opentext™

OpenText™ Embedded Output Transformation Engine

z/OS User Guide

This document provides information needed to develop projects in Output Transformation Engine to interact with an IBM mainframe operating system.

VDTOTS240200-UZE-EN-1

OpenText™ Embedded Output Transformation Engine
z/OS User Guide
VDTOTS240200-UZE-EN-1
Rev.: 2024-Apr-16

This documentation has been created for OpenText™ Embedded Output Transformation Engine CE 24.2.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

OpenText Embedded Output Transformation Engine z/OS User Guide

1.1 Introduction

This user guide includes the information needed to develop projects to interact with the IBM mainframe operating system. This interaction occurs via the Output Transformation Engine software.

Output Transformation Engine interacts with MVS as well as z/OS USS -- the Unix layer above MVS. Output Transformation Engine is installed on USS and must run from USS. The user can choose to do things such as:

- Create a JCL that calls the USS script to run the engine.
- Read/write to MVS data sets instead of USS file paths.

1.2 Audience

This guide is intended for the configuration and deployment of Output Transformation Engine projects on the IBM mainframe operating system.

1.3 Information in this Guide

This guide discusses the steps required to write and deploy a z/OS Output Transformation Engine project to the IBM mainframe operating system and how to develop those projects to use specific MVS configurations.

1.4 Collected Files

Projects are typically developed on Windows and later deployed to the z/OS production server. The files transferred to the production machine are generated by Output Transformation Engine. On Windows, the default location of the program project files and supporting resources will normally be:

```
C:\OpenText\OTS\<version number>
```

During the installation process you can define any installation location. Throughout the remainder of this document, the installation location will be indicated as `<install directory>`.

1.5 Generated Files

Output Transformation Designer creates a Samples folder and installs sample projects in:

```
<install_home>\initialfiles\common\_sample\OutputTransformation\
applications\sample
```

You can explore Output Transformation Designer by loading and reviewing these sample project files. You may also model your own project file and scripts on the samples we have provided.

Files generated by Output Transformation Designer are kept in either a single workspace or a workspace for each project. A workspace is a network location or mapped drive/directory defined by the user.

Applications equate to a sub-directory of a workspace. Any number of applications can be part of a given workspace. Applications hold resource files, test input files, and any MFCT files that are specific to this project. The project files are also kept there.

Output Transformation Engine uses at the project level:

- d2eproj - project file
- d2emfct - font correlation table
- d2efontenv - font environment map

Output Transformation Engine uses at the system level:

- d2esys
- licence
- ICU tables (AFP encoding tables)

1.6 Deploying to z/OS

Before preparing your project files for transfer and deployment to z/OS, you need to ensure three things about all files that will be part of the z/OS package:

- File names for resources stored in MVS datasets cannot begin with a numeric digit. See Section 1.6.2.2: “Xerox Centralized Object Resources - FRMS/FNTs/LGOs/IMGs/PDEs” on page 8. Filenames for files stored on USS file systems can use any Unix naming convention.
- File names for resources stored in MVS datasets cannot contain underscore characters. Often PC font names include underscores. Rename any applicable file or font names. Filenames for files stored on USS file systems can use any Unix naming convention.
- z/OS file names are a maximum of 8 characters (1 to 8 characters). Any file or resource names that exceed 8 characters must be renamed.

In order to deploy Output Transformation Engine to z/OS, you must use the pre-packaged engine file known as the Embedded Output Transformation Engine ZIP file to assist with manual deployments to another machine. The engine's file name is `EOTE-<version>.zip` and is shipped in the `subproducts` directory along with your installation package.

1.6.1 Transferring Output Transformation Engine Project Files to z/OS

The project file, MFCT file, and font environment map file are XML files that can only be transferred to the USS file system on the z/OS production machine through a file transfer process such as FTP. On PCs and workstations, these files are in UTF-8.

Note: UTF-8 is a multi-byte Unicode encoding where codepoints 0 through 128 are identical to those in the ASCII encoding scheme.

Using FTP, there are two methods to properly transfer the UTF-8/ASCII formatted XML files from Windows to the z/OS mainframe.

- Leave the encoding as UTF-8 and FTP the files in BINARY mode from the Windows platform to the mainframe.

If production is done on the z/OS IBM mainframe, these files are not required to be converted to EBCDIC when they are transferred. This will keep the native ASCII bytes as is; Output Transformation Engine's XML parser will process them as ASCII/UTF-8 data. If using this method, if you need to view or modify any files, it must be done using an ASCII vi editor.

- Manually change the encoding=”UTF-8” statement to be encoding=”ebcdic-cp-us” and FTP the file in ASCII/TEXT mode.

In ASCII/TEXT mode, the FTP process automatically converts ASCII bytes into EBCDIC bytes for z/OS mainframe conversion. The encoding specified (ebcdic-cp-us) will be correct and you will be able to view and modify the XML files as normal text files from the mainframe console. This approach will only work if all of the values in the XML are within the ASCII range (0-128). If this is not the case, binary mode is recommended.

If you choose this method and you need to make changes to the file in Output Transformation Designer on any non-EBCDIC platform later, you need to first change the encoding back to UTF-8.

Note: It is recommended that references to file locations in the script be set using job variables in the d2eproj or d2esys file.

There is no need to change the direction of the slashes in the file paths between Windows and USS environments. Output Transformation Engine understands the paths with the slashes in either directions, so it takes care of that automatically.

1.6.2 Transferring Print Resources

Some print resource files need to be modified for proper functioning on the mainframe. This section provides information on which resources need changes, and how to complete the changes.

1.6.2.1 AFPDS

All AFPDS resources being moved from the PC to the mainframe need to be reblocked. AFP resources have internal length fields and the final record blocking needs to reflect that. AFRREBLK, a free utility from IBM, can be used to do the reblocking. When viewing the file on the mainframe, with carriage control view set to ON, 0x5A is required in the first position.

1.6.2.2 Xerox Centralized Object Resources - FRMS/FNTs/LGOs/IMGs/PDEs

Upload Xerox resources to a fixed LRECL 128 partitioned data set. The resources must be uploaded in binary mode.

Xerox resource names are a maximum of six characters. They may start with numbers, which violates the z/OS naming system. If the Xerox resources used in your projects begin with a number, they must be renamed to include an alphabetic prefix. Use the prefixes in the following examples:

- FN = fonts
- FR = forms
- IM = images
- JS = JSL

For instance, a font called 1141BP on the printer should be loaded to z/OS as FN1141BP. In your Output Transformation Engine project, you would reference the MVS data set from a DDName like this:

```
<filedeflist name=”FdFnt”>
<filedef directives=”{nocrlf}”
     location=”DSN”DD”METARES(FN{0})”
     cache=”Job”/>
  </filedeflest>
```

You can also reference a data set like this:

```
<filedeflist name=”FdFnt”>
<filedef directives=”{nocrlf}”location= ”DSN:SOME.DATA.SET(FN{0})”cache=”Job”/>
</filedeflist>
```

If a different prefix was used, the sample should be changed to match the actual name on the system.

1.6.2.3 Xerox Centralized Source Resources - JSL

Upload as a text file. Translate and remove CRLF.

1.6.2.4 Moving PDF fonts

If shareware or custom PDF fonts are used, they must be uploaded in binary mode.

AFM resources need to be uploaded to a LRECL 255, RECFM VB partitioned data set.

PFB resources need to be uploaded to a LRECL 27816, RECFM VB partitioned data set.

Note: If resources contain underscores, the names must be changed prior to uploading to the mainframe. If you change resource names, remember to also update the font table as it must map to the MVS dataset member exactly.

1.6.2.5 Tuning Requirements for z/OS

You need to enable the TuningOverride parameter when running Output Transformation Engine on z/OS mainframe, version 1.8 or higher, as the OS no longer supports multiple threads writing to the same dataset.

To enable Tuning Override, open the system configuration file (d2esys) used for the application and select the TuningOverride checkbox.

1.7 Accessing Data Sets using DSN/DDName

1.7.1 Reading the DSNs and DDNAMES

DDNAME referencing existing DSN

```
jcl-> //MYDDNAME DD DSN=XENOS.VISION.AFPINPUT,DISP=SHR
d2eproj-> location="DSN:DD:MYDDNAME"
```

DDNAME referencing single PDS and the member specified dynamically

```
jcl-> //MYDDNAME DD DSN=XENOS.VISION.AFPFONTS,DISP=SHR
d2eproj-> location="DSN:DD:MYDDNAME({0})"
```

DDNAME referencing single PDS and the member specified in the JCL

```
jcl -> //INPDS DD DSN=XENOS.VISION.INPUT(FILE),DISP=SHR
d2eproj -> location="DSN:DD:INPDS"
```

DDNAME referencing chained PDSs

This is not supported due to a limitation in IBM's JRIO. The recommended approach to referencing a chained PDS is to code multiple file definitions with different DDNames and list each DDName, one per PDS, in the JCL.

```
jcl ->
//METARES  DD DSN=XENOS.VISION.METARES.FNT1,DISP=SHR
```

```
//FONTPDS2 DD DSN=XENOS.VISION.METARES.FNT2,DISP=SHR
//FONTPDS3 DD DSN=XENOS.VISION.METARES.FNT3,DISP=SHR
<filedeflist name="FdFnt">
<filedef directives="{nocrlf}"
     location="DSN:DD:METARES(FN{0})"
     cache="job"/>
<filedef directives="{nocrlf}"
     location="DSN:DD:FONTPDS2(FN{0})"
     cache="job"/>
<filedef directives="{nocrlf}"
     location="DSN:DD:FONTPDS3(FN{0})"
     cache="job"/>
</filedeflest>
```

DDNME DD *(reading from lines contained in the JCL)

This functionality is currently not supported.

1.7.2 Writing the DSNs and DDNAMES

DDNAME DD SYSOUT *

The output will be in the JCL sysout listing as per the message class.

```
jcl-> //MYDDNAME DD SYSOUT=*
d2eproj-> location="DSN:DD:MYDDNAME"
```

DDNAME referencing existing DSN

```
jcl-> //MYDDNAME DD DSN=XENOS.VISION.PDFOUT,DISP=SHR
d2eproj-> location="DSN:DD:MYDDNAME"
```

DDNAME referencing newly created DSN

This is the same as an existing DSN because the JCL will create it before the JVM starts.

```
jcl->//OUTSEQ DD DSN=XENOS.META.OUT.SEQ,
     //       DISP=(NEW,CATLG,CATLG),UNIT=3390,
     //       SPACE=(TRK,(10,10),RLSE),VOL=SER=XENU07,
     //       RECFM=VB,BLKSIZE=0,LRECL=255,DSORG=PS
d2eproj-> location="DSN:DD:OUTSEQ"
```

DDNAME referencing PDS and the member specified dynamically

```
jcl-> //MYDDNAME DD DSN=XENOS.VISION.RESOUT,DISP=SHR
d2eproj-> location="DSN:DD:MYDDNAME({0})"
```

1.7.3 TIFF Parsing Concerns for DSNs

The TIFF parser is limited in how it handles TIFFs when the input stream is not seekable. TIFF files with backward references are not supported within data sets.

1.7.4 Best IO Directive for DSNs and DDNames

If the DSN exists, it is best to use the {nocrlf} IO Directive.

If you need to create a DSN, use either {Fxx}, {MRDWxxx}, or {MBDW,xxx,xxx}.

1.7.4.1 {Fxxx}

The {Fxxx} IO Directive indicates that the records are all the same length, as indicated by xxx. This is equivalent to an MVS Fixed data set, recfm=F.

1.7.4.2 {MRDWxxx}

The {MRDWxxx} IO Directive is only used to read data sets on MVS. This IO directive indicates that the data set has variable length records with a maximum length of xxx bytes.

1.7.4.3 {MBDWxxx,yyy}

The {MBDWxxx,yyy} IO Directive is only used to read data sets on MVS. This IO directive indicates that the records have variable length records of up to xxx bytes and are grouped into blocks of up to yyy bytes.

Note: A block can contain one or more records, but records cannot be split between blocks.

For more information about these and other IO directives and their uses, see the IO Directives section of the *OpenText Embedded Output Transformation Engine User Guide*.

1.8 Using Output Transformation Engine with Partitioned Data Sets

The following sections describe how to set up Output Transformation Engine to read and write to partitioned data sets (PDS) on z/OS.

To enable Output Transformation Engine to read and write using PDS on z/OS, you need to be familiar with the issues discussed in the*OpenText Embedded Output Transformation Engine User Guide* section “Transferring Configuration Files”.

1.8.1 Reading Resources from PDS

To set up Output Transformation Engine to read resources from PDS:

- Make sure all resources have been stored in PDS as members.

If you would like to put all types of resources into one PDS, make sure every resource has a unique name.

- In the project file, specify the resources as follows:

```
<filedeflist name="FdAfpfonts">
<filedef directives="{nocrlf}"
location="DSN:XENOS.AFP.VISION.SAMPLE.RES.FONTS({0})"
cache="job"/>
</filedeflist>
```

- If the AFP input file is stored as a sequential data set, you can specify it in the project as follows:

```
<filedeflist name="FdInput">
<filedef directives="{afp}"
location="DSN:XENOS.AFP.VISION.SAMPLE.AFP"
cache="job"/>
</filedeflist>
```

1.8.2 Writing Resources to PDS

To set up Output Transformation Engine to write resources to PDS:

- Create a PDS on the mainframe, making sure the record length is big enough to hold the maximum record in all of the resources.

Note: Because the current version of IBM Java for OS/390 (the IBM SDK for z/OS, V1.4) only supports FB and VB format, you can create only these two types of PDS

Example: XENOS.AFP.VISION.PAYROLL.RES

- In the project, specify the export resource parameters as below:

```
<resGroupOption exportOverlays="true"
     exportPageSegs="true"
     exportFormDefs="true">
<filedeflist name="FdExportOverlays">
<filedef directives="{nocrlf}"
     location="DSN:XENOS.AFP.VISION.PAYROLL.RES({0})"
     cache="Job"/>
</filedeflist>
<filedeflist name="FdExportPageSegs">
<filedef directives="{nocrlf}"
     location="DSN:XENOS.AFP.VISION.PAYROLL.RES{0})"
     cache="Job"/>
</filedeflist>
<filedeflist name="FdExportFormDefs">
<filedef directives="{nocrlf}"
```

```
     location="DSN:XENOS.AFP.VISION.PAYROLL.RES({O})"
     cache="Job"/>
</filedeflist>
</resGroupOption>
```

You may choose which type of resource you want to add to the PDS by turning exportXXX to true or false.

Chapter 2

Appendix

2.1 Sample JCL

The following JCL illustrates a typical METACODE to PDF configuration. The following is the JCL in its entirety.

```
//RUND2E  JOB 'D2E VISION',('VISION EXAMPLE'),
//         CLASS=A,MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
//*
//*--------------------------------------------------------------------
//*                     RUN D2E VISION
//*--------------------------------------------------------------------
//*
//*--------------------------------------------------------------------
//* Runs Xenos d2e Vision(tm) transform process.  The d2e
//* Vision applicaiton is a Java based product which must
//* be executed in the USS environment.  This is
//* accomplished by using BPXBATCH to run a Unix style shell
//* script (.sh file) which resides in the USS filesystem,
//* and invokes the Java Virtual Machine (JVM) with the
//* appropriate arguments.
//*
//* This sample JCL issustrates how environment variables,
//* input and output files can be defined and passed into
//* the USS script.  Its sole purpose is an example only and
//* is not suggested that it be put directly into a production
//* system.
//*
//*--------------------------------------------------------------------
//*    EXECUTE BPXBATCH WHICH INVOKES USS SHELL SCRIPT
//*--------------------------------------------------------------------
//*
//* Call BPXBATCH using its 'PGM' parameter.  The value is
//* the location of the Unix Shell script which runs the
//* d2e Vision process
//*
//JAVASTEP EXEC PGM=BPXBATCH,PARM='PGM /u/user/vision.sh'
//*
//*--------------------------------------------------------------------
//*         DEFINE LOCATION TO WRITE STANDARD OUT
//*--------------------------------------------------------------------
//*
//* Standard out messages will be routed to the following
//* file definition to the USS filesystem.
//*
//STDOUT DD PATH='/u/user/stdout-vision.txt',
//          PATHOPTS=(OCREAT,OTRUNC,OWRONLY),
//          PATHMODE=SIRWXU
//*
//*--------------------------------------------------------------------
//*        DEFINE LOCATION TO WRITE STANDARD ERROR
//*--------------------------------------------------------------------
//*
//* Standard err messages will be routed to the following
//* file defnition to the USS filesystem.
//*
//STDERR DD PATH='/u/user/stderr-vision.txt',
//          PATHOPTS=(OCREAT,OTRUNC,OWRONLY),
//          PATHMODE=SIRWXU
//*
//*--------------------------------------------------------------------
```

```
//*
//SYSIN  DD DUMMY
//*
//*--------------------------------------------------------------------
//*           SET UP SOME ENVIRONMENT VARIABLES
//*--------------------------------------------------------------------
//*
//* These variables will be passed dynamically to the Unix
//* Shell script.  You can define any number of variables
//* here as required.  This allows changes to the JCL which
//* will be dereferenced in the d2e Vision Shell Script.
//* This way the Unix Shell Script should never have to
//* change.
//*
//STDENV DD *
_BPX_BATCH_SPAWN=YES
_BPX_SHAREAS=MUST
JAVA_HOME=/u/Java5_31/J5.0
JAVA_OPTS=-Xms128m -Xmx512m
D2E_ENGINE_PATH=/u/user/vision/engine/3.1.27
APPLICATION_HOME=/u/user/vision/workspaces/default/MyWorkspace
D2E_PROJECT_FILE=citi-meta2pdf-1to1.d2eproj
//*
//*--------------------------------------------------------------------
//*              SET UP THE DD NAMES
//*--------------------------------------------------------------------
//*
//* DD Names can be referenced a number of ways in the d2e
//* Vision application.  Please refer to the d2e Vision
//* z/OS Users Guide for further information.
//*
//*--------------------------------------------------------------------
//*
//* Define input from a sequential data set
//*
//*INSEQ   DD DSN=XENOS.META.IN.RCS63D42,DISP=SHR
//*
//*--------------------------------------------------------------------
//*
//* Define input from a partitioned data set
//*
//INPDS   DD DSN=XENOS.META.IN(RCS63D42),DISP=SHR
//*
//*--------------------------------------------------------------------
//*
//* We can dynamically create a dataset to write the
//* output if required.
//*
//*OUTSEQ   DD DSN=XENOS.META.OUT.SEQ,
//*            DISP=(NEW,CATLG,CATLG),UNIT=3390,
//*            SPACE=(TRK,(10,10),RLSE),VOL=SER=XENU07,
//*            RECFM=VB,BLKSIZE=0,LRECL=255,DSORG=PS
//*
//*--------------------------------------------------------------------
//*
//* Location of the resource libraries
//*
//MTAFNT   DD DSN=XENOS.META.RES.FNT,DISP=SHR
//MTAFRM   DD DSN=XENOS.META.RES.FRM,DISP=SHR
//MTAIMG   DD DSN=XENOS.META.RES.IMG,DISP=SHR
//MTAJSL   DD DSN=XENOS.META.RES.JSL,DISP=SHR
//MTAPDE   DD DSN=XENOS.META.RES.PDE,DISP=SHR
//FDAFM    DD DSN=XENOS.META.RES.AFM,DISP=SHR
//FDPFB    DD DSN=XENOS.META.RES.PFB,DISP=SHR
//*
//*--------------------------------------------------------------------
//
```

2.2 Sample Shell Script

This sample shell script invokes jobs from the USS file system, called from the JCL.

```
#!/bin/sh

#########################################################################
# This will run d2eVision passing up to 9 arguments into d2eMain.
# If no args are given, d2eMain will still be called causing its
# command line syntax to be shown.
#########################################################################

#########################################################################
# Use the following flags to the JVM to enable JRIO debugging
# -DRIOJADEBUG
# and
# -DRIOJATRACE_LEVEL=XX
# where XX is the level (20 is a good start)
# When enabled, debugging of JRIO will go to stderr
######

#########################################################################
# Set up a few more variables locally in this script for use later on
# $D2E_ENGINE_PATH is set in the JCL that calls this script

D2E_TEMP_CLASSPATH=.
D2E_TEMP_CLASSPATH=$D2E_TEMP_CLASSPATH:$D2E_ENGINE_PATH/bin/d2evision.jar
######

#########################################################################
# Build the classpath of all required jars
# $D2E_ENGINE_PATH is set in the JCL that calls this script

for f in $(ls $D2E_ENGINE_PATH/lib/*.jar)
do
  D2E_TEMP_CLASSPATH=$D2E_TEMP_CLASSPATH:$f
done
######

#########################################################################
# Let's log the classpath above and the environment variables passed from
# the JCL for validation.  The results will be written to stdout

echo
echo Using classpath: $D2E_TEMP_CLASSPATH
echo
echo The following variables have been passed from the JCL to be used in
echo this script:
echo
echo JAVA_HOME        = $JAVA_HOME
echo JAVA_OPTS        = $JAVA_OPTS
echo D2E_ENGINE_PATH  = $D2E_ENGINE_PATH
echo APPLICATION_HOME = $APPLICATION_HOME
echo D2E_PROJECT_FILE = $D2E_PROJECT_FILE
echo
echo Now lets run the job...
echo
echo ..................................................................
echo
#######

#########################################################################
# Now let's run the Job via d2eMain by setting up the JVM
# and passing in a number of variables set from this script
# and in the JCL
# Note: The following is a single command. All spaces in the command
# line below appear inline. Do not insert spaces at the line breaks.

$JAVA_HOME/bin/java $JAVA_OPTS -Djava.awt.headless=true -cp $D2E_\ TEMP_CLASSPATH
```

```
com.xenos.d2e.d2eMain "-config=$D2E_ENGINE_\
PATH/config/sample.d2esys" "-app=$APPLICATION_HOME/d2e/$D2E_\
PROJECT_FILE" "-input=DSN:DD:INPDS" "-output=/u/user/vision\
/workspaces/default/MyWorkspace/output/{0}.pdf"
#####

### THE END ###
```

2.3 Sample MVS Data Set Characteristics

The following are examples of MVS data set format. Input files may vary depending on the application, and individual installations may already have a format to copy.

Table 2-1: AFP Parser

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdAcif	SEQ	READ	Binary/AFP	Sets path and filename of ACIF resource file.	VB 8096+
FdAfpfonts	PDS	READ	Binary/AFP	Sets the path and file extension of font files.	VB 8096+
FdFormDefs	PDS	READ	Binary/AFP	Sets the path and file extension of form definition files.	VB 8096+
FdInput	SEQ	READ	Binary/AFP	Sets the path and name of the input file to be parsed.	VB 8096+
FdOverlays	PDS	READ	Binary/AFP	Sets the path and file extension of overlays.	VB 8096+
FdPageDefs	PDS	READ	Binary/AFP	Sets the path and file extension of page definition files.	VB 8096+
FdPageSegs	PDS	READ	Binary/AFP	Sets the path and file extension of PSEG files.	VB 8096+

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdPageEcho	SEQ	WRITE	Binary/AFP	Optional file definition for creating an AFP file containing the original pages.	VB 8096+

Table 2-2: AFP Parser.ResGroupOption

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdAcif	SEQ	READ	Binary/AFP	Sets path and file name of ACIF resource file.	VB 8096+
FdAfpfonts	PDS	READ	Binary/AFP	Sets the path and file extension of font files.	VB 8096+
FdFormDefs	PDS	READ	Binary/AFP	Sets the path and file extension of form definition files.	VB 8096+
FdInput	SEQ	READ	Binary/AFP	Sets the path and name of the input file to be parsed.	VB 8096+
FdOverlays	PDS	READ	Binary/AFP	Sets the path and file extension of overlays.	VB 8096+
FdPageDefs	PDS	READ	Binary/AFP	Sets the path and file extension of page definition files.	VB 8096+
FdPageSegs	PDS	READ	Binary/AFP	Sets the path and file extension of PSEG files.	VB 8096+

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdPageEcho	SEQ	WRITE	Binary/AFP	Optional file definition for creating an AFP file containing the original pages.	VB 8096+

Table 2-3: Metacode Parser

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255
FdCme	PDS	READ	Binary	Sets the path and file extension of external Cme files.	FIXED=128 or 512
FdExternRes	??	??	Binary	Sets the path and file extension of external resource files.	FIXED=128 or 512
Fdfnt	PDS	READ	Binary	Sets the path and file extension of external font files.	FIXED=128 or 512
FdFrm	PDS	READ	Binary	Sets the path and file extension of external form files.	FIXED=128 or 512
FdImg	PDS	READ	Binary	Sets the path and file extension of external image form files.	FIXED=128 or 512
FdInput	SEQ	READ	Binary/Meta	Sets the path and name of the input file to be parsed.	VB or Fixed

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdJsl	PDS	READ	Readable	Sets the path and file extension of external Jsl files.	FIXED=80 or VB 80
FdLgo	PDS	READ	Binary	Sets the path and file extension of external Lgo files.	FIXED=128 or 512
FdPde	PDS	READ	Binary	Sets the path and file extension of external Pde files.	FIXED=128 or 512
FdXfw	PDS	READ	Binary	Sets the path and file extension of external Xfw files.	RECFM=VB, LRECL=any

Table 2-4: PCL Parser

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdInput	SEQ	READ	Binary	Sets the path and name of the input file to be parsed.	any
FdPclDump	SEQ	WRITE	Mixed	Sets the optional path to a debug dump file containing the original PCL bytes.	any
FdStats	SEQ	WRITE	Readable	Location of the optional stats file. When this parameter is present, the parser/ generator will write out the tuning stats to the file.	VB 255

Table 2-5: PDF Parser

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdInput	SEQ	READ	Binary	Sets the path and name of the input file to be parsed.	VB any
FdCmap	PDS	READ	Binary	Sets the path that specifies where the CMAP files are located.	VB any
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-6: TIFF Parser

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdInput	SEQ	READ	Binary	Sets the path and name of the input file to be parsed.	VB any
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-7: AFP Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdOutput	SEQ/PDS	WRITE	Binary/AFP	Specifies the destination for the generated document(s).	VB 8096+
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdFormDefs	PDS	WRITE	Binary/AFP	Specifies the location to be used to write out formdefs that are generated by the AFP Generator.	VB 8096+
FdOverlays	PDS	WRITE	Binary/AFP	Specifies the location of overlays that are generated by the AFP Generator.	VB 8096+
FdPageSegs	PDS	WRITE	Binary/AFP	Specifies the location of page segments that are generated by the AFP Generator.	VB 8096+
FdAcifOutput	SEQ	WRITE	Binary/AFP	Specifies the filename of the ACIF file.	VB 8096+
FdFonts	PDS	WRITE	Binary/AFP	Used to locate existing external font resources whenever they are needed.	VB 8096+

Table 2-8: Image Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdOutput	SEQ	WRITE	Binary	The path and name of the output file to be generated.	VB any

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdStats	SEQ	WRITE	Readable	Location of the optional stats file. When this parameter is present, the parser/ generator will write out the tuning stats to the file.	VB 255

Table 2-9: Metacode Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommend ed Format
FdOutput	SEQ/PDS	WRITE	Binary/Meta	The destination for the generated document(s).	VB 300 (or as per MaxMetaRec ord Parm)
FdFnt	PDS	READ	Binary	Used to locate existing external font resources whenever they are needed.	FB 128 or 512
FdImg	PDS	READ	Binary	Used to locate existing external IMG resources whenever they are needed.	FB 129 or 512
FdFrm	PDS	READ	Binary	Used to locate existing external form resources whenever they are needed.	FB 128 or 512

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdJSL	PDS	READ	Readable	Locates the existing JSL resource that the resulting Metacode output document will be printed with.	FB 80 or VB any
FdLgo	PDS	READ	Binary	Used to locate existing external Lgo resources whenever they are needed.	FB 128 or 512

Table 2-10: PCL Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdOutput	SEQ/PDS	WRITE	Binary	The destination for the generated document(s).	VB any (large)
FdPCLfnt	PDS	READ	Binary	Specifies the location of external PCL fonts (typically *.fnt files).	VB any
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-11: PDF/A Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdIcm	SEQ	READ	Binary	Color profile (*.icm) file. This is required for the PDF/A file.	VB any

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdOutput	SEQ/PDS	WRITE	Binary	Sets path and filename of the output PDF file.	VB any (large)
FdAfm	PDS	READ	Readable	Optional font metrics file.	VB any
FdPfb	PDS	READ	Binary	Optional font binary data file.	VB any
FdTtf	PDS	READ	Binary	Optional font binary data file for TrueType fonts.	VB any
FdEncoding	PDS	READ	Readable	Optional file that maps code points to glyph names.	VB 255
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-12: PDF Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdTemp	SEQ	READ/ WRITE	Binary	Sets the path and filename of the temporary file to use for streaming linearized PDF.	VB any
FdOutput	SEQ/PDS	WRITE	Binary	Sets the path and filename of the output PDF file.	VB any (large)
FdAfm	PDS	READ	Readable	Optional font metrics file.	VB any
FdPfb	PDS	READ	Binary	Optional font binary data file.	VB any

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdTtf	PDS	READ	Binary	Optional font binary data file for TrueType fonts	VB any
FdEncoding	PDS	READ	Readable	Optional file that maps code points to glyph names.	VB any
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-13: PostScript Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdOutput	SEQ/PDS	WRITE	Binary (ASCII)	Sets path and filename of the output PostScript file.	VB any (large)
FdPSHeader	SEQ	READ	Binary (ASCII)	Specifies a file that is appended to the Header section of the output PostScript file.	VB any
FdPSSetup	SEQ	READ	Binary (ASCII)	Specifies a file that is appended to the Setup section of the output PostScript file.	VB any
FdTicket	SEQ	READ	Binary (ASCII)	Specifies a job ticket that is placed at the beginning of the output PostScript file.	VB any

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-14: TIFF Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdOutput	SEQ/PDS	WRITE	Binary	The path and name of the output file to be generated.	VB any (large)
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-15: Web Generator

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
FdOutput	SEQ/PDS	WRITE	Binary/ASCII	Sets the output path and filename for HTML files.	VB any
FdImageOutput	PDS	WRITE	Binary	Sets the output path and filename for images.	VB any (large)
FdVectorOutput	PDS	WRITE	Binary	Sets the output path and filename for vector images.	VB any
FdReplacement Images	PDS	READ	Binary	Sets the input path containing images to be replaced.	VB any
FdStats	SEQ	WRITE	Readable	Location of the optional stats file.	VB 255

Table 2-16: Constant Print Component

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
Constant Image Path	PDS	READ	Binary	Location of images used in ConstantImage.	VB any

Table 2-17: Index Writer Component

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
OutputFile	SEQ/PDS	WRITE	Varies	The filename to write the index file to.	VB any/varies

Table 2-18: XFT Writer

Parameter Name	Data Set Type	Mode	Data Format	Description	Recommended Format
OutputFile	SEQ/PDS	WRITE	Readable/XML	VB 255	
FdOdIndex	SEQ/PDS	WRITE	Readable	The path and name of the output index file to be generated.	VB 255