name and path of the INI file you want the utility to use, typically TRANSLAT.INI. |

You can omit all of these command line parameters and simply specify the information in your FSIUSER.INI or FSISYS.INI files.

## TRANSLAT.INI file

The TRANSLAT.INI file is used to customize or localize log and error messages in the system. For instance, if you want the system's error messages to appear in a language other than US English, you can use this INI file to translate the error messages.

All messages must have a unique number. The error message section contains a complete list of numbers already assigned in the system. If the message is only used in the log message section, the number should appear in the error message section, followed by a comment stating that it is only used in the log section. This table shows how the numbers are assigned:

| Range | Description |
|---|---|
| 100-499999 | Reserved general messages |
| 10000-10999 | Reserved for RULLIB |
| 11000-11999 | Reserved for GENLIB |
| 12000-12999 | Reserved for RPLIB |

| Range | Description |
|---|---|
| 13000-13999 | Reserved for RCBLIB |
| 14000-14999 | Reserved for A2WBLIB |
| 15000-15999 | Reserved for the GenTrn program |
| 16000-16999 | Reserved for the GenData program |
| 17000-17999 | Reserved for the GenPrint program |
| 18000-18999 | Reserved for the GenArc program |
| 19000-19999 | Reserved for the GenWip program |
| 20000-20999 | Reserved for CUSLIB (the base system libraries) |
| 500000-999999 | Custom messages |

**NOTE:** Each library has its own section within the ERROR MESSAGE SECTION, complete with a range of valid numbers.

When you add message numbers and text to a library, use the next available message number in the section.

The TRANSLAT.INI file does not contain control groups.

## UP2LOW

Use the up2low utility to convert all file names in a directory to lowercase names.

### Program names

| | |
|---|---|
| UNIX | up2low.exe |

This utility will only convert files in the directory in which you run it. If you have additional subdirectories in which there are additional files to convert, go to those subdirectories and run the utility again.

**NOTE:** The up2low utility is a UNIX script and should always be entered in lowercase. The up2low utility is version independent.

# VB2AFP

Use this utility to convert files written in variable block record format.

For instance, Documerge creates print streams and printer resource files which have variable block record format. You can use this utility to remove that formatting and create a file which you can use as an AFP print stream.

AFP or Metacode printer resource files (AFP page segments, AFP overlays, Metacode FRMs, and so on) may also be in variable block format. Other utilities expects these files to be in their native format as does the printer. You can use this utility to remove the variable block formatting from either AFP or Metacode printer resource files.

## Program names

| | |
|---|---|
| Windows | VB2AFP.EXE |

## Syntax

**VB2AFP /I /O**

| Parameter | Description |
|---|---|
| /I | Enter the name of the input file (in variable block format). |
| /O | Enter the name of the output file—without variable block format. |

**NOTE:** To convert variable block files into BARR format, see VB2BARR on page 207.

# VB2BARR

Use this utility to convert files written in variable block record format.

For instance, Documerge creates print streams and printer resource files which have variable block record format. You can use this utility to remove that formatting and create a file which you can use as a Metacode print stream with BARR record formatting.

## Program names

Windows VB2BARR.EXE

## Syntax

**VB2BARR /I /O /W**

| Parameter | Description |
|---|---|
| /I | Enter the name of the input file (in variable block format). |
| /O | Enter the name of the output file—in BARR record format. |
| /W | Include this parameter to tell the utility to write the output file in BARRWORD record format. |

**NOTE:** To convert variable block files into AFP format, see VB2AFP on page 206.

# VRF2EXP

Use this utility to convert a Documerge VRF file into various types of output files, including:

- Documaker Workstation-style import files
- Extract files
- INI files

## Program names

Windows VRF2EXPW.EXE

## Syntax

**VRF2EXP /I=vrffile /O=dapfile /F=format /INI=inifile**

| Parameter | Description |
|---|---|
| /I | Enter the name of the file you want to convert. If omit the extension, the system defaults to *VRF* though the input file can have any extension. If the input file does not have an extension, add a period to the end of the file name. |
| /O | (Optional) Enter a file name for the output DAP file. If you omit this parameter, the input name will be used with the extension of *DAP*. Since a Documaker Workstation-style import file typically has an extension of *DAT*, you may want to use this parameter and be sure to specify the extension. |
| /F | Choose from these types of output formats:<br>0 - Documaker Workstation-style import file<br>1 - a flat extract file<br>2 - INI file<br>Examples of these formats are at the end of this topic. The default is zero (0) for a Documaker Workstation-style import file. |
| /INI | (Optional) If you chose to create a Documaker Workstation-style import file, the utility looks for an INI file. By default, the utility looks for a file named VRF2EXP.INI in the current directory. If not found, the program will try to continue on without it. |
| /D | Include this parameter to run the utility in debug mode. For more information, see Using debug mode on page 209. |

**NOTE:** The INI file parameter has special uses. Here is an example:

```
VRF2EXP /I=MAPPER
```

This example will input a file called MAPPER.VRF, assume a Documaker Workstation-style import format, load the VRF2EXP.INI file, and then output an import file named MAPPER.DAP.

The VRF2EXP utility sets the scope of the field data to *Global*. Therefore, to import the field data during Entry, the FAP file's variable field must also have a scope of *Global*.

## Using debug mode

If the utility does not appear to be working correctly, you may have a problem with your INI file. You can find out by running in debug mode to see what the utility thinks the INI options are.

Here is an example of the output without debug mode:

```
--- Docucorp VRF2EXP Utility Program (C) ---
---  Convert Docucorp VRF to DAP Format  ---
Using standard import format for DAP output.
Reading VRF File - MAPPER.VRF
Creating DAP File - MAPPER.DAP
Using INI File - VRF2EXP.INI
Merge sets found=1
```

Here is an example of the output when running in debug mode:

```
--- Docucorp VRF2EXP Utility Program (C) ---
---  Convert Docucorp VRF to DAP Format  ---
Run in debug mode
Using standard import format for DAP output.
Reading VRF File - MAPPER.VRF
Creating DAP File - MAPPER.DAP
<VRF2ImportHeaderTags> DESCRIPTION =
<VRF2ImportHeaderTags> KEY1 = "PMI"
<VRF2ImportHeaderTags> KEY2 = "LB1"
<VRF2ImportHeaderTags> KEYID = INSURED.NAME
<VRF2ImportHeaderTags> STATUSCODE = "W"
<VRF2ImportHeaderTags> TRANSCODE = "NB"
Using INI File - VRF2EXP.INI
VRF2EXP context:
<VRF2ImportHeaderTags> DESCRIPTION =
<VRF2ImportHeaderTags> KEY1 = "PMI"
<VRF2ImportHeaderTags> KEY2 = "LB1"
<VRF2ImportHeaderTags> KEYID = INSURED.NAME
<VRF2ImportHeaderTags> STATUSCODE = "W"
<VRF2ImportHeaderTags> TRANSCODE = "NB"
End of VRF2EXP context
Using these INI options:
    KEY1 = "PMI"
    KEY2 = "LB1"
    KEYID = INSURED.NAME
    TRANSCODE = "NB"
    STATUSCODE = "W"
    DESCRIPTION = DESCRIPTION
Merge sets found=1
```

## Setting up the VRF2EXP.INI file

The VRF2EXP.INI file supplies key fields needed by a Documaker Workstation-style import header file. There are several ways to provide these values:

- Default in a constant value. Do this by supplying a constant value enclosed in quotation marks.
- Variable data from tags. Enable this by supplying the VRF tag-name (not in quotation marks).

Here is an example:

```
< VRF2ImportHeaderTags >
    KEY1 = "PMI"
    KEY2 = "LB1"
    KEYID = INSURED.NAME
    TRANSCODE = "NB"
    STATUSCODE = "W"
    DESCRIPTION =
```

For help reading the FAP file that is produced, add these INI options to have a header and footer that separates each transaction. Here is an example:

```
< VRF2ImportTransaction >
    Header = ************Begin Transaction************
    Footer = *************End Transaction*************
```

These settings produce the following output:

```
************Begin Transaction************
;PMI;LB1;;NB;W
$TAG\GO$OD
DMG.MERGESET.ID\124
DMG.SRC.JBGRP1\
DMG.SRC.JBGRP2\
\NA=\PMI;LB1;DOL.MET;
*************End Transaction*************
```

To see the print options for a VRF file, use this INI options:

```
< VRF2ImportData >
    ShowPrintOptions = Yes
```

The default is to not print them out.

### Format o - Documaker Workstation-style import file

Here is an example of the format used for a Documaker Workstation-style import file:

```
;PMI;LB1;021999BMW;NB;W
1A\your second
1B\OUTPATIENT PROTECTION
1C\applying for
1D\ALL NEW
1G\On behalf of Physicians Mutual and AAA Chicago Motor Club.
Welcome!
1H\Insurance
2B\Your insurance policy, which has been in force since September 12,
1999 is enclosed.
2E\Insurance
3A\you and your family
3B\$250.00
3D\$1,000.00
3E\$5,000.00
5\$500.00
6A\you or a covered family member's
6B\you or a covered family member are
7\have
AAA\AAA Chicago Motor Club.
AMOUNT\500.50
CC301\print
INSURED.NAME\021999BMW
```

```
ISSUE\SHORT
ISSUEDATE\September 12, 1999
MONEY\REGULAR
PLAN\FAMILY
POLICY.TYPE\LIFE
SPO\2
STATE\TX
\NA=\PMI;LB1;N150-72A;
\NA=N150HDR\<INSURED>
\NA=N150-72A\<INSURED>
\NA=N150-72B\<INSURED>
\NA=\PMI;LB1;Coverage Outline;
\NA=PTYPE\<INSURED>
\NA=PPOLNO\<INSURED>
\NA=PCONF\<INSURED>
\NA=PTYPE\<INSURED>
\NA=PTYPE\<INSURED>
\NA=PACC\<INSURED>
\NA=PEXC\<INSURED>
\NA=PPREM\<INSURED>
\NA=PNUM\<INSURED>
\NA=PFORM\<INSURED>
\NA=PLAST\<INSURED>
```

### Format 1 - Extract file

The format of a flat extract file is shown here:

```
Pos 1 to  6Numeric relative merge-set within the VRF file
Pos7Blank
Pos8Record-type flag
M = Begins a merge set
T = A merge tag/data record
E = Ends a merge set
Pos9Blank
Pos10 to 49Merge tag name
Pos50Blank
Pos51 to 56Numeric length of data
Pos57Blank
Pos58Data (any length)
<EOR>Cr/lf
```

Each merge set found in the VRF file creates a block of records. All tags found are included in the output. Here is an example:

```
000001 M
000001 T INSURED.NAME                              000010 021999BMW
000001 T POLICY.TYPE                               000004 LIFE
000001 T STATE                                     000002 TX
000001 T ISSUE                                     000005 SHORT
000001 T DMG.FLST.INSURED                          000588 N150-
12A\N150HDR...
000001 T DMG.OPT.INSURED                          000784 SIM SEP MAI
STA ...
000001 E
000002 M
000002 T INSURED.NAME                              000009 Bob Adams
...etc
```

### Format 2 - INI file

Use this format to see the internal organization used to create a Documaker Workstation-style import file. This can be useful for debugging purposes. Here is an example:

```
< DMG.FLST >
    Form          = N150-72A\N150HDR
    Form          = N150-72A\N150-72A
    Form          = N150-72A\N150-72B
    Form          = Coverage Outline\PTYPE
    Form          = Coverage Outline\PPOLNO
    Form          = Coverage Outline\PCONF
    Form          = Coverage Outline\PTYPE
    Form          = Coverage Outline\PTYPE
    Form          = Coverage Outline\PACC
    Form          = Coverage Outline\PEXC
    Form          = Coverage Outline\PPREM
    Form          = Coverage Outline\PNUM
    Form          = Coverage Outline\PFORM
    Form          = Coverage Outline\PLAST
< Form:Coverage Outline\PACC >
    RECIP         = INSURED
< Form:Coverage Outline\PCONF >
    RECIP         = INSURED
< Form:Coverage Outline\PEXC >
    RECIP         = INSURED
< Form:Coverage Outline\PFORM >
    RECIP         = INSURED
< Form:Coverage Outline\PLAST >
    RECIP         = INSURED
< Form:Coverage Outline\PNUM >
    RECIP         = INSURED
< Form:Coverage Outline\PPOLNO >
    RECIP         = INSURED
< Form:Coverage Outline\PPREM >
    RECIP         = INSURED
< Form:Coverage Outline\PTYPE >
    RECIP         = INSURED
< Form:N150-72A\N150-72A >
    RECIP         = INSURED
< Form:N150-72A\N150-72B >
    RECIP         = INSURED
< Form:N150-72A\N150HDR >
    RECIP         = INSURED
< VRF:1 >
    1A            = your second
    1B            = OUTPATIENT PROTECTION
    1C            = applying for
    1D            = ALL NEW
    1G            = On behalf of Physicians Mutual and AAA Chicago
Motor Club.  Welcome!
    1H            = Insurance
    2B            = Your insurance policy, which has been in force
since September 12,
    2E            = Insurance
    3A            = you and your family
    3B            = $250.00
```

```
3D          = $1,000.00
3E          = $5,000.00
5           = $500.00
6A          = you or a covered family member's
6B          = you or a covered family member are
7           = have
AAA         = AAA Chicago Motor Club.
AMOUNT      = 500.50
CC301       = print
```

# XERDNLD

Use the XERDNLD utility to convert Xerox Metacode printer resources, such as FNT files, IMG files, or FRM files, into downloadable files.

## Program names

| | |
|---|---|
| Windows | XERDNLDW.EXE |
| MVS | See the Documaker Server Installation Guide |

## Syntax

**XERDNLDW /I /O /P**

| Parameter | Description |
|---|---|
| /I | Enter the file name. Include the extension, such as FRM, FNT, or IMG. |
| /O | (Optional) Enter the name of the output DAT file. |
| /P | Enter the INI file PrtType to use, such as *XER*. |

You can substitute dashes (-) for backslashes (/) with these parameters. You can also use wildcards to select multiple files or specify a single file. For instance, on Windows machines, adding this parameter

```
/i=*.FNT
```

selects all files in current directory with extension FNT. Include a file extension when using wildcards, otherwise the XERDNLD utility will try to process all of the files in the directory.

**NOTE:** You must have the FSISYS.INI file to run the XERDNLD utility.

This utility lets you get resources to a Xerox printer if you do not have other software programs to accomplish this task. The utility accepts as input a Xerox Metacode resource, such as a font, image, or FRM file, encapsulates it with location-specific DJDE information and writes the resource back out as a .DAT file. You can then send the .DAT file to the Xerox printer as if it were a print stream. The DJDE information at the beginning of the .DAT file will cause the Xerox printer to store the resource on the printer's local disk device.

## Xerox 4235 printers

When you use the XERDNLD utility to download Xerox resources to a 4235 printer, you must set the printer to use a JDL called 9EBC, and a JDE called LAND. You can set the JDL and JDE from the Document Formatting menu under the XPPM option. In addition, you must set the EBCDIC Parallel Meta / GHO Enabler - IB option to *disabled*, via the User Services/Printer Options menu.

After you set these options, set the 4235 in Print Mode and send your .DAT files. After you finish sending the .DAT files, remember to reset your JDL, JDE, and EBCDIC Parallel Meta / GHO Enabler - IB options to resume normal print settings.

# XFNT2PCL

Use the XFNT2PCL utility to convert a Xerox Metacode font into a PCL bitmap font.

**NOTE:** You can also perform this task using the Font Manager.

## Program names

| | |
|---|---|
| Windows | XFNT2PCW.EXE |

## Syntax

**XFNT2PCW /I**

| Parameter | Description |
|---|---|
| /I | Enter the name of the FNT file. |

The utility creates a converted PCL font file. The output file will have the extension PCL.

The XFNT2PCL utility is the reverse of the PCL2XFNT utility. The XFNT2PCL utility creates a PCL font file from a Xerox FNT font file.

# Index

## A

## B



## C

## D

## E



## F

## G

## H



## I



## J



## K



## L

## M



## N



## O

# P

## Q



## R



## S



## T



## U

# V



# W



# X