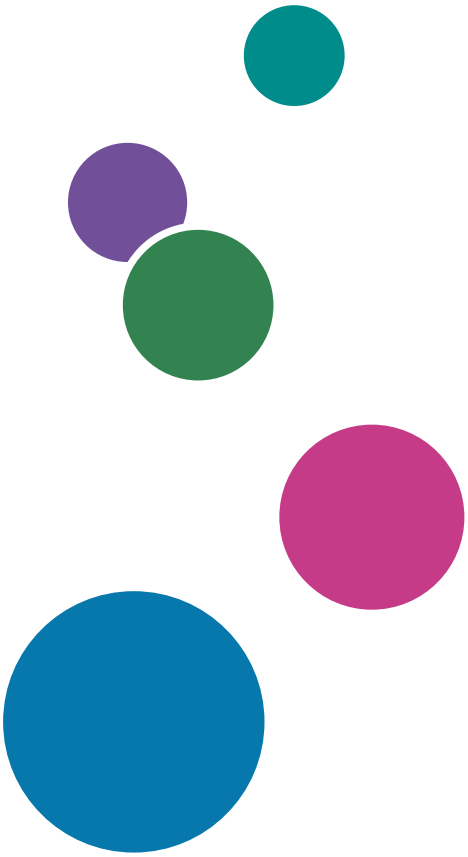
RICOH

Data Transforms

AFP2PDF Plus Transform User's Guide

Version 1.300

For information not in this manual, refer to the Help System in your product.



Read this manual carefully and keep it handy for future reference.

About This Publication

This publication uses the following symbols:

Bold	Bold type indicates the names of commands and parameters.
Bold underline	Underlined bold type indicates the default value.
Italic	*Italic type* indicates variables that you must replace with your own information.
Monospace	Monospace type indicates computer input and output and file names.
[ ]	Square brackets indicate that a value is optional.
|	A vertical bar indicates a choice between values.
...	An ellipsis indicates that a series can continue.

TABLE OF CONTENTS

1. Overview

- **Benefits**
- **Limitations**

The AFP2PDF Plus Transform coverts Advanced Function Presentation (AFP) documents into Adobe Acrobat Portable Document Format (PDF) files by exactly mapping AFP format to PDF format.

The AFP2PDF Plus Transform lets you:

- Operate on any system that supports JAVA Version 1.7 or later.
- View documents with the same fidelity as if they were printed. If the Adobe Acrobat plug-in is installed with a Web browser, you can view and print these documents within the browser application.
- Use configuration files to customize how AFP documents are transformed.
- Fully integrate with the IBM Content Manager OnDemand Web Enablement Kit and the IBM Content Navigator.

Benefits

The AFP2PDF Plus Transform gives you these added benefits:

- Support for all types of bar code objects from the Bar Code Object Content Architecture (BCOCA).
- Client/Server implementation.
- Greatly improved transformation speed over AFP2PDF (5876-W01).
- Larger than 2–GB input and output file support.
- Reduce costs associated with printing and mailing by delivering documents electronically.
- Quickly retrieve your information within multi-page documents using Adobe Acrobat search and navigation features.
- Print AFP2PDF Plus Transform documents on any local printer using the print function in Adobe Acrobat.
- Increase control over your information with an added layer of security and encryption. To control modification, copying, and printing, define an owner password; define an end-user password to control document access; and add a digital signature to provide more security.

Limitations

The current limitations of the AFP2PDF Plus Transform include:

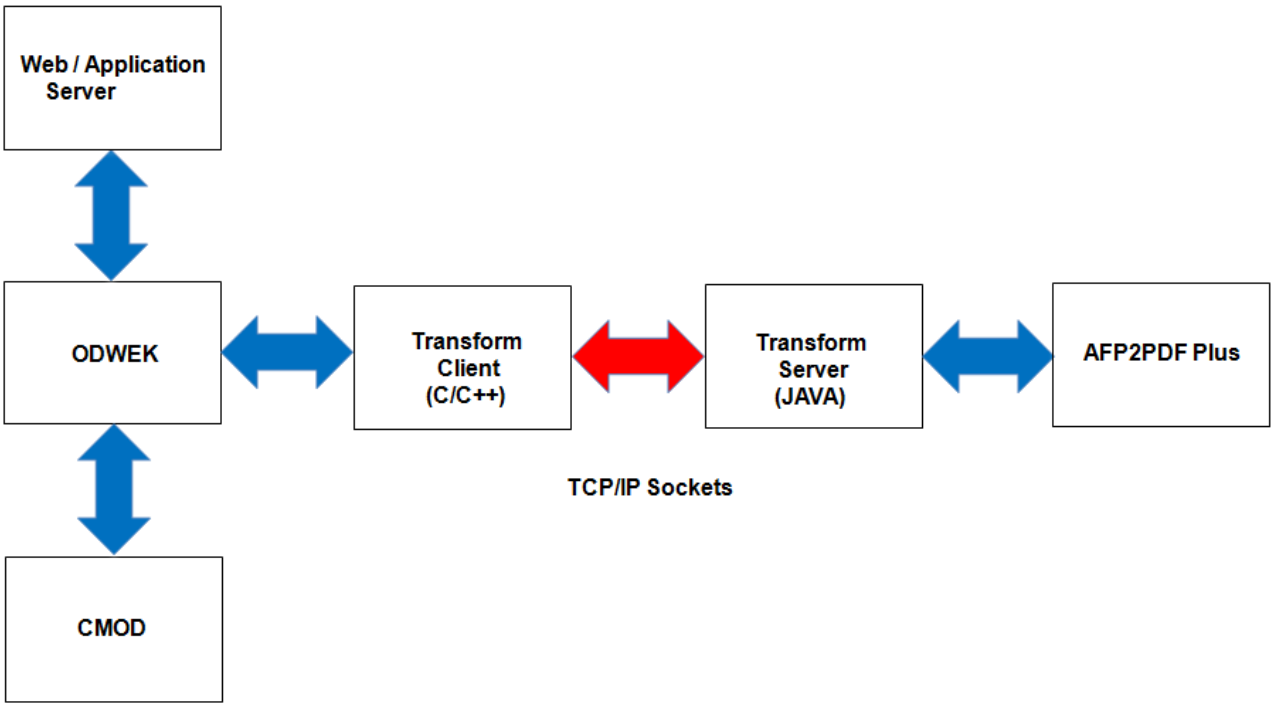
- When AFP data formatted for N-up portioning is encountered, the multiple partitions that make up the physical AFP page are converted to separate pages in the PDF file.
- Only limited support is available for object containers with TIFF and JFIF image formats.
- Color management resource (CMR) information contained in AFP resources is ignored.
- Unicode Complex Text, such as bidirectional layout processing and glyph processing, is not supported.
- TrueType font collections, linked TrueType fonts, and the use of a Resource Access Table (RAT) are not supported.

2. Installing AFP2PDF Plus Transform

- **Upgrading from AFP2PDF to AFP2PDF Plus Transform**
- **Installing AFP2PDF Plus Transform using the installer**
- **Installing AFP2PDF Plus Transform on a Windows Server from a zip archive file**
- **Installing AFP2PDF Plus Transform on a UNIX Server from a tar.gz archive file**
- **Installing AFP2PDF Plus Transform with OnDemand**
- **Installing AFP2PDF Plus Transform with Content Navigator**

AFP2PDF Plus Transform has a different architecture from the AFP2PDF Transform.

AFP2PDF Plus Transform Architecture



Since the new AFP2PDF Plus for JAVA Transform is designed to integrate seamlessly with existing AFP2PDF installations, it has both client and server functions.

The transform client is a C/C++ command-line program that has parameters identical to the current AFP2PDF Transform.

The transform server is JAVA-based and performs various functions:

- Accepts requests from the transform client for document conversion
- Keeps the JAVA Virtual Machine (JVM) active without starting and stopping after each conversion
- Spawns a thread for each document to be converted, taking advantage of multi-core processors

These are the transform client and server requirements and end-user client requirements for the AFP2PDF Plus Transform:

- Transform client requirements:
  - IBM AIX 7.1 or later
  - IBM z/OS UNIX System Services V1.13 or later
  - Microsoft Windows 10 Pro, Enterprise

  - Microsoft Windows Server 2016 Std
  - Linux Kernel 2.6.18 or later (x86)
  - Linux Kernel 2.6.09 or later (IBM System z)
  - Oracle Solaris 10.9 or later (SPARC only)
- Transform server requirements: JAVA V1.7 or later
- End-user client requirements: Adobe Acrobat, Acrobat Reader, or Acrobat Plug-In 9.0 or later
- To run the installer, you need to have at least 350 MB of free space in the operating system temporary directory.

Upgrading from AFP2PDF to AFP2PDF Plus Transform

The Java jar file, `Upgrade.jar`, is an executable program that upgrades C/C++ AFP2PDF transform to AFP2PDF plus.

`Upgrade.jar` performs these steps to upgrade from AFP2PDF transform to AFP2PDF Plus:

1. Backs up executable modules in C/C++ AFP2PDF transform:
   1) Creates a native backup folder in the C/C++ AFP2PDF install directory. If the `native_backup\` folder exists, the backup folder name appends "-1", that is, `native_backup-1\`. The maximal iteration of appending is 9 times of "-1".
   2) Moves all existing license\folder and java_api\folder in C/C++ install directory to the backup folder.
   3) Moves all AFP2PDF* and split_afp2pdf* to the backup directory.
2. Copies AFP2PDF plus modules, documents, server configuration file and property files to the C/C++ AFP2PDF install directory:
   1) Copies AFP2PDF or AFP2PDF.exe to the C/C++ AFP2PDF install directory.
   2) Copies *. jar files to the C/C++ AFP2PDF install directory.
   3) Copies LICENSE/* to the C/C++ AFP2PDF install directory.
   4) Copies *.sh or *.bat to the C/C++ AFP2PDF install directory.
   5) Copies *.properties to the C/C++ AFP2PDF install directory.
   6) Copies Server.cfg to the C/C++ AFP2PDF install directory.
   7) Copies AFP2PDF_Plus_UG.pdf to the C/C++AFP2PDF install directory.
   8) Copies a2pxopts.cfg to the C/C++ AFP2PDF install directory.
   9) Copies a2pclient.cfg to the C/C++AFP2PDF install directory.

Usage

The jar file `Upgrade.jar` must be in the AFP2PDF plus directory. From the AFP2PDF plus directory, run the command:`java -jar Upgrade.jar <C/C++ afp2pdf install directory>`. If no parameter is given, this message is displayed: Upgrade the C/C++ AFP2PDF transform to Java transform (AFP2PDF Plus).

You need to run the jar file `Upgrade.jar` in the AFP2PDF Plus installed directory.

Installing AFP2PDF Plus Transform using the installer

The AFP2PDF Plus Transform can now be installed using an executable jar file, `setupafp2pdfPlus`, from the ISO file.

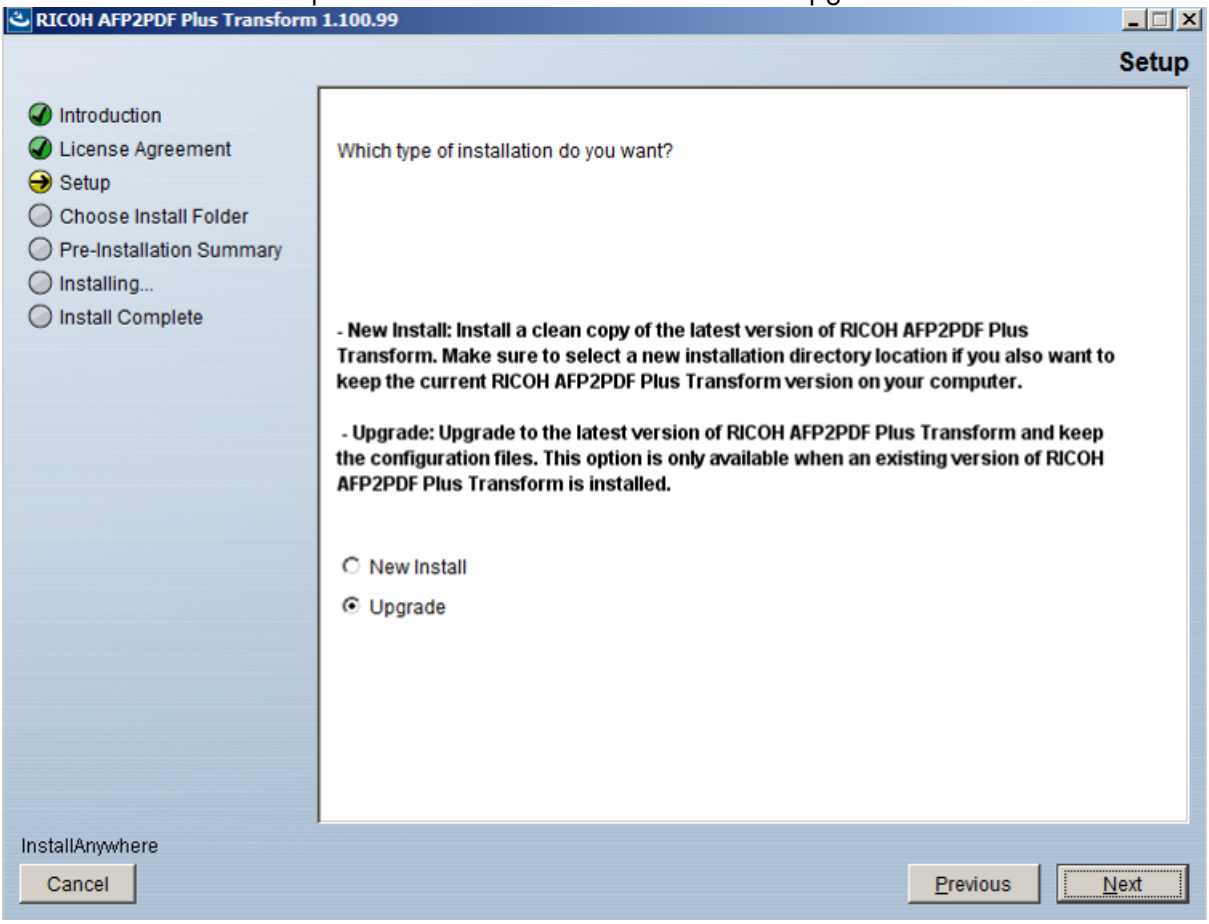
To run the transform, perform these steps:

1. Use the following command: `java -jar setupafp2pdfPlus.jar`.
   Depending of the capabilities of your operating system, the AFP2PDF Plus transform opens either using a graphical user interface or a command line.

   Note

   - To install the transform, you need Java 1.7.
     See Overview, p. 5 and Installing AFP2PDF Plus Transform, p. 7 for more information on requirements.

2. Follow the installation steps and select either a new install or an upgrade.



Note

- If you select the **Upgrade** option, you install AFP2PDF Plus in a directory where a previous transform exists. During an upgrade the configuration files are not modified, only the binaries are updated.

Installing AFP2PDF Plus Transform on a Windows Server from a zip archive file

For Windows operating systems, the AFP2PDF Plus Transform program is delivered in a ZIP file with the file name in this format:

`afp2pdf_plus_v`*`xxx`*`_WIN.ZIP`

where *xxx* is the version number of the program.

For example, `afp2pdf_plus_v100_WIN.ZIP` is the name of the AFP2PDF Plus Transform with a version number of "100" for Windows operating system.

To install the transform:

1. Unzip the file in the directory of your choice.
2. Retain the directory structure as you unzip the file.
   The `\font` subdirectory is created in the install directory.

The split_afp2pdf and archiveload_afp2pdf programs are delivered as a part of the AFP2PDF Plus Transform. The operating system prerequisites are the same as for the AFP2PDF Plus Transform.

Installing AFP2PDF Plus Transform on a UNIX Server from a tar.gz archive file

For UNIX environments (AIX, Linux, Oracle Solaris, and z/OS UNIX System Services), the AFP2PDF Plus Transform is delivered in a compressed TAR file with the file name in this format:

`afp2pdf_plus_v`*`xxx`*`_`*`yyy`*`.tar.gz`

where *xxx* is the version number and *yyy* indicates the server ID of the program.

For example, `afp2pdf_plus_vxxx_yyy.tar.gz` is the name of the AFP2PDF Plus Transform with a version number of "100" and a server ID of "AIX". The server IDs are:

- **AIX** for IBM AIX
- **LINUX** for Linux (x86)
- **zLINUX** for IBM zLinux
- **SOL** for Oracle Solaris
- **USS** for IBM z/OS UNIX System Services

To install the transform:

1. Uncompress and unpack the file in the directory of your choice.
2. Retain the directory structure as you unzip the file.
   The `\font` subdirectory is created in the install directory.
3. Set the permissions so that the process that runs AFP2PDF Plus can read from the directory in which the program is installed.

The split_afp2pdf and archiveload_afp2pdf programs are delivered as a part of the AFP2PDF Plus Transform. The operating system prerequisites are the same as for the AFP2PDF Plus Transform.

Installing AFP2PDF Plus Transform with OnDemand

Since the new AFP2PDF Plus Transform is designed to integrate seamlessly with existing Content Manager OnDemand Web Enablement Kit / AFP2PDF Transform installations, see the appropriate IBM documentation for more details about configuring the programs to operate together:

- IBM Content Manager OnDemand for Multiplatforms, V8.5: Web Enablement Kit Implementation Guide, SC19-2941
- IBM Content Manager OnDemand for Multiplatforms & z/OS, V9.0 / V9.5: Web Enablement Kit Implementation Guide, SC19-3353
- IBM Content Manager OnDemand for z/OS, V8.5: Web Enablement Kit Implementation Guide, SC19-1215

Specifically, you must modify the `arswww.ini` file to call the AFP2PDF Plus Transform when processing an AFP file. At a minimum, you must make these configuration changes:

- The `AFPViewing` option must be `pdf` in the browser sections.
- The `InstallDir` option in the `afp2pdf` section must point to the directory on the server that contains the AFP2PDF Plus Transform client program.

Installing AFP2PDF Plus Transform with Content Navigator

To use the AFP2PDF Plus Transform with IBM Content Navigator (ICN), you must configure the ICN Administrator plug-in.

1. Open the web browser where the ICN plug-ins have been installed and log in to the ICN Administrator plug-in.
2. To configure the installation directory:
    1. On the left-hand side of the browser screen, select **Settings**.
    2. In the main panel, select the **Content Manager OnDemand** tab.
    3. In the **AFP2PDF installation directory** field, enter the fully qualified path name for the directory where the AFP2PDF Plus transform is installed.
    4. Select **Save and Close**.
3. To configure the viewer map:
    1. On the left-hand side of the browser screen, select **Default Viewer Map**.
    2. Make sure that the **Repository Type** is set to `Content Manager OnDemand`, **Viewer** is set to `AFP2PDF Conversion`, and **MIME Type** is set to `application/afp`.

       If any of the values are incorrect, refer to the Planning, installing, and configuring IBM Content Navigator manual for more details on how to change these values.
4. Start the AFP2PDF Plus server.

   See Starting and Stopping the AFP2PDF Plus Server on a Windows Server, p. 21 or Starting and Stopping the AFP2PDF Plus Server on a UNIX Server, p. 21 for more details.

3. Architecture and Configuration

- **Architecture**
- **Configuration**
- **AFP2PDF Plus Transform Security**

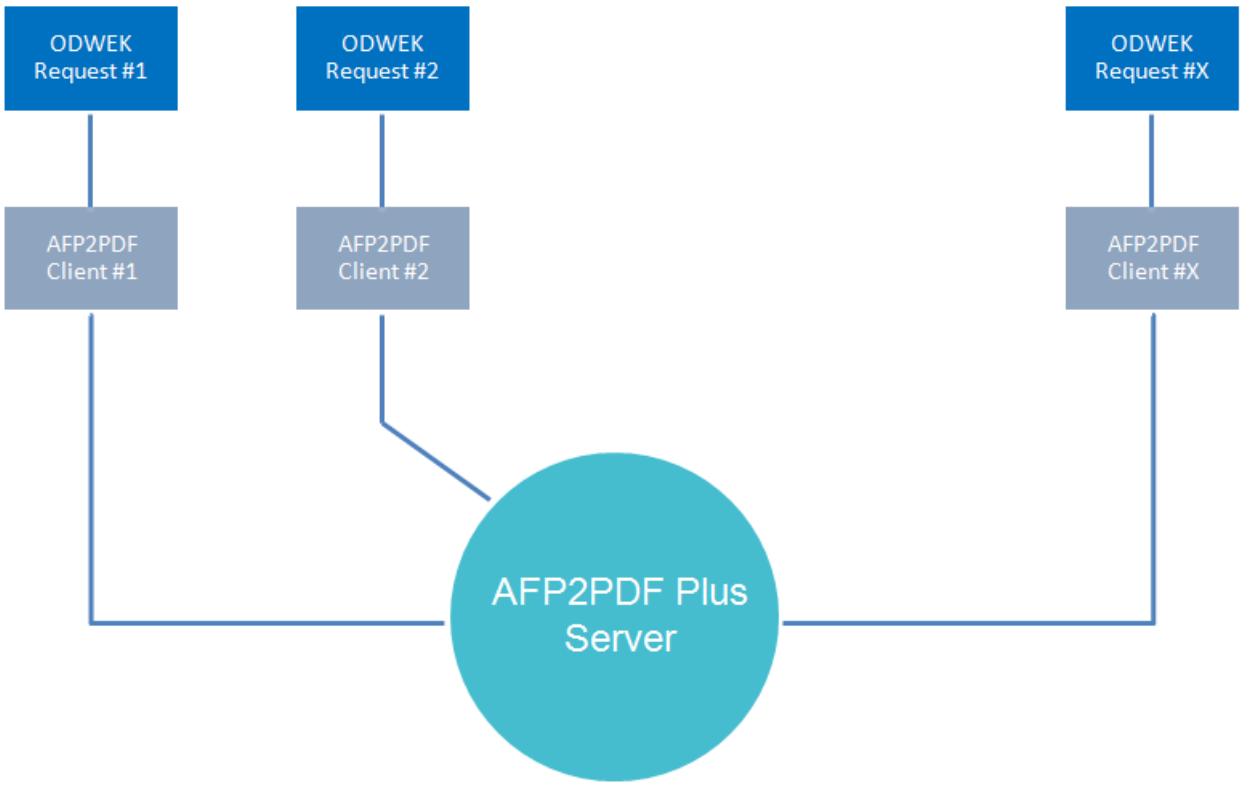
This chapter describes the overall architecture of the AFP2PDF Plus Client/Server implementation and the files used during the configuration process. It also describes security for the transform.

Architecture

The transform is written in JAVA to take advantage of multi-threaded client/server technology and platform independence. Both a JAVA command line and Application Programming Interface (API) are available.

The transform is also designed to integrate easily with existing installations of IBM Content Manager OnDemand by using the AFP2PDF Plus Transform client programs. These client programs are C/C++ based and unique to each operating system supported by Content Manager OnDemand. They mimic the previous `AFP2PDF.exe` command syntax and are designed to less integrate into existing AFP2PDF installations with minimal migration effort. OnDemand Web Enablement and AFP2PDF Plus, p. 13 shows details of the OnDemand Web Enablement and AFP2PDF Plus architecture.

OnDemand Web Enablement and AFP2PDF Plus



When the OnDemand Web Enablement Kit retrieves an AFP document, it issues a conversion request via the `AFP2PDF.exe` command-line interface. The AFP2PDF client passes the file to be converted, along with the transform parameters to the AFP2PDF Plus Server (via a TCP/IP socket port), where the file is converted. The AFP2PDF Plus Server writes the results directly back to the specified file system directory. Input requests are handled on a First In First Out (FIFO) basis. The maximum number of client requests handled at any given time depends on the thread count setting in the `Server.cfg` file.

Configuration

This section describes the commands and files used during the initial configuration process. The basic operation of the AFP2PDF Plus client and server are controlled by three configuration files: `Server.cfg`, `logging.properties`, and `a2pclient.cfg`.

Note

The directory and file examples in this chapter are specified for the Windows environment. To use these examples in a UNIX environment, use the UNIX file naming convention for any file name. For example, `\font\maps` in Windows is `/font/maps` in a UNIX environment, or the input AFP file `c:\documents\afpdon.afp` in Windows would be `/documents/afpdoc.afp` in a UNIX environment.

Configuring the `Server.cfg` File

The `Server.cfg` file controls the overall server operation of the AFP2PDF Plus Transform.

The `Server.cfg` file consists of these entries:

port:*n*

Indicates the TCP/IP port number that the AFP2PDF Plus client and server use to communication with one another. It comes preset to `6512`, which is currently unassigned in the Service Name and Transport Protocol Port Number Registry, which is administered by the Internet Assigned Numbers Authority (IANA). You can change this port number to one of your own choosing, but be careful that it does not conflict with any others in use in your organization.

socketTimeout:*n*

Indicates that the sleep time on socket accept in milliseconds. It comes preset to `3000` (3 seconds).

stopFile:*filename*

Specifies the fully qualified name of the stop server file indicator. As a part of the server shutdown process, `StopAFP2PDFServer.bat` or `StopAFP2PDFServer.sh` reads the `Server.cfg` file for the **stopFile** name and creates it. When the **stopFileTimeout** expires, the server checks for the stop server file indicator and if present, stops the server. The default value is `C:\temp\stopServer.txt`.

stopFileTimeout:*n*

Indicates the sleep time between **stopFile** checks in milliseconds. It comes preset to `10000` (10 seconds).

threadCount:*n*

Indicates the maximum number of AFP2PDF Plus server threads. It comes preset to `3`. It can be changed to improve performance, but should not exceed the number of processor cores in your server hardware.

transformHandler:*classname*

Indicates the name of the AFP2PDF Plus Transform client handler class. This comes preset to `com.ricoh.ips.transform.fromAFP.server.AFPPDFClienthandler`. The end user should not change this value.

initializationHandler:*classname*

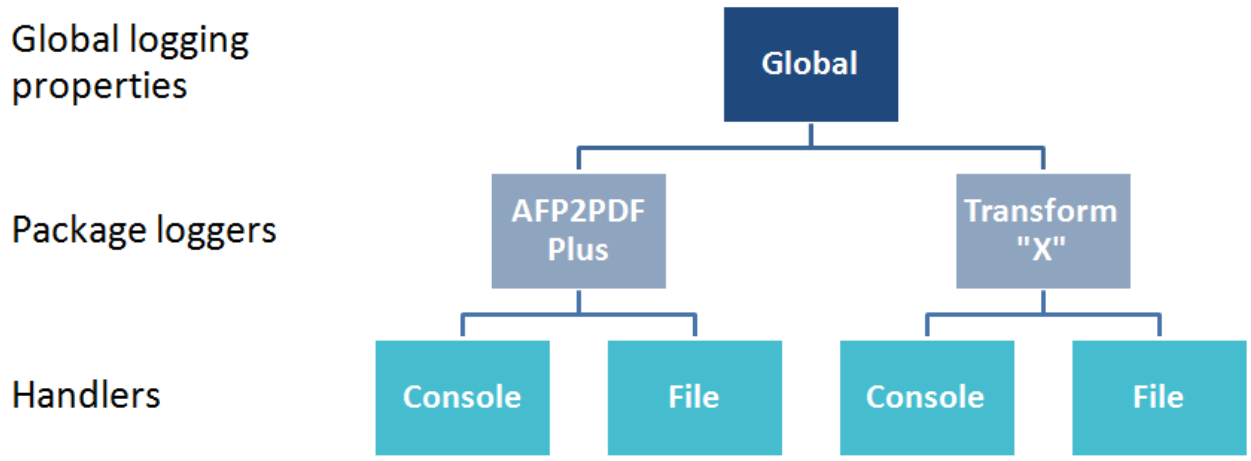
Indicates the name of the transform initialization class, called once the server starts up. This comes preset to: `com.ricoh.ips.transform.fromAFP.server.AFP2PDFInitializationHandler`. The end user should not change this value.

Configuring the `logging.properties` File

The `logging.properties` file specifies the type of error logging that is turned on for program debugging.

`logging.properties` File Structure, p. 15 shows its structure, which is designed to allow more transforms to be added in the future.

`logging.properties` File Structure



The `logging.properties` file configures the operation of the AFP2PDF Plus logging facility. It consists of these three sections:

Global logging properties

handlers=*classname*[,*classname*]

Specifies the set of global logging handlers to be loaded upon startup. This comes preset to `java.util.logging.FileHandler,java.util.logging.ConsoleHandler`. The end user should not change this value.

Package loggers

com.ricoh.ips.transform=INFO | WARNING | SEVERE | ALL | OFF

Specifies the logging level for the AFP2PDF Plus transform. The default is **ALL**. The end user should not change this value.

Console and file handlers

The console and file handlers operate independently of each other. For example, critical messages could go directly to the console, while other messages are written to the log file.

The console handler has these entries:

java.util.logging.ConsoleHandler.level=INFO | WARNING | SEVERE | ALL | OFF

Indicates the global logging message override setting. The default is **SEVERE**. Change this setting to the level of detail you want to appear on the console.

This value overrides the global logging level, whose default is **INFO**

java.util.logging.ConsoleHandler.formatter=*classname*

This comes preset to `java.util.logging.SimpleFormatter`. The end user should not change this value.

The file handler has these entries:

java.util.logging.FileHandler.level=INFO | WARNING | SEVERE | ALL | OFF

Indicates the global logging message override setting. The default is **WARNING**. Change this setting to the level of detail you want to appear in the log file.

This value overrides the global logging level, whose default is **INFO**

java.util.logging.FileHandler.append=true | false

Indicates whether trace information is appended to the end of the log file. The default is **true**.

java.util.logging.FileHandler.formatter=*classname*

This comes preset to `java.util.logging.SimpleFormatter`. The end user should not change this value.

java.util.logging.FileHandler.pattern=*filename*

Specifies the fully qualified AFP2PDF Plus log file name. It comes preset to `./Afp2Pdf.log`. You can change this value.

java.util.logging.FileHandler.limit=*n*

Indicates the maximum size of the log file. When the log file reaches this size, it wraps.

java.util.logging.FileHandler.count=*n*

Indicates the number of log files to cycle through by appending an integer to the base file name.

Configuring the `a2pclient.cfg` File

The `a2pclient.cfg` specifies the port number for the client to server communication.

The `a2pclient.cfg` file consists of:

port:*n*

Specifies the TCP/IP port number that the client uses to communicate with the server. The default port is **6512**. This can be changed, but has to match the server port number in the `Server.cfg` file.

Note

- This client configuration file is optional.

AFP2PDF Plus Transform Security

The AFP2PDF Plus Transform Encryption supports these Adobe Acrobat security features: encryption and certificate signatures.

Encryption

Encryption protects the contents of the PDF document and lets you restrict certain Adobe Actobat display functions. The Adobe Acrobat password features let users access protected documents and restricted functions.

A PDF document can have these levels of password protection:

- A document open password (also known as a user password) controls read access to the PDF document.
- A permissions password (also known as an owner or master password) controls the use of restricted functions. See Parameters, p. 22 for information about using the **–a** and **–e** parameters to restrict display functions.

  Note

  Only Adobe software fully supports and respects these settings. Users of third-party PDF-enabled programs might be able to bypass some of these restrictions.

Both types of passwords can be set for a document. If the PDF document has both types of passwords, it can be opened with either password.

AFP2PDF Plus supports the following encryption algorithms:

- RC4 40-bit (Default)
- RC4-128-bit
- AES 128-bit
- AES 256-bit

RC4 (originally developed by RSA Security) is a symmetric stream cipher: the same algorithm is used for both encryption and decryption, and the algorithm does not change the length of the data. 40-bit and 128-bit refer to the length of the encryption key. 40-bit keys in general are susceptible to “brute force” attacks, which try all possible passwords. The design of the RC4 encryption algorithm does have some vulnerabilities, which tend to make it less secure than AES; but both RC4 40-bit and 128-bit are still used extensively around the world.

The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST). The algorithm described by AES is also a symmetric stream cipher. These ciphers each have a block size of 128 bits, but two different key lengths: 128 bit and 256 bit, hence the terms: AES 128-bit and AES 256-bit. AES is considered more secure because of weaknesses in RC4 keys.

Certificate Signatures

Adobe Acrobat supports the use of certificate signatures (also known as digital signatures), which let you sign PDF files with a certificate-based digital ID. This certificate ID is issued by a trusted third-part certificate authority, such as CAcert.org or VeriSign.

A digital signature can be used to authenticate the identity of a user and the document's content. It stores information about the signer and the state of the document when it was signed. Any subsequent changes to a signed PDF file invalidate the signature.

The AFP2PDF Plus Transform requires a digital certificate in the PKCS#12 format. PKCS stands for Public Key Cryptography Standard, originally developed by RSA Security, Inc. PKCS#12 defines a file format used to store private keys with accompanying public key certificates, protected with a password-based symmetric key.

The AFP2PDF Plus Transform makes use of a digital certificate through the use of one of these options in the `a2pxopts.cfg` file:

- **Certificate** requires both the name of the PKCS#12 file and its password to read the PKCS#12 file.
- **Certificate2** requires the name of the PKCS#12 file, but lets you point to a file that contains the needed password.

Once the file has been signed, the validity can be checked by opening the file with Adobe Acrobat.

4. Using the AFP2PDF Plus Transform

This chapter describes how to start and stop the AFP2PDF Plus Server and how to run the AFP2PDF Plus Transform client.

Starting and Stopping the AFP2PDF Plus Server on a Windows Server

For Windows operating systems, two programs are supplied:

- `StartAFP2PDFServer.bat`
- `StopAFP2PDFServer.bat`

They contain the necessary commands to properly start and stop the JAVA environment, along with the AFP2PDF Plus Server.

Starting and Stopping the AFP2PDF Plus Server on a UNIX Server

For UNIX operating systems, two programs are supplied:

- `StartAFP2PDFServer.sh`
- `StopAFP2PDFServer.sh`

They contain the necessary commands to properly start and stop the JAVA environment, along with the AFP2PDF Plus Server.

AFP2PDF Plus Transform Command

The new afp2pdf client command is plug-compatible with the previous afp2pdf command (5876-W01). It transforms AFP files and resources into PDF files that you can distribute over the Internet and view with a Web browser installed with the Adobe Acrobat plug-in. If the plug-in is not installed on your Web browser, you can view the PDF files with Adobe Acrobat Reader.

Syntax

Only some of the parameters needed to control the AFP2PDF Plus Transform are available with the client command. The other options are specified in an options file. By default, this file is named `a2pxopts.cfg`. It must reside in the same directory as the client program module. See AFP2PDF Plus Transform Options File, p. 25 for more information about the options file.

The syntax for the afp2pdf client command is:

```
afp2pdf [-host host] [-port port] [-a functions -e password]
[-f fdeffile] [-g ( timestamp)] [-i optfile] [-l] [-m]
[-n fontpath] [-o outputfile] [-p page_no] [-r resfile]
[-s (timestamp)] [-t rotation] [-u password] [-v mapfile]
inputfile
```

Parameters

-host *host*

Specifies the host name where the AFP2PDF Plus server is running. The default is **localhost**.

-port *port*

Specifies the TCP/IP port number that the client uses to communicate with the server. The default port is **6512**. This can be changed, but has to match the server port number in the `Server.cfg` file.

Note

- This parameter overrides any existing **port:*n*** specified in the optional `a2pclient.cfg` file.

-a *functions*

When used with an owner password (**-e** *password*), specifies which PDF display functions are restricted. The codes for the display functions, which can be used in any order, are:

a

Add or modify text annotations and interactive form fields.

c

Modify the document contents.

p

Print the document.

s

Copy text and graphics from the document.

-e *password*

Specifies an alphanumeric password that gives permission to change document security settings (also known as a master, owner, or document open password). For example, if the printing function in the PDF display is restricted (`-a p`), you must supply a password to override the setting.

-f *fdeffile*

Specifies the fully qualified file name of the form definition resource that is used when transforming an AFP file. For example, `afp2pdf -f c:\mydirectory\myformdef.fde afpfile.afp`.

-g (*timestamp*)

Sets a time stamp for when the document was created.

Note

This command can be specified, but is not supported in the current implementation. It exists for compatibility purposes only.

The format of the time stamp is:

```
YYYMMDDHHmmSSOHH'mm'
```

where:

- *YYYY* is the year.
- *MM* is the month.
- *DD* is the day (01-31).
- *HH* is the hour (00-23).
- *mm* is the minute (00-59).
- *SS* is the second (00-59).
- *O* is the relationship of local time to Universal Time (UT), denoted by one of the characters +, -, or Z.
- *HH* is the absolute value of the offset from UT in hours (00-23).
- *mm* is the absolute value of the offset from UT in minutes (00-59).

For example, December 23, 1998, at 7:52 PM, U.S. Pacific Standard Time, is represented by the string `19981223195200-08'08'`.

-i *optfile*

Specifies the location and name of the transform options file. The default is `a2pxopts.cfg` in the same directory as the program module. This file name must not use relative paths and must be fully qualified. See AFP2PDF Plus Transform Options File, p. 25 for more information about the options file.

-l

Turns off all the generated error and informational console messages and sends them to the `afp2web.err` log file.

Note

This command can be specified, but it is not supported in the current implementation. It exists for compatibility purposes only.

AFP2PDF Plus error logging is controlled by the `logging.properties` file. For information about this file, see Configuring the `logging.properties` File, p. 15.

-m

Generates a linearized PDF (or Fast Web View PDF) that reorganizes the data for more efficient processing in a network environment. When specifying this parameter, large PDF files are displayed without long download delays.

Note

To use this parameter, verify that the Web server or Web application sending the PDF data over the network provides the page-by-page or "byte-serving" downloading function.

-n *fontpath*

Specifies the location of the font definition files needed by the transform. The default is the `\font` subdirectory of the directory where the transform modules are installed.

-o *outputfile*

Specifies the location and file name of the output PDF file. The default has the same location and name as the input file, but with a file extension of `.pdf`. For example, when the PDF is generated from an AFP file named `afpdoc.afp`, a file named `afpdoc.pdf` is created.

–p *page_no*

Specifies the page number that is to be transformed in the AFP document.

-r *resfile*

Specifies the fully qualified file name of the AFP resource group to be used when transforming the AFP file. For example, `afp2pdf -r c:\mydirectory\afpresfile.res afpfile.afp`.

-s (*timestamp*)

Sets a time stamp for when the document was signed.

Note

This command can be specified, but is not supported in the current implementation. It exists for compatibility purposes only.

The format of the time stamp is:

```
YYYMMDDHHmmSSOHH'mm'
```

where:

- *YYYY* is the year.
- *MM* is the month.
- *DD* is the day (01-31).
- *HH* is the hour (00-23).
- *mm* is the minute (00-59).
- *SS* is the second (00-59).
- *O* is the relationship of local time to Universal Time (UT), denoted by one of the characters +, -, or Z.
- *HH* is the absolute value of the offset from UT in hours (00-23).
- *mm* is the absolute value of the offset from UT in minutes (00-59).

For example, December 23, 1998, at 7:52 PM, U.S. Pacific Standard Time, is represented by the string `19981223195200-08'08'`.

-t *rotation*

Specifies the rotation value to use when transforming the file. Valid values are **0**, **90**, **180**, and **270**. The default is **0**.

Some AFP files might have already been formatted with a rotated orientation. In this case, use this parameter to align the text in an upright position.

Note

- The transform applies the rotation to the entire page, including new images added with image mapping.

-u *password*

Specifies an alphanumeric password that gives permission to open the PDF document (also known as a user or document open password).

-v *mapfile*

Specifies the location and the name of the image map configuration. This file name does not use relative paths and is fully qualified. See Creating the image map configuration file, p. 62, for more information about the image map configuration file.

inputfile

Specifies the AFP input file that is to be transformed to PDF. This parameter is required.

Return Codes

The afp2pdf client command returns one of these values:

0

The transform completed successfully.

Nonzero

An error occurred.

AFP2PDF Plus Transform Options File

Parameters to control settings for the AFP2PDF Plus Transform are specified in an options file. By default, the name of this file is `a2pxopts.cfg`. When running the transform function with the afp2pdf client command-line program, the options file must reside in the same directory as the program module.

If running the transform function from the API, you can locate the options file in any directory on the system and you can use any file name. By setting the **szOptionsFile** option in the structure passed to the transform function, you can specify different option files for different types of documents.

Note

- The format of the `a2pxopts.cfg` file has changed. All available options are listed in the file, along with an explanation of each option. Uncomment the option you wish to use and add any required parameters. Where appropriate, you can manually specify multiple options of the same type. See Comparison with AFP2PDF Transform, p. 77 for a comparison list of AFP2PDF vs. AFP2PDF Plus transform options.
- Parameters in the options file must be specified on separate lines and have the format `parameter=value`. For example:

```
Disable_Compression=True
Auto_Rotate=True
```

- Variable text should be enclosed in double quotes if the value contains blanks. For example, `Subject="Advance Function Presentation"`.
- Parameters and values are not case-sensitive.
- Lines starting with a semicolon (;) or a pound (#) character are comments.
- Previous versions of the options file work with AFP2PDF Plus.

The AFP2PDF Plus Transform option parameters are:

Append_PDF_File=*file*,*n*[,"[*tm1 tm2 tm3 tm4 tm5 tm6*]"]

Indicates where or how the first page of the specified PDF file is appended to the generated PDF file, where:

file

Specifies the name and location of the PDF file that is to be appended to the generated PDF file.

n

Specifies where the PDF is appended or if the PDF is used as a background form or overlay. Values are:

0

Appended to the beginning of the generated file.

1

Appended to the end of the generated file.

2

Used as a preprinted form on all pages of the generated file.

3

Used as a preprinted form on even pages of the generated file.

4

Used as a preprinted form on odd pages of the generated file.

5

Used as a preprinted form on the first page of the generated file.

6

Used as a preprinted form on all pages of the generated file, except for the first page.

7

Used on top of the content from all pages of the generated file.

8

Used on top of the content from the even pages of the generated file.

9

Used on top of the content from the odd pages of the generated file.

10

Used on top of the content from the first page of the generated file.

11

Used on top of the content from all pages of the generated file, except for the first page.

tm1 tm2 tm3 tm4 tm5 tm6

Specifies the linear transformation matrix values that define the origin, rotation, or scale of the appended PDF file. For example:

1 0 0 1 *ox oy*

Specifies origin changes, where:

ox

Specifies the distance to translate the horizontal dimension.

oy

Specifies the distance to translate the vertical dimension.

sx 0 0 *sy* 0 0

Specifies scaling changes, where:

sx

Specifies the scale factor (1.0-100%) for the horizontal direction.

sy

Specifies the scale factor (1.0-100%) for the vertical direction.

cos θ sin θ -sin θ cos θ 0 0

Specifies rotation changes by an angle θ counterclockwise.

The origin for this operation is the bottom left-hand corner of the page.

For example, `Append_PDF_File=C:\term.pdf,0` appends the `c:\term.pdf` file to the beginning of the generated PDF file, while `Append_PDF_File=C:\term.pdf,1` appends the `c:\term.pdf` file to the end of the generated PDF file.

Append_PDF_File2=*file,n*[,"[*tm1 tm2 tm3 tm4 tm5 tm6*]"]

Indicates where or how the first page of the specified PDF file is appended to the generated PDF file. This parameter is the same as **Append_PDF_File** except that the origin is the top left-hand corner of the page

Author=*name*

Specifies 1–62 characters for the author name in the Info Dictionary of the output PDF file. This information is displayed to the user if the **File** → **Properties** function is selected in Adobe Acrobat.

Auto_Rotate=True | False

Indicates whether the transform determines the orientation of each page and rotates it so that the text appears right-side up. By default, the AFP document is converted as is, so if the AFP page is formatted in a rotated orientation, the output PDF is also rotated. Setting this parameter to **True** is useful when pages in a document are rotated with different orientations.

If a document rotation setting (an input parameter to rotate the entire document) is also given, the **Auto_Rotate** parameter overrides this setting.

Auto_Rotate_CountBlanks=True | False

Indicates whether blanks are counted in the auto-rotate algorithm. This parameter is used only with **Auto_Rotate=True**.

Auto_Rotate_ParseOverlays=True | False

Indicates whether page overlays are parsed in the auto-rotate algorithm. This parameter is used only with **Auto_Rotate=True**.

Auto_Rotate_UseCharacterRotation=True | False

Indicates whether character rotation is used along with the text orientation in the auto-rotate algorithm. This parameter is used only with **Auto_Rotate=True**.

AVE_Control_File=*file*

Specifies the fully qualified name of the AFP Visual Environment control file. This control file contains the names and parameters of the Pipeline Manager filter programs to be run on the AFP before conversion to PDF. See Working with AFP Visual Environment Control Files, p. for information about their use.

Bookmarks_Csname=*font character set name*[,*font character set name...*]

Specifies that bookmarks are created from strings of text that use the font character sets listed.

You can include multiple values, but they must be separated by commas. For example, `Bookmarks_Csname="C0FONT01","C0FONT02"`.

When **Bookmarks_Csname** is specified and a match is found, **Show_Pageids** is ignored.

For more information on bookmarks, see the description of **Disable_Bookmark_Generation**.

Cache_AFP_Overlay=True | False

Indicates whether AFP overlays are saved during transformation. The default setting this parameter is **True**, which can result in a smaller PDF file and a faster conversion.

Certificate=*pkcs12_file*,*pkcs12_password*[,*cache_directory*]

Specifies the PKCS#12 certificate file used to sign the document, where:

pkcs12_file

Specifies the fully qualified path name of the PKCS#12 certificate file.

pkcs12_password

Specifies the password needed to read the PKCS#12 file.

cache_directory

Specifies the fully qualified path to a directory that is used for certificate storage. This is an optional value that can be used to improve the performance of the conversion.

Certificate2=*pkcs12_file*,*pwd_file*[,*cache_directory*]

Specifies the PKCS#12 certificate file used to sign the document.

pkcs12_file

Specifies the fully qualified path name of the PKCS#12 certificate file.

pwd_file

Specifies the fully qualified path name of a file that contains the password needed to read the PKCS#12 certificate file.

cache_directory

Specifies the fully qualified path to a directory that is used for certificate storage. This is an optional value that can be used to improve the performance of the conversion.

Color=*afpname*,*red*,*green*,*blue*

Specifies the RGB value color setting that overrides the display of named color values in the AFP OCA (graphics objects), where:

afpname

Specifies one of the colors defined in AFP: BLUE, BROWN, CYAN, DARKBLUE, DARKCYAN, DARKGREEN, GREEN, GREY, HIGHLIGHT0, HIGHLIGHT1, HIGHLIGHT2, HIGHLIGHT3, MAGENTA, MUSTARD, ORANGE, PURPLE, RED, WHITE, YELLOW.

red

green

blue

Specifies the RGB value settings in the range of 0 to 255.

For example, the AFP named color BLUE changes to a light gray color with these values: Color=BLUE, 220, 220, 220.

To override text (PTOCA) named color values, include **Modify_Text_Colors**. For more information, see the **Modify_Text_Colors** parameter.

To override bar code objects (BCOCA) named color values, include **Modify_BCOCA_Colors**. For more information, see the **Modify_ BCOCA_Colors** parameter.

Concatenate_PDF_File=*file*

Indicates that the entire specified PDF file is appended to the end of the generated PDF file, where:

file

Specifies the name and location of the PDF file that is to be appended to the generated PDF file.

Convert_Text_CMYK=True | False | Legacy

Indicates whether all AFP text defined with the CMYK color space are converted to the RGB color space in the PDF output, when the **Preserve_CMYK** parameter is not specified or set to false. Valid values for this parameter:

- **False**: for no conversion.
- **True**: converts color space values from CMYK to RGB.
- **Legacy**: uses the CMYK to RGB formula from the legacy AFP2PDF transform.

See **Preserve_CMYK** for more information on the APF image conversion to RGB.

Creator=*name*

Specifies 1–62 characters for the creator name in the Info Dictionary of the output PDF file. This information is displayed to the user if the **File** → **Properties** function is selected in Adobe Acrobat.

Default_Encryption_Permissions=[*Acrobat_functions*[,*n*[,*encryption_functions*]]]

Sets one or more default encryption permissions to be used when encrypting the PDF output. The permission codes are:

Acrobat_functions

a

Adding or changing annotations or form fields in Acrobat is denied.

c

Changing the document in Acrobat is denied.

p

Printing the document in Acrobat is denied.

s

Selecting and copy text and graphics in Acrobat is denied.

n

Specifies the level of encryption:

5

RC4 128-bit encryption

6

AES 128-bit encryption

7

AES 256-bit encryption

encryption_functions

You can set one or more of these codes with 128-bit and 256-bit encryption:

e

Extracting text and graphics in Acrobat is denied.

d

Assembling text and graphics in Acrobat is denied.

i

Editing form fields in Acrobat is denied.

q

Printing high quality in Acrobat is denied.

Note

- Using RC4 128-bit encryption produces a file that can only be opened with Acrobat 5.0 or later.
- Using AES 128-bit encryption produces a file that can only be opened with Acrobat 7.0 or later.
- Using AES 256-bit encryption produces a file that can only be opened with Acrobat 10.0 or later.
- Codes **5**, **6**, and **7** can be used in combination with one of the **a**, **c**, **p**, or **s** codes to force 128-bit or 256-bit encryption without setting the **i**, **e**, **d**, or **q** codes.

Default_Linearization=True | False

Indicates whether the PDF output file is linearized (or enabled for Fast Web Viewing).

See the **-m** parameter from Parameters, p. 22 for more information about linearization.

Default_Owner_Password=*password*

Specifies a default alphanumeric owner encryption password.

Default_User_Password=*password*

Specifies a default alphanumeric user encryption password.

Disable_Bookmark_Generation=True | False

Indicates whether PDF bookmarks are created. If page level indexing information is available in the input AFP document, PDF bookmarks are automatically generated. If you do not want bookmarks created, set this parameter to **True**.

Disable_Compression=True | False

Indicates whether the transform compresses PDF data. Setting this parameter to **True** to turn off compression can be useful when trying to determine problems that might exist in the PDF output file.

DotDensity,*patternid*,*afpcolorname*=*red*,*green*,*blue*

Specifies the RGB value color setting for the AFP GOCA pattern that is used to fill the interior of an area, where:

patternid

Specifies the GOCA pattern identifier, which defines various dot fill patterns of varying density. Pattern ID supports values from 1 to 8.

afpcolorname

Specifies one of the pattern foreground colors defined in AFP: BLACK, BLUE, BROWN, CYAN, DARKBLUE, DARKGREEN, DARKTURQUOISE, DEFAULT, GREEN, GREY, MAGENTA, MUSTARD, ORANGE, PURPLE, RED, WHITE, YELLOW.

red

Specifies the red value settings in the range of 0 to 255.

green

Specifies the green value settings in the range of 0 to 255.

blue

Specifies the blue value settings in the range of 0 to 255.

For example: **DotDensity,5,GREY=220,220,220** specifies the RGB value of 220, 220, 220 for GOCA pattern 5 with a grey foreground color.

Enable_Auto_Font_Image=True | False

Indicates whether a font is mapped automatically. Setting this parameter to **True** causes the font to be converted to image data if the font resources exist and a font mapping does not exist.

Enable_Auto_Image_Cache=True | False

Indicates whether images are saved.

FIPS_Mode=True | False

Indicates whether the transform functions in an environment that is compliant with the Federal Information Processing Standard (FIPS). When set to **True**, encryption options and digital certificate options are disabled. For example, any request to add user passwords or owner passwords to the PDF file and any attempt to digitally sign a PDF file are ignored.

Flate_Compression_Level=*n*

Specifies the level of the Flate compression used in the PDF file, where *n* is 0 to 9. The larger the number the longer the conversion takes, but a smaller PDF file should be created. The default is 6.

FontExt=*.extension*

Specifies the file extension that the transform uses to search for font resources in resource directories. For example, if FontExt=*.*EXT* and the AFP document reference a font named C0FONTCS, the transform searches for a file named C0FONTCS.EXT in the resource directories.

Note

- The *extension* value is case-sensitive on most operating systems.
- If you add this parameter to the configuration file, the transform starts the search with the customized extension. If not set or not found, AFP2PDF Plus searches for the file extension by default, using this search order: no extension, *.OLN*, *.FONTOLN*, *.240*, *.FONT3820*, *.FONT38PP*, *.CFT*, *.CDP*, *.300*, *.FONT300*.

Font_Path=*directory*

Specifies the base directory that the transform users to search for the font configuration files. By default, this base directory is the `\font` subdirectory where the transform modules were installed. If the parameter is used when the font configuration files have been moved to a different location, the

same directory structure of the font configuration files must be preserved. For example, if this entry was given: `Font_Path=c:\fontfils`, the code page map files must reside in the `c:\fontfils\maps` directory.

Generate_Type3=True | False

Indicates whether Type 3 fonts are generated in the PDF output instead of the raster images of font characters. Type 3 fonts are raster fonts that can be used to display the exact AFP raster font characters. For more information, see Using Custom AFP Raster Font Files, p. 55.

Note

- If the AFP font was generated with the standard AFP Graphic Character Global Identifiers (GCGIDs), the Type 3 font uses the searchable text. You can use the text search function in a PDF to find any text that uses a Type 3 font and standard IDs.

GOCA_Use_Circles=True | False

Indicates whether the transform uses circles or dots for GOCA patterns 1 through 8, instead of shaded areas. Setting this parameter to **True** increases the PDF file size.

Note

- **DotDensity** overrides the **GOCA_Use_Circles** setting, for the **DotDensity** specific pattern.

GOCA_Pattern=*patternid,var1,var2*

Specifies the characteristics for the AFP Graphics Object Content Architecture (GOCA) pattern that is used to fill the interior of an area, where:

patternid

Specifies the GOCA pattern identifier, which defines various line fill patterns. Pattern ID supports values from 9 to 14.

var1

Defines the line width.

var2

Defines the line spacing.

For example: **GOCA_Pattern=*9,10,250*** specifies GOCA pattern 9 with a line width of 10 and a line spacing of 250.

Honor_Constant_Forms=True | False

Indicates whether the transform generates extra pages created from the constant forms control function, which is a function in AFP architecture that produces one or more static overlays on a page without variable data from the print file.

Honor_Medium_Orientation=True | False

Indicates whether the transform applies the medium orientation specified in the form definition to created PDF pages.

Horizontal_Offset=*n*

Applies a horizontal offset to all the data on the page relative to the paper or media represented in the PDF. The units of this setting are 1440 units per inch.

Ignore_Bar_Code_Object_Area_Size=True | False

Indicates whether the object area size of a bar code object is ignored when checking if the generated bar code symbol fits within a specified area. By default, the object area size of the bar code is used to check if the generated bar code symbol is larger than the specified area to fit.

Note

- Setting this parameter to **True** ignores the area size of the bar code object and increases the size of the bar code symbol.

Ignore_Data_Font_Height=True | False

Indicates whether the transform ignores the font height. By default, the transform uses the font height value specified in the AFP data. If this parameter is set to **True**, the font height setting in the AFP data is ignored and the font size specified in the font configuration file `csdef.fnt` is used.

Ignore_PTOCA_STC=True | False

Indicates whether the color specified with the Set Text Color (STC) control to display text in Presentation Text Object Content Architecture (PTOCA) is honored. When **True** is specified, the STC-specified color is ignored and black is used.

ImageMapEntries_File=*file*

Specifies the file that the transform uses to output the image information from the AFP file. The information in this file can then be modified and passed as input back into the transform function to change the conversion method for individual AFP images. See Mapping AFP images, p. 61 for more information about the mapping images.

JPEG_Quality_Level=*n*

Specifies the JPEG quality level, where 0.0 is the least quality and maximum compression and 1.0 is the best quality with no compression. The default is 0.95.

Keywords=*text*

Specifies 1 to 120 characters for the Keywords entry in the Info Dictionary of the output PDF file. This information is displayed to the user if the **File** → **Properties** function is selected in Adobe Acrobat.

Layer=*n*

n

Specifies the default layer where **Shaded_Area** is placed, in relation to the text of the enclosing AFP object. Values are:

1

Bottom: Place **Shaded_Area** under the entire text.

2

In-place: Place **Shaded_Area** in the same position as the image that it is replacing.

3

Top: Place **Shaded_Area** on top of the entire text.

For more information about shaded areas, see Substituting AFP shaded images with colored areas, p. 65.

Locale_Path=*path*

Specifies the path location of the locale-specific information files needed during the transform process. Only use this parameter when converting multiple-byte language files (Traditional Chinese, Simplified Chinese, Japanese, Korean, or Unicode).

Logging=[True | Append], *logDir, lvl*

Specifies the type of tracing that is turned on for program debugging and where the log file is located. When set to **True**, logging traces on a document boundary and overwrites the trace data from a previous document in the log file. When set to **Append**, the logging process appends data to the end of the log file. The *logDir* setting specifies the fully qualified directory where the log file is saved. The log file name is `Afp2Pdf.log`. The *lvl* setting specifies the log level: **SEVERE**, **WARNING**, or **INFO**. The default is **WARNING**. This command overrides the settings in the `logging.properties` file.

Modify_BCOCA_Colors=True | <u>False</u>

Indicates whether RGB value color settings override the display of named color values in the AFP Bar code objects (BCOCA). For more information, see the **Color** parameter.

Modify_Text_Colors=True | <u>False</u>

Indicates whether RGB value color settings override the display of named color values in the AFP text (PTOCA). For more information, see the **Color** parameter.

Output_IndexInfo=True | <u>False</u>

Indicates whether the group level index information of the document is placed as a bookmark with this format:

```
index attribute : index value
```

Other applications processing the PDF file can search for the bookmark text.

Note

- If the parameter is set to **True** on password protected files, the bookmark text becomes encrypted.

Output_IndexInfo2=True | <u>False</u>

Indicates whether the group level index information of the document is placed as a bookmark with this format:

```
index attribute
      index value1
      index value2
```

Other applications processing the PDF file can search for the bookmark text.

Note

- If the parameter is set to **True** on password protected files, the bookmark text becomes encrypted.
- This parameter overrides **Output_IndexInfo**, if both **Output_IndexInfo** and **Output_IndexInfo2** are specified.

OverlayExt=*.*.*extension*

Specifies the file extension that the transform uses to search for overlay resources in resource directories. For example, if OverlayExt=*.*OVE* and the AFP document reference an overlay named O1OVERLY, the transform searches for a file named O1OVERLY.OVE in the resource directories.

Note

- The *extension* value is case-sensitive on most operating systems.
- If you add this parameter to the configuration file, the transform starts the search with the customized extension. If not set or not found, AFP2PDF Plus searches for the file extension by default, using this search order: no extension, *.OVLY3820*, *.OVLY38PP*, *.OVL*, *.OLY*.

Page_rotation=0 | 90 | 180 | 270

Specifies the rotation value to use when transforming the file. Some AFP files might have already been formatted with a rotated orientation. In this case, you must use this parameter to align the text in an upright position.

4

PageSegExt=*.*extension*

Specifies the file extension that the transform uses to search for page segment resources in resource directories. For example, if PageSegExt=*.*SEG* and the AFP document reference a page segment named S1PAGSEG, the transform searches for a file named S1PAGSEG.SEG in the resource directories.

Note

- The *extension* value is case-sensitive on most operating systems.
- If you add this parameter to the configuration file, the transform starts the search with the customized extension. If not set or not found, AFP2PDF Plus searches for the file extension by default, using this search order: no extension, *.PSEG3820*, *.PSEG38PP*, *.PSG*, *.PSE*.

PDF/A=True | False [,conformance level]

Indicates whether the transform generates output that conforms to the PDF/A-1b, PDF/A-2b, or PDF/A-3b specification. The accepted values for the optional parameter **[conformance level]** are **1b**, **2b**, or **3b**.

Note

The default conformance level is **1b** when the value is missing or the value specified after **True** is not accepted.

When this parameter is set to **True**, these conditions must be met for the transform to generate output that conforms to the specification:

- All fonts must be embedded for PDF/A-1b, PDF/A-2b, or PDF/A-3b compliance. See Working with AFP Resources, p. 45 for information about embedding fonts; see Using Custom AFP Raster Font Files, p. 55
- The PDF file must not contain any passwords or encryption; therefore, none of the functions described in Encryption, p. 18 should be used when this parameter is set.

PfmPfb_Directory=directory

Specifies the directory that the transform uses to search for PFB and PFM (or AFM) files when mapping AFP fonts to Type1 fonts. If not set, the default directory is "Font_Path/Type1".

Preserve_CMYK=True | False

Indicates if all AFP images are converted to the RGB color space in the PDF output. Setting this parameter to **True** preserves AFP images defined with the CMYK color space so that they are not converted to RGB. This parameter uses a compression in the PDF that does not cause any loss of color quality.

PTX_Drawrule_Use_Line=True | False

Indicates whether the transform uses line operators in the generated output for PTX draw rules (DIR and DBR). When this parameter is set to **False**, the system uses rectangle operators. For some PDF viewers this can impact the way the thin lines are rendered at different zoom levels. The default value is **False**.

ResourceDataPath=*directory*[;*directory*...]

Specifies the directories that the transform uses to search for AFP resources. You can specify multiple directories, but they must be separated with a semicolon (;). See Working with AFP Resources, p. 45 for more information.

Shade_RGB=*red, green, blue*

Specifies the intensity of the red, green, and blue colors when generating the color of shaded areas. The color values are in the range of 0.0 to 1.0, with 0.0 for black and 1.0 for white. For example: Shade_RGB=0.5, 0.5, 0.5.

See Substituting AFP shaded images with colored areas, p. 65 for more information about shaded areas.

Show_Outline=True | False

Indicates whether the outline window is displayed. If an AFP document contains index data, the transform converts this index information into PDF outline and bookmark functions. If the output PDF file contains any bookmark information, the outline window is always displayed when viewed with Adobe Acrobat. This parameter can be set to **False** so the outline window is not displayed.

Show_Pageids=True | False

Indicates whether the Page Identifier values are displayed in the bookmark pane.

Static_Paper_Length=*n*

Overrides the AFP logical page size used by the default for the page or media dimensions represented in the PDF. Using 72 units per inch, you can set a specific paper length for the entire document.

Static_Paper_Width=*n*

Overrides the AFP logical page size used by the default for the page or media dimensions represented in the PDF. Using 72 units per inch, you can set a specific paper width for the entire document.

Subject=*text*

Specifies 1–62 characters for the Subject entry in the Info Dictionary of the output PDF file. This information is displayed to the user if the **File** → **Properties** function is selected in Adobe Acrobat.

Substitute_Default_Medium_Map_Allowed=True | False

Allows the use of the default Medium Map when processing the AFP pages, if the Medium Map requested with the Invoke Medium Map (IMM) or Begin Page (BPG) structured field is not found in the active form definition.

Title=*text*

Specifies 1–62 characters for the Title entry in the Info Dictionary of the output PDF file. This information is displayed to the user if the **File** → **Properties** function is selected in Adobe Acrobat.

Transform_All_Subgroups=True | False

Specifies whether the transform converts all of the subgroup formatting from the AFP form definition resource. By default, the transform processes the first subgroup.

TrueType_Directory=*directory*

Specifies the directory that the transform uses to search for TrueType font files when mapping AFP fonts to TrueType fonts. If not set, the default directory is "Font_Path/TrueType".

UDC_Range=*low1,high1[,low2,high2][,low3,high3][,low4,high4]*

Specifies one to four section ranges that the transform can search when processing user-defined characters for double-byte character set (DBCS) text. The ranges restrict which DBCS sections are searched, which causes the transform to run more efficiently.

For example, if the user-defined characters are present in sections 129, 130, and 131, the parameter can specify these low and high sections: UDC_Range=129,131.

Use_Default_Hash_Code

Specifies whether the transform uses a default method to determine if two page segments or overlays are identical, based on the resource name and content. The alternative is an optimized method where two overlays are equal when they have the same content, even if they are named differently. Setting this to **True** can improve performance in some cases. The default value is **False**.

Vertical_Offset=*n*

Applies a vertical offset to all the data on the page relative to the paper or media represented in the PDF. The units of this setting are in 1440 units per inch.

5. Using the AFP2PDF Plus Utility Programs

- **split_afp2pdf Command**
- **ArchiveLoad_afp2pdf Command**

The following utilities for working with AFP files are included with the APF2PDF Transform Plus program. The jar file a2psplit.jar contains two utilities: split_afp2pdf and Archiveload_afp2pdf.

split_afp2pdf Command

The split_afp2pdf command takes an indexed AFP print file, splits it up into separate statements, and invokes the AFP2PDF Plus transform. The input AFP data must be divided into separate page groups (statements) and contain AFP index values for each group. This type of data is created either by the AFP Conversion and Indexing Facility (ACIF), the AFP Visual Environment (AVE) or an equivalent function.

The split_afp2pdf command creates a separate PDF output file for each statement. The name of the output file corresponds to at least a single associated index value. For example, if the index field **ACCOUNT** is selected, each output file is named with the actual account ID (such as `123456.pdf`).

A Windows batch file, split_afp2pdf.bat, or a UNIX executable script, split_afp2pdf, invokes the JAVA application to split the AFP file. The command invokes the AFP2PDF Plus Transform for each statement so the same configuration and setup for the transform program applies. The split_afp2pdf module must be placed in the same directory where the AFP2PDF Plus Transform files are installed.

Syntax

```
split_afp2pdf [-d outputpath] [-e password] [-f fdeffile]
[-g codepage] [-h fontdefpath] [[-i index1  [-i2 index2
[-i# index# [... [-i10 index10]]]]]|[-ix hexindex1 [-ix2 hexindex2
[-ix# hexindex# [... [-ix10 hexindex10]]]]]] [-indexafporder]
[-k path]  [-legacylog ]  [-n ] [-nt nThreads][-o optfile]
[-p functions] [-r resfile] [-t rotation] [-temp] [-tle]
 [-u password] inputfile
```

Parameters

-d *outputpath*

Specifies the directory for all the output files. If this parameter is not specified, the output files are placed in the current directory.

-e *password*

Specifies an alphanumeric password that gives permission to change document security settings (also known as a master, owner, or permissions password). For example, if the printing function in the PDF display has been restricted (`-p p`), you must supply a password to override the setting. The same password is applied for each output file.

-f *fdeffile*

Specifies the fully qualified name of the form definition resource that is used on each statement when transforming the AFP document.

-g *codepage*

Specifies the code page global identifier (CPGID) to be used when interpreting the index information in the AFP file. By default, code page 500 is used to interpret all index fields. See Code Page File Map, p. 52 on page 45 for more information.

-h *fontdefpath*

Specifies the location of the font definition files when a code page ID is specified with the **-g** parameter. Use the **-h** parameter if the font definition files are not located in the `\font` subdirectory of the current directory.

-i *index1*

Specifies the index field used for generating the output file names. For example, if `-i RouteID` is specified and if a statement has a routing ID of ABC, the output file name ABC is generated.

The user must know which index fields are available for the AFP data and must pick a field that is unique for each statement. For example, a single index field, such as **ZIPCODE**, might not be have unique values for all statements. In that case, statements with the same ZIP Code would have the same file name, causing an overwrite of the output file.

If this parameter is not given, the first index field found in the AFP data is used. If an index field name not available in the AFP data is given, no corresponding PDF file is generated.

-i2 *index2* **to -i10** *index10*

Specifies the index field number used for generating the output file names. The index attributes range from 2 to 10.

Note

- You can specify either all -i or -ix type parameters.

-ix *hexindex1*

Specifies the index field used for generating the output file names. This parameter is the same as the **-i** parameter except that is used when the index name is specified in hexadecimal notation.

-ix2 *hexindex2* **to -ix10** *hexindex10*

Specifies the index field number used for generating the output file names. The index attributes range from 2 to 10.

-indexafporder

Organizes the entries from the index file in the same document order as the input AFP file.

inputfile

Specifies the AFP input file that is to be split and transformed to PDF. This parameter is required.

-k *path*

Specifies a fully qualified path of an alternate location for the log files generated during the transformation. Each corresponding PDF output file comes with an associated log file if errors occur during transformation. The log file name is *output_PDF_file_name.pdf.log*. The corresponding log file will not be generated if there are errors generated on the associated statement.

-n

Generates a different PDF output file name if the file already exists. For example, the PDF output file `ABCD__2.pdf` is generated if the output file name `ABCD.pdf` already exists.

-legacylog

Indicates that the log file format for `split_afp2pdf` and `legacy split_afp2pdf` should match.

-nt *`nThreads`*

Specifies the number of threads for the transform to run. The default is 4.

-o *`optfile`*

Specifies the location and name of the transform options file. The default is `a2pxopts.cfg` in the same directory as the program module. This file name must not use relative paths and must be fully qualified. See AFP2PDF Plus Transform Options File, p. 25 for more information about the options file.

-p *`functions`*

When used with an owner password (**-e** *`password`*), specifies which PDF display functions are restricted. The codes for the display functions, which can be used in any order, are:

a

Add or modify text annotations and interactive form fields.

c

Modify the document contents.

d

Copy text and graphics from the document.

p

Print the document.

-r *`resfile`*

Specifies the fully qualified file name of the AFP resource group to be used when transforming the AFP file. For example, `afp2pdf -r c:\mydirectory\afpresfile.res afpfile.afp`.

-t *`rotation`*

Specifies the rotation value to use when transforming the file. Valid values are **0**, **90**, **180**, and **270**. The default is **0**.

Some AFP files might have already been formatted with a rotated orientation. In this case, use this parameter to align the text in an upright position.

-temp

Specifies the use of a temporary file to hold the statement for transformation. In certain circumstances, the input statement may exceed the available JAVA memory space. The temporary file is created in the directory specified by **–d** and is deleted when the transform is finished, whether successful or not. The number of concurrent temporary files created is twice the number of threads. Use this parameter only when Support tells you to do so.

-tle

Generates an xml formatted output file that contains the index information for each PDF file output from the split_afp2pdf program. The default output file name is: `\output_dir\input_afp_file_name.xml`.

-u *password*

Specifies an alphanumeric password that gives permission to open the PDF document (also known as a user or document open password).

The XML Format of Output TLE Information when -tle is Specified

```
<?xml version="1.0" encoding="UTF-8"?>
<Split_afp2pdf_plus>
     <Document>
          <Filename> output pdf file name</Filename>
          <TLE>
                <Name>name of the TLE pair</Name>
                <Value>value of TLE pair</Value>
          </TLE>
          … more <TLE> </TLE> clauses …
     </Document>
     … more <Documents> </Document> clauses …
</Split_afp2pdf_plus>
```

Return Codes

The split_afp2pdf command returns one of these values:

0

The transform completed successfully.

Nonzero

An error occurred.

split_afp2pdf generates a summary log file containing the list of the output files generated. The log file is placed either in the current directory or the directory specified by **–d**. This log also contains any warning messages created during the conversion. For example, if an output file was overwritten, it would be indicated by a warning message in this log file. The log also contains the total number of statements processed, along with an overall error return code for the **split_afp2pdf** process.

Additionally, an individual error log is generated for each AFP statement that encounters an error during transform processing.

ArchiveLoad_afp2pdf Command

The Archiveload_afp2pdf command is designed to convert AFP into PDF for efficient loading into Enterprise Content Management (ECM) systems, including IBM's Content Manager OnDemand. It uses page group level Tag Logical Elements (TLEs) to determine document boundaries and then converts

each document to PDF. The resulting documents are concatenated into one PDF object. An output index file is created that contains the byte level offset and extent, along with the TLE index information for each document in the PDF object. This index file is available in an XML format or an IBM's CMOD Generic Indexer format.

A Windows batch file, Archiveload_afp2pdf.bat, or a UNIX executable script, Archiveload_afp2pdf.sh, invokes the JAVA application to create the PDF object and the output index file. The command invokes the AFP2PDF Plus Transform for each statement so the same configuration and setup for the transform program applies. The Archiveload_afp2pdf.bat module must be placed in the same directory where the AFP2PDF Plus Transform files are installed.

Syntax

archiveload_afp2pdf[**-d** *outDir path*] [**-c** *opt file name*] [**-I** *ind file name*] [**-o** *pdf file name*] [**-t** *rotation* [**-r** *res file*] [**-temp** *temp dir* [**-nt** *nThreads* [**-h***font directory*] [**-utf8**] [**-z** *ind config file* [**-log** *log file*] [**-odload**] [**afpfile**]

Parameters

-c *opt file name*

Specifies the fully qualified or relative path of the option file name for the afp2pdf transform. This parameter is optional.

-d *outDir path*

Specifies the directory for the output files. If this parameter is not specified, the output files are placed in the current directory. This parameter is optional.

-h *font directory*

Specifies the location of the font directory. This parameter is optional.

-log *<path>*

Specifies the log file name. The default log file name is `<outDir path>/afp` with extension ".log" without ".afp". This parameter is optional.

-i *ind file name*

Specifies the index field used for generating the output file names. The default file name is `<outDir path>/afp` with extension ".xml" without ".afp".

AFP2PDF ignores this flag if *-odload* is specified. This parameter is optional.

-nt *<nThreads>*

Specifies the number of threads that run concurrently for the transform. The default number of threads is 8. This parameter is optional.

-o *output name*

Specifies the output archive data file name. The default file name is <outDir path>/afp with extension ".pdf" without ".afp". If *-odload* is specified, the default output archive data file name is *<outDir path>/<afpfile>.ARD.out*. This parameter is optional.

-r *resfile*

Specifies the fully qualified or relative path of the external resource file for the afp2pdf transform. This parameter is optional.

-t *rotation*

Specifies the rotation value to use when transforming the file. Valid values are **0**, **90**, **180**, and **270**.

The default value is **0**. This parameter is optional.

-z *ind config file*

Specifies the fully qualified or relative path of the index manipulation configuration file for the afp2pdf transform. This parameter is optional.

The configuration file contains this list:

Index_AFPOrder

Arranges the entries from the index file in the same document order as the input AFP file.

Index_Remove, indexFieldName

Index_Remove removes the index that matches the value specified as the indexFieldName.

Index_Rename, indexFieldName, NewIndexFieldName

Index_Rename renames the index that matches the value specified for indexFieldName to the value specified for NewIndexFieldName.

-afpfile

Specifies the fully qualified or relative path name of the input afp file. For example, `ArchiveLoad_afp2pdf -c opt.cfg -o sample_out.pdf -i sample_index.xml sample.afp`. This parameter is required.

-odload

Generates -odload output files and formats. This parameter is optional. The default -odload output files are:

1. **output data file** <outDir path>/<afpfile>.ARD.out
2. **index file** <outDir path>/<afpfile>.ARD.ind
3. **trigger file** <outDir path>/<afpfile>.ARD

-temp

Specifies the use of a temporary space for each statement instead of memory buffer. It is used to avoid from using up memory heap for huge statement. If *-temp* is specified, the default temporary file is <output directory>/temp. This parameter is optional.

-utf8

Use UTF-8(code page 1208) as the text encoding for output index file. The default is to use JVM "file.encoding" property. This parameter is optional.

6. Working with AFP Resources

- **Files Supplied for Mapping Fonts**
- **Process for Mapping Fonts**
- **Using Custom AFP Raster Font Files**
- **Using Custom Font Metric Files**
- **Mapping AFP Fonts to Type 1 Fonts**
- **Mapping AFP Fonts to TrueType Fonts**
- **Using AFP TrueType Fonts**
- **Using User Defined Characters within a DBCS Font**
- **Character encoding**

The AFP resources used by the AFP2PDF Plus Transform include:

- Page segments
- Overlays
- Form definitions
- Character set raster and outline font files.

The page segment, overlay, form definition and font file resources can be passed to the transform from these locations:

Inline resource group

The AFP resources needed by the AFP data file are combined into a logical resource library for the document. This resource group is contained in the AFP file along with the AFP document.

External resource group

The AFP resources needed by the AFP data file are combined into a logical resource library for the document and are passed to the transform as a separate file.

The afp2pdf command specifies this resource file with the **–r** parameter. See Parameters, p. 22 for more information.

Resource directories

AFP resource files can be placed in specific directories that the transform program searches for when converting a document. The user can specify multiple directories and the directories are searched in the order that they are given.

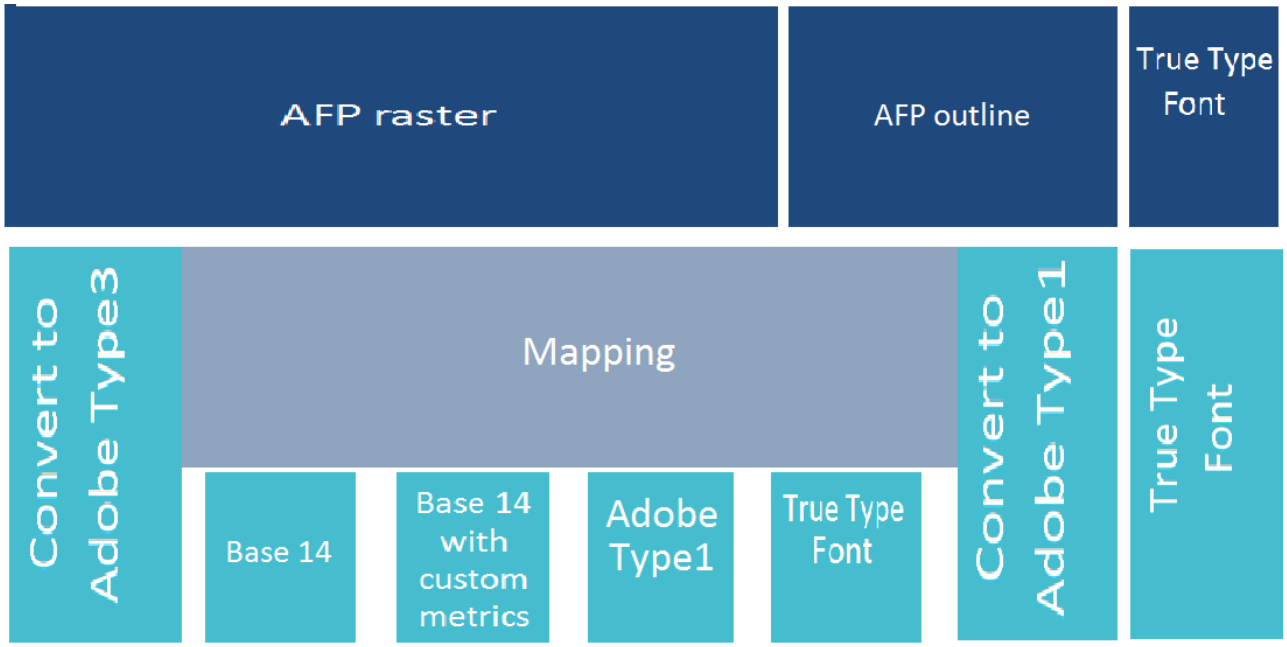
By default, resources are placed in a `\resource` subdirectory where the transform code modules were installed. You can specify other directories with the **ResourceDataPath** parameter in the transform options file. See AFP2PDF Plus Transform Options File, p. 25 for more information.

Note

The directory and file examples in this chapter are specified for the Windows environment. To use these examples in a UNIX environment, use the UNIX file naming convention for any file name. For example, `\font\maps` in Windows is `/font/maps` in a UNIX environment, or the input AFP file `c:\documents\afpdoc.afp` in Windows would be `/documents/afpdoc.afp` in a UNIX environment.

The AFP2PDF Plus Transform must process the AFP fonts your document was created with to fonts that can be displayed with Adobe Acrobat. This means either mapping an AFP font to one of the Base 14 fonts that Acrobat has internally defined or embedding a True Type or Adobe Type 1/Type 3 font. Font Processing Options, p. 46 shows the available ways to process fonts.

Font Processing Options



6

Files Supplied for Mapping Fonts

Font Files and Subdirectories, p. 46 lists the AFP2PDF Plus Transform font support files and the subdirectories of the installation directory in which they are installed.

Font Files and Subdirectories

File	File Name	Subdirectory	Description
Alias file	`Alias.fnt`	`\font`	Maps the font type families to PDF font name and optionally specifies the font metric file to use during the transform.
Character set definition file	`csdef.fnt`	`\font`	Defines AFP character set attributes, such as point size. It also maps the font character set to the font global identifier.
FND name defaults file	`csdef.def`	`\font`	Provides default processing for FND names.
Coded font files	`icoded.fnt`, `coded.fnt`	`\font`	Specify which AFP code page and AFP font character set make up the coded font.
Code page definition file	`cpdef.fnt`	`\font`	Maps each AFP code page to indicate which code page map file to use for the AFP2PDF Plus Transform.
Code page map file	*`cpgid`*`.cp`	`\font\maps`	Defines character identifier mappings. It matches the AFP

File	File Name	Subdirectory	Description
			code page character identifiers and their hexadecimal code points with a corresponding character identifier and ASCII code point available in the PDF environment. The default is Code Page 1252.
Font mapping file	`afpfont.fnt`	`\font`	Defines the fonts where the raster character images for the AFP font files are extracted and placed as images in the output PDF.
Custom font metric files	`Custom-metrics-xx.met`	`\font\Type1`	The default location for metric information for custom fonts (optional).

Using the `country.cfg` configuration file

The `afpfont.fnt`, `alias.fnt`, `cpdef.fnt`, `csdef.fnt`, `icoded.fnt`, and `coded.fnt` font configuration files can be renamed using a different extension by adding a `country.cfg` configuration file in the font directory. The font directory path is specified either by the **-n** flag or the *`Font_path`* entry in the `a2pxopts.cfg` file.

The contents of the `country.cfg` file is Country=*`zzz`*, where *`zzz`* can be one of these values: *`cht`*, *`chs`*, *`jpn`*, *`kor`*, *`fnt`*, matching the file name extension. The default extension is `.fnt`.

Coded Font Files

The coded font files (`coded.fnt` and `icoded.fnt`) map AFP coded fonts to their AFP character sets and code pages.

Coded Font Files

Coded Font File Name	Description
`coded.fnt`	Contains user-defined coded fonts. This file is optional, but if it exists must be placed in the `\font` subdirectory.
`icoded.fnt`	Contains standard definitions for approximately 2500 coded fonts supplied by Ricoh.

If a `coded.fnt` file exists in the `\font` subdirectory, the AFP2PDF Plus Transform searches it first for the coded fonts used in an AFP file. Example of a `coded.fnt` File, p. 48 shows an example of the contents of the `coded.fnt` file.

Example of a `coded.fnt` File

```
X?A155N2=C?A155N1,T1DCDCFS
X?AE10=C?SOAE10,T1SOAE10
X?GT10=C?DOGT10,T1DOBASE
X?ST15=C?DOST15,T1DOBASE
X?A0770C=C?A07700,T1GI0361
XOT0550C=COT05500,T1DCDCFS
```

Syntax rules

- A question mark (?) can be used only as the wildcard character for the second character in the coded font name and the character set name. This allows all the character rotations of the coded fonts to be handled with one entry while searching.

  Note

  A sequential search is performed for the coded font, and the first match (including the wildcard character) is used.
- 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.

Character Set Definition File

The character set definition file (`csdef.fnt`) specifies the character set attributes and font global identifier of the font. It is split into two sections, one for character sets (**CHARSET**) and one for font global identifiers (**FGID**).

The **CHARSET** section lists each AFP font character set and its corresponding attributes. **CHARSET** Section of the `csdef.fnt` File, p. 48 shows an example of the **CHARSET** section in the `csdef.fnt` file.

CHARSET Section of the `csdef.fnt` File

```
[CHARSET]
;charset = fgid, height, width, strikeover, underline
C?H200A0=2304,110,73,0,0
C?H200D0=2304,140,93,0,0
C?N200B0=2308,120,80,0,0
C?4200B0=416,120,144,0,0
C?DOGT15=230,80,96,0,0
C?A155A0=33207,110,73,0,0
C?A175A0=33227,110,73,0,0
C?T055D0=4407,140,93,0,0
C?T17500=4555,100,67,0,0
C?T17560=4555,60,40,0,0
DEFAULT=2308,80,0,0
```

Attribute Values for **CHARSET**, p. 49 describes the attributes and values for **CHARSET**.

Attribute Values for CHARSET

Attribute	Values	Shipped Default	Description
fgid	An FGID in one of these ranges: • 3840 to 4096 • 65260 to 65534	2308	A unique font global identifier (FGID) value, which identifies the type family, typeface, and sometimes the point size of the character set. This can be a predefined FGID or your own FGID.
height	1 to 990	80	The vertical size of the character set (minimal baseline-to-baseline value) expressed in tenths of a point. For example, a 9-point font would have a height of 90.
width	0 to 99 (currently ignored)	0	The average horizontal size of the characters in 1440th of an inch. Currently, 0 is always used because an appropriate font width is based on the height of the font.
strikeover	**1** = YES **0** = NO	0	A font whose characters all have a line, parallel to the character baseline, placed over the middle of the character.
underline	**1** = YES **0** = NO	0	A font whose characters all have a line underneath the character.

The **FGID** section lists each font global identifier and its corresponding attributes. **FGID** Section of the `csdef.fnt` File, p. 49 shows an example of the **FGID** section in the `csdef.fnt` file.

FGID Section of the `csdef.fnt` File

```
[FGID]
;fgid = familyname, style, weight, italic 230=Gothic, MODERN,MED,0
416=Courier,MODERN,MED,0
2304=Helvetica,SWISS,MED,0
2308=TimesNewRoman,ROMAN,MED,0
4407=SonoranSerif,ROMAN,MED,0
4555=SonoranSerif,ROMAN,BOLD,1
33207=SonoranSansSerif,SWISS,MED,1
33227=SonoranSansSerif,SWISS,BOLD,1
```

Attributes and Values for **FGID**, p. 50 describes the attributes and values for **FGID**.

Attributes and Values for FGID

Attribute	Values	Shipped Default	Description
familyname	Font family name reference	Times New Roman	An outline font family name or an AFP family name. "Family name" is the same as "type family" in AFP fonts and "typeface name" in the PDF environment.
style	**SWISS**, **ROMAN**, **SCRIPT**, **MODERN**, **DISPLAY**	**ROMAN**	A type of character face or specific characteristics of the font. Note • **SWISS** is a proportionally spaced, sans serif font. • **ROMAN** is a proportionally spaced, serif font. • **SCRIPT** is a fixed-pitch font designed to look like handwriting. • **MODERN** is a fixed-pitch, sans serif or serif font.
weight	**LIGHT**, **MED**, **BOLD**	**MED**	The degree of boldness of a typeface caused by different thickness of the strokes that form a graphic character.
italic	**1** = YES **0** = NO	0	A font with right-slanting characters.

Syntax rules

- A comma must separate attributes
- A question mark (?) can be used only as the wildcard character for the second character in the character set name. This allows all the character rotations of the coded fonts to be handled with one entry while searching.

  Note

  A sequential search is performed for the character set, and the first match (including the wildcard character) is used.
- The **CHARSET** section must come before the **FGID** section in the file.
- In the **CHARSET** section of the file, only the *fgid* and *height* attributes are required.
- In the **FGID** section of the file, only the *familyname* and *style* attributes are required.
- If you define a default character set in the file, it must be the last entry in the **CHARSET** section.
- If you add your own AFP font character set to the **CHARSET** section, you must assign it a font global identifier. If the new character set has the same *familyname*, *style*, *weight*, and *italic* attributes as an existing character set, you can use the same font global identifier; otherwise, you must add a unique font global identifier to the **FGID** section.

Code Page Definition File

The code page definition file (`cpdef.fnt`) maps the AFP code page name to its code page global identifier (CPGID) and, optionally, to an encoding classification and an encoding type for Asian languages.

The section header, **CODEPG**, is followed by a list of AFP code pages and their attributes. The first attribute in each line is the AFP code page global identifier that maps to a code page map file. The second attribute is the encoding classification that you decide is the best match for your AFP code page. The last line gives the default attribute values to be used when a default is required.

CODEPG Section of `cpdef.fnt` File, p. 51 shows an example of the contents of the `cpdef.fnt` file.

CODEPG Section of `cpdef.fnt` File

```
[CODEPG]
;codepage = cpgid,wincp
T1DCDCFS=1003,ANSI
T1DEBASE=2058,ANSI
T1DOBASE=2063,ANSI
T1DOGP12=2085,ANSI
T1GIO395=2079,ANSI
T1GPI363=2066,SYMBOL
T1V10037=37,ANSI
T1V10273=273,ANSI
T1000290=290,ANSI
T1000310=310,ANSI
T1000423=423,ANSI
T1000905=905,ANSI
DEFAULT=361,ANSI
```

Attribute Values for CODEPG

Attribute	Value	Shipped Default	Description
cpgid	A CPGID in the range of 65280 to 65534	361	An AFP-defined code page global identifier (CPGID), your own defined CPGID, or any other CPGID not already used in the file.
wincp	**ANSI**, **SYMBOL**, **OEM**, **NONSTD**	**ANSI**	PDF encoding

Syntax rules

- A comma must separate attributes
- Only the first attribute, *cpgid*, is required.

- If you create your own code page, you must assign it a unique code page identifier. Leading zeros are not valid. (You can use a predefined code page global identifier, but only if the character-to-hexadecimal code mapping is the same for your code page.)
- If you define a default code page in the file, it must be the last entry in the file.

Code Page File Map

AFP2PDF Transform provides one code page map file for each AFP code page supplied with Print Services Facility (PSF) and the Data1 and Sonoran licensed programs. These files are installed in the `\font\maps` subdirectory.

The file is named for its code page global identifier (CPGID) and has a file extension of `.cp` (for example, if `2063.cp` is the file name for the T1D0BASE code page map; its CPGID is 2063). Each file contains the character identifiers (and associated EBCDIC hexadecimal code points) for an AFP code page and maps them to character identifiers (and associated ASCII code points) for an ANSI or SYMBOL PDF encoding.

Code page map file example, p. 52 shows an example of the contents of the code page map file `395.cp` for the T1000395 code page mapped to the ANSI encoding.

Code page map file example

```
;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
LF570000 8F NOMATCH 00
SM190000 90 SM190000 B0
LJ010000 91 LJ010000 6A
LF510000 A0 NOMATCH 00
;;;;;;;; ; SD150000 5E
;;;;;;;; ; SD130000 60
;;;;;;;; ; LT630000 FE
/*
```

Syntax rules

- Blanks must separate character identifiers.
- **NOMATCH** means there is not a matching character in the output character set.
- The **NOMATCH** hexadecimal code of 00 is mapped to the undefined code point. When a document contains a character that does not exist in the output character set, that character cannot be displayed. If the character has not been remapped in the code page map file or the alias file, the undefined code point character is displayed as a substitute.

- The string of semicolons (`;;;;;;;;;`) means this line is ignored as a comment. It also indicates that the output code page contains a character that does not exist in the AFP code page. The code point for a character not found in the AFP code page can be used for replacing **NOMATCH** characters.
-
  - If the input code point maps to `NOMATCH 00`, and the corresponding AFP code page and character set resources are available (inline or in a resource directory), the transform extracts the raster character from the AFP font and places it as an image in the PDF output.

Alias File

The alias file (`alias.fnt`) lists the font metric file name and the font family name aliases in the **FONT** section. Font family name aliases let you change all of the requested instances of a font family name (as defined in the character set definition file) to another font family name.

Alias File Example, p. 53 shows an example of how the `alias.fnt` file is used with the AFP2PDF Transform to change all requests for the SonoranSerif font to requests for the Times font, which is one of the base fonts available in the Adobe Acrobat Viewer.

Alias File Example


```
[FONT]
; ***** Requested font = font name,Font metric/AFM filename (or 'NULL' for
SonoranSerif=Times, NULL
```

Syntax rules

- If multiple mappings are listed in the file for the same family name, only the first match is used.
- The alias file is processed sequentially. Items within the alias file are not chained. That is, if `Century Schoolbook` is set equal to `Times`, and `Times` is set equal to `Times New Roman`, then `Century Schoolbook` is not set equal to `Times New Roman`.
- Blanks in family names are treated as characters. For example, `New Century Schlbk` is not the same font as `NewCenturySchlbk`.

Font Mapping File

The `afpfont.fnt` file specifies the AFP font objects (coded fonts, code pages, and character sets) that should be explicitly processed by the AFP2PDF Plus Transform. AFP raster fonts can be converted to Adobe Type 3 fonts and embedded in the output PDF file. AFP outline fonts can be converted to Adobe Type 1 fonts and embedded in the output PDF file.

Font Mapping File Example, p. 54 shows examples of specifying AFP font objects in this file. The `?` wildcard in the second character of character set names (`C?RAST01`) and coded font names (`X?RAST01`) can be used to handle efficiently different rotations of raster fonts. The `*` wildcard used in typical file name matching can also be used to specify multiple resource names (`C?MRAS*`) with a single entry.

If the AFP font is a raster font, it is also possible to map it to a PDF environment font but have the transform process only specific characters from the AFP font object. For example, `C?H00040 84,98, C1` uses AFP raster fonts for code points X'84', x'98', and X'C1' and the information defined in the font definition files for all other characters.

AFP fonts consist of multiple parts, including the coded font, code page, and character set objects. All the necessary object names must be listed in the appropriate section. Otherwise, the AFP font is not explicitly processed.

Font Mapping File Example

```
[CODEDFNT]
;Coded Font
;X?*X?RAST01

[CODEPG]
;Code Page
T1RAST01;T?*

[CHARSET]
;Character Set
C?RAST01
C?MRAS*
C?H00040 84,98,C1
C?* 0
```


Process for Mapping Fonts

If your document uses an AFP font that is not listed in the font definition file, if you have modified the AFP fonts, or if you have created your own AFP fonts, you must edit the font definition files to add the fonts so that documents using those fonts display correctly with the AFP2PDF Plus Transform.

For example:

- If you created a new coded font (or renamed one), you need to define the coded font in the coded font file (`icoded.fnt` or `coded.fnt`).
- If you created a new character set, you must define it in the character set definition file (`csdef.fnt`).
- If you created a new code page, you must define it in the code page definition file (`cpdef.fnt`).
- If you created a new code page or modified a code page by moving characters, you need to create a new code page map file (*`cpgid`*`.cp`).

If you modified an existing font component, for example, by deleting code points in the code page, you might not need to edit some of the definition files.

After determining which font files you need to modify, follow these steps to map your fonts:

1. Gather the information needed to define the fonts in the font definition files.
2. Make backup copies of any of these font definition files that you plan to modify:

   `afpfont.fnt`

   `alias.fnt`

`coded.fnt`
`cpdef.fnt`
`csdef.fnt`

Backup copies let you restore the original file if the modified copy becomes corrupted.

3. Assign substitutes for nonmatching characters in the code page map file.

   See Code Page File Map, p. 52 for information about code page map files.

4. Edit the `cpdef.fnt` file and add your code page name, code page identifier, and the best matching encoding classification for the fonts you are using.
5. If you created a new character set, edit the `csdef.fnt` file.
   1. Add your character set name in the **CHARSET** section.
   2. Specify the correct attributes for your font.
   3. Add the appropriate information in the **FGID** section of the file if you are naming a new font global identifier.
6. If you have created a coded font, create or edit the `coded.fnt` file and add your coded font.
7. If you use AFP raster fonts, list the corresponding AFP coded fonts, code pages, and character sets in the `afpfont.fnt` file.

Using Custom AFP Raster Font Files

AFP font objects whose names are not listed in the font definition files like `csdef.fnt` and `cpdef.fnt` are considered custom.

If the custom font consists of raster characters, the AFP2PDF Plus Transform can be configured to process the AFP font data for high fidelity output. To do this, the transform first looks for the AFP coded fonts, code pages, and character sets entries in the `afpfont.fnt` file. If the font objects are listed, the transform looks for the necessary AFP font resource data, using this search order:

1. Resources that are inline with the document data
2. The resource group file specified on the afp2pdf command
3. Resource directories

Processing AFP raster font characters provides higher-fidelity output than AFP printed output; but when these small raster characters are presented on the screen, the display may not be optimal because image data is dynamically scaled.

Using Custom Font Metric Files

The AFP2PDF Plus Transform can be configured with custom font metric files to control the placement of individual characters and to aid in text alignment. The transform uses the default fonts available in the PDF display application (for example, Adobe Acrobat) and applies special character widths as specified in the font metric file.

Font metric files should be set up to use the default PDF WinANSI encoding (code page 1252). They cannot be used for double-byte text (Asian languages). The files are located in the `\font\Type1` subdirectory and are named `Custom-Metrics-`*xx*`.met`, where *xx* is 1 to 99.

To configure the transform, specify the name of the font metric file as the first parameter in the `alias.fnt` file. Set the second parameter, specifying the Adobe Font Metric (AFM) file name, to **NULL**.

Custom Font Metric files Defined in the `alias.fnt` File, p. 56 shows an example of the `alias.fnt` file with entries that define custom font metric files to the transform.

Custom Font Metric files Defined in the `alias.fnt` File

```
;**** Requested font=font name,Font metric/AFM file name (or
'NULL' for not used) *****
Font1=Custom-Metrics-1,NULL
Font2=Custom-Metrics-2,NULL
;******* End User-defined/Custom names *******
```

The format of custom font metric files follows the AFM specification, including these additional parameters that describe the font used for the display:

Flags

Specifies various characteristics of the font. Values for the Flags Parameter, p. 56 shows the allowable values.

Values for the Flags Parameter

Typeface	Flag
Courier—Serif Fixed Pitch	35
Courier Bold	262179
Courier Italic	99
Courier Bold Italic	262243
Helvetica (Arial)—Sans Serif Proportional	32
Helvetica Bold	262176
Helvetica Italic	96
Helvetica Bold Italic	262240
Sans Serif Fixed Pitch	33
Sans Serif Bold Fixed Pitch	262177
Sans Serif Italic Fixed Pitch	97
Sans Serif Bold Italic Fixed Pitch	262241
Times New Roman—Serif Proportional	34
Times New Roman Bold	262178
Times New Roman Italic	98
Times New Roman Bold Italic	262242

StemV

Specifies the width, measured in the x direction, of the dominant vertical stems of characters in the font.

In practical terms, **StemV** represents the weight or boldness of the font. A typical medium weight font has a value of 80. For a light font, the value might be in the 65–70 range. A bold font might have a value of 120. In all cases, use these values as a starting point. Adjust the **StemV** value to get the output PDF file to look like the original printed output.

ItalicAngle

Specifies the angle, in degrees counterclockwise from the vertical, of the dominant vertical strokes of the font. Typically, this value is in the range of -10 to -15. Adjust as necessary to get the output PDF file to look like the original printed output.

Example of a Custom Font Metric File, p. 57 shows an example of a custom metric file called `Custom-Metrics-1.met`.

Example of a Custom Font Metric File

```
FontName COCUSTOM
; Description - 'Custom Font 1'
Ascender 924
CapHeight 720
Descender -216
Flags 32
FontBBox -47 -204 996 924
ItalicAngle 0
StemV 90
XHeight 720
StartCharMetrics
C 32 ; WX 240 ; N space ;
C 33 ; WX 252 ; N exclam ;
C 34 ; WX 408 ; N quotedbl ;
C 35 ; WX 480 ; N numbersign ;
C 36 ; WX 480 ; N dollar ;
C 37 ; WX 756 ; N percent ;
C 38 ; WX 552 ; N ampersand ;
C 39 ; WX 240 ; N quotesingle ;
   .
   .
   .
EndCharMetrics
```


Mapping AFP Fonts to Type 1 Fonts

You can map an AFP Font file to a Type 1 font by specifying within the alias.fnt file the font name. Figure 15 shows an example of the mapping AFP fonts to type 1 fonts.

In order for the mapping to work, Afp2pdf Plus searches the "PFB-Font-Name-1.pfb" Printer Binary file within the PFBPFMDirectory font directory. Also, a font metrix has to exist in the PFBPFMDirectory: so, in Figure15 example either "PFB-Font-Name-1.pfm" or "PFB-Font-Name-1.afm" need to exist. If the “PFB-Font-Name-1.afm” files exists, the AFP2PDF Plus transform uses it, if it does not exist, it looks for the “PFB-Font-Name-1.pfm” file. You can specify the PFBPFMDirectory within the configuration file as stated in the AFP2PDF Plus Transform Options File, p. 25. If the "PFB-Font-Name-1.pfb" file is missing from the PFBPFMDirectory path, Afp2pdf Plus uses the default "Times Roman" font.

Note

You can only map single byte fonts to Type1 fonts.

Afp font to Type 1 font mapping in the alias.fnt file

```
;**** Requested font=font name,Font metric/AFM file name (or
'NULL' for not used) *****
Font1=PFB-Font-Name-1,NULL
Font2=PFB-Font-Name-2,NULL
;******* End User-defined/Custom names *******
EndCharMetrics
```

Mapping AFP Fonts to TrueType Fonts

You can map an AFP font file to a TrueType font by specifying within the alias.fnt file the TrueType font name. Figure 16 shows an example of the mapping AFP fonts to TrueType fonts.

The extension is specified to differentiate from a mapping to a Type1 font. For the mapping to work, Afp2pdf Plus searches the "TrueTypeFont-Name-1.ttf" TrueType file within the TrueType_Directory. You can specify the TrueType_Directory within the configuration file as stated in the AFP2PDF Plus Transform Options File, p. 25. If the "TrueTypeFont-Name-1.ttf" file is missing from the TrueType_Directory path, Afp2pdf Plus uses the default "Times Roman" font.

Any code page defined in the Code Page Definition File (cpdef.fnt) can be used when mapping AFP Fonts to TrueType Fonts. To use custom character encoding, the mapping must be placed in the Code Page Definition File and the *.ucm file must contain the code page id in the name structure and must be located in the Locale Directory.

Note

Afp2pdf Plus does not support TrueType Collection files (*.ttc) and OpenType fonts (*.otf) files for the mapping.

Afp font to TrueType font mapping in the alias.fnt file

```
;**** Requested font=font name,Font metric/AFM file name (or
'NULL' for not used) *****
Font1=TrueTypeFont-Name-1.ttf,NULL
Font2=TrueTypeFont-Name-2.ttf,NULL
;******* End User-defined/Custom names *******
EndCharMetrics
```

Using AFP TrueType Fonts

When TrueType fonts are specified in an AFP object container, the AFP2PDF Plus Transform automatically extracts, converts and embeds a TrueType subset into the output PDF file.

TrueType font collections, linked TrueType fonts, and the Resource Access Table (RAT) are not supported.

Using User Defined Characters within a DBCS Font

You can configure the AFP2PDF Plus to process a double-type character set (DBCS). AFP font characters from certain sections are converted, while the rest are mapped. The sections to convert are specified in the UDC_Range configuration entry. The Font resources containing User Defined Characters (Character Set and Code page) need to be placed in the path specified by the ResourceDataPath transform option.

Character encoding

The AFP2PDF Transform uses the Unicode character encoding standard to reference characters in TrueType fonts. If the AFP text is encoded in a format other than a standard Unicode encoding (UTF-8 or UTF-16), the text must be converted to Unicode during the transformation process. You may require new conversion tables that use the International Components for Unicode (ICU) function.

The `locale` subdirectory is the default location for the ICU converter files (`*.ucm`). You can use the Locale_Path parameter in the transform options file to define a directory that is different from the default. For more information, see AFP2PDF Plus Transform Options File, p. 25.

The files must have a `.ucm` extension and contain the code page ID. For example: `ibm-500.ucm`.

The code page ID is the numerical CPGID defined in the `cpdef.fnt` code page definition file. See Code Page Definition File, p. 51 for more information.

7. Working with AFP Images

- **Adding background image from a PDF file**
- **Mapping AFP images**

Adding background image from a PDF file

AFP2PDF Plus can add a background image or logo to the PDF output file by using commands in the AFP2PDF trransform options file.

The two commands that you can use are described in AFP2PDF Plus Transform Options File, p. 25:

- Append_PDF_File specifies the page origin as the bottom left-hand corner.
- Append_PDF_File2 specifies the page origin as the top left-hand corner.

You can specify various different options with this command, such as logo placement and scaling. Logo placement and scaling are controlled by specifying a set of linear transformation matrix values.

For more information on these values, see AFP2PDF Plus Transform Options File, p. 25 or the Adobe PDF Reference V1.7, Section 4.2.2, "Common Transformations", and Section 4.2.3 "Transformation Matrices".

Mapping AFP images

When an AFP file is converted, images are identified using parameters that specify the name, the position, and the size of each image. If an AFP page contains images, AFP2PDF Plus Transform creates image information entries in an output file. The output file entries can be copied into an image map configuration file to define and map a particular image.

Mapping images lets you handle AFP images in different ways during the conversion process, such as:

- Identifying individual images in your converted files.
- Removing all or some of the images from your converted output.
- Substituting all or some of the images with previously generated images in the PDF output with JPEG images.
- Substituting all or some of the images with a solid-colored rectangle. This is especially useful for improving the look of the shaded areas, defined as images in the AFP data stream.
- Adding an image, which is not part of the AFP data, to the PDF display that models the use of a preprinted form used during printing.
- Storing frequently used images to reduce the size of a PDF file.

The configuration file handles all the image conversion processing. For example, when the transform program is run against an AFP document and an image is encountered, the program searches for a matching image entry in the configuration file. If an entry that matches the name, position, size, or a combination of the three is defined, the information from the configuration file is used to convert the image. If a matching image entry in the configuration file contains an empty entry, the image is not generated in the PDF output file.

Note

- The file examples in this chapter are specified for the Windows environment. To use these examples in a UNIX environment, use the UNIX file naming convention for any file name. For example, the input AFP file `c:\documents\afpdoc.afp` in Windows is `/documents/afpdoc.afp` in a UNIX environment.

Creating the image map configuration file

To map images for the AFP files perform these steps:

1. Create an image map configuration file, transforming a sample AFP document that contains all documents with images.
2. Identify the image entries and define them in the configuration file.

The image information in the configuration file is used to identify the images in the AFP file and map them during the conversion process.

Image information in the image map configuration file

```
<IMAGE position: (5.250in, 0.613in) size: (0.667in, 0.800in)>
<IMAGE_END>
<IMAGE position: (0.863in, 8.483in) size: (2.400in, 0.667in)>
<IMAGE_END>
<IMAGE position: (3.596in, 8.550in) size: (2.633in, 0.700in)>
<IMAGE_END>
<IMAGE name: (S1PSEG01) position (6.162in, 8.483in) size: (2.067in,
0.604in)>
<IMAGE_END>
```

Each IMAGE tag along with its corresponding < IMAGE_END> tag defines a single image information entry in the configuration file. The first value for position and size is the horizontal dimension and the second value is the vertical dimension. The position measurements are for the upper, left-hand corner of the image relative to the upper, left-hand corner of the page.

By default, the image map configuration file is named `imagemap.cfg`. AFP2PDF Plus Transform searches for the file in the same directory where the program was installed. However, you can specify a different location and name for the file.

To create the image map configuration file:

1. Create or modify the AFP2PDF Plus Transform options file with the entry **ImageMapEntries_File=** ***outputfile***, where *outputfile* is the location and name of the file for the AFP file image information. For example, `ImageMapEntries_File=c:\imagemap.out`.
   See AFP2PDF Plus Transform Options File, p. 25 for more information on the AFP2PDF Plus Transform option parameters.
2. Enter **afp2pdf** `afpfile` to run the AFP2PDF Plus Transform, where `afpfile` is the directory and file name of the AFP document to convert. For example, `afp2pdf c:\documents\afpdoc.afp`.
   The system generates an output file with image information for the AFP file and a PDF file for the AFP file For example:
   `c:\imagemap.out`
   `c:\documents\afpdoc.pdf`
3. Copy the image lines in the output file (such as `imagemap.out`) into the image map configuration file (`imagemap.cfg` by default).
4. Add image information between the starting <IMAGE> and ending <IMAGE_END> lines for the images you want to generate in the PDF output file.

The image information in the configuration file is used to identify the images in the AFP file. When AFP2PDF Plus Transform matches an image from the AFP file with the image information from the configuration file, it generates the image in the PDF output file.

Identifying AFP images in the image map configuration file

You can identify the images from the configuration file from the name, position, and size information, or you can work with the **afp2pdf** transform command to visually identify individual images. By creating empty image entries and then commenting out a single entry in the file, you can identify the image when you rerun the transform and the image is generated in the output PDF file.

To identify individual images:

1. Define empty image information entries for all images in the image map configuration file. Empty entries do not include any image information between the starting <IMAGE> and ending <IMAGE_END> lines. For example:

```
<IMAGE position:(0.863in,8.483in) size:(2.400in,0.667in)>
<IMAGE_END>
```

2. Comment out the first image in the configuration file by adding slashes before the entry. For example:

```
//<IMAGE position:(5.250in,0.613in) size:(0.667in,0.800in)>
//<IMAGE_END>
<IMAGE position:(0.863in,8.483in) size:(2.400in,0.667in)>
<IMAGE_END>
<IMAGE position:(3.596in,8.550in) size:(2.633in,0.700in)>
<IMAGE_END>
<IMAGE name:(S1PSEG01) position:(6.162in,8.483in) size:(2.067in,0.604in)>
<IMAGE_END>
```

3. Run **afp2pdf** to generate a PDF file that contains only the image that you commented out. For example: `afp2pdf c:\documents\afpdoc.afp`, where `c:\documents\afpdoc.afp` is the directory and file name of the converted AFP document .

Removing images using the image map configuration file

The image information in the configuration file is used to identify the images in the AFP document and map them to the PDF file during the conversion process. If a matching image entry in the configuration file contains an empty entry, the image is not generated in the PDF output file. Empty entries do not include any image information between the starting <IMAGE> and ending <IMAGE_END> lines. For example:

Empty entries in the image map configuration file

```
<IMAGE position: (0.863in, 8.483in)  size: (2.400in, 0.667in)>
<!-- IMAGE_END -->
```

To remove AFP file images, so they are not generated in the output PDF, create empty image entries in the configuration file.

To remove images:

1. Define empty image information entries in the image map configuration file for the images you do not want to generate in the PDF file.
2. Run **afp2pdf** on the AFP file (for example, `afp2pdf c:\documents\afpdoc.afp`). A PDF file is generated (such as `c:\documents\afpdoc.pdf`) that does not contain any images with empty entries in the configuration file.

Substituting existing images with AFP2PDF Transform

During the AFP2PDF PlusTransform process, you can use the image map configuration file to substitute an AFP image with a previously generated image. The only type of image to substitute is the JPEG format.

To use an existing image, add IMAGE definition parameters between the starting <IMAGE> and ending <IMAGE_END> lines of an image information entry in the image map configuration file.

The image definition parameters are:

XPos=*n*

Defines the position of the left edge of the image relative to the left edge of the page. The units of this parameter are 1440 units per inch.

YPos=*n*

Defines the position of the top edge of the image relative to the top edge of the page. The units of this parameter are 1440 units per inch.

XSize=*n*

Defines the target area width to match the size of the image. The units of this parameter are 1440 units per inch.

YSize=*n*

Defines the target area height to match the size of the image. The units of this parameter are 1440 units per inch.

Filename=*path*

Specifies the fully qualified location and file name for the image.

Note

- When you use a blank within the value, the text is enclosed in double quotes.

ColorFlag=0 | 1

Specifies the type of image to substitute. This is an optional parameter; if specified, the value is set to **1**.

Example of existing images in the image map configuration file

```
<IMAGE position:(5.250in, 0.613in) size:(0.667in, 0.800in)>
IMAGE XPos=0 YPos=0 XSize=900 YSize=200 Filename="c:\images\logo1.jpg"
<IMAGE_END>
```

```
<IMAGE position:(0.863in, 8.483in) size:(2.400in, 0.667in)>
IMAGE XPos=0 YPos=0 XSize=500 YSize=300 Filename="c:\images\logo2.jpg"
<IMAGE_END>
<IMAGE position:(3.596in, 8.550in) size:(2.633in, 0.700in)>
<IMAGE_END>
<IMAGE name:(S1PSEG01) position:(6.162in, 8.483in) size:(2.067in, 0.604in)>
<IMAGE_END>
```

The first image is replaced by `logo1.jpg` and the second image by `logo2.jpg`.

You can define abbreviated versions of the image entries to expand the matching capabilities of incoming AFP images. This can simplify and reduce the number of image entries defined in the configuration file. To define abbreviated versions, edit the image entry and specify any combination of name, position, or size. If the incoming AFP image matches all the characteristics listed for the image entry, the image is substituted.

Example of abbreviated image entries in the image map configuration file

```
<!-- IMAGE name: (S1PSERG01)>
IMAGE XPos=0 YPos=0 XSize=500 YSize=300 Filename="c:\images\logo2.
jpg"
<!-- IMAGE_END -->
```

When an incoming AFP image matches the name, the substituted image is added to the output.

Substituting AFP shaded images with colored areas

Many AFP documents contain page areas shaded with a gray box. The AFP data stream defines an image with pels laid out in a regular checker pattern to create a gray shading effect. When trying to display this type of image, however, it often becomes distorted because of the scale factor and resolution of the display hardware.

Note

- To avoid this problem, you can use a colored area instead of the shaded image.

To substitute a shaded image with a color area, add SHADED_AREA definition parameters between the starting <IMAGE>and ending <IMAGE_END> lines of an image information entry in the image map configuration file.

The shaded area definition parameters are:

XPos=*n*

Specifies the position, in inches, of the left edge of the colored area relative to the left edge of the page.

This parameter is optional because the horizontal position of the incoming AFP image is used if XPos is not specified.

YPos=*n*

Specifies the position, in inches, of the top edge of the colored area relative to the top edge of the page.

This parameter is optional because the vertical position of the incoming AFP image is used if YPos is not specified.

XSize=*n*

Specifies the colored area width, in inches.

This parameter is optional because the transform uses the width of the incoming AFP image if XSize is not specified.

YSize=*n*

Specifies the colored area height, in inches.

This parameter is optional because the transform uses the height of the incoming AFP image if YSize is not specified.

Shade_R=*n*

Shade_G=*n*

Shade_B=*n*

Specifies the intensity of the red, green, and blue colors used when generating the color of the area. The values given must be in the range 0.0 to 1.0.

When all three values are set to 1.0, white is generated. When all three values are set to 0.0, black is generated.

If these parameters are not specified, the transform uses the color described by the Shade_RGB parameter in the AFP2PDF Transform options file (see AFP2PDF Plus Transform Options File, p. 25). By default, Shade_R=0.8, Shade_G=0.8, and Shade_B=0.8 creates a light gray color that is used if no other color specification is found.

Radius=*n*

Indicates the radius of all corners of the shaded area. If specified, the value applies to all four corners that become equally rounded.

The radius value, measured in inches, should be smaller than half the width or half the height of the shaded area.

You can specify individual values for each corner of the shaded area:

Radius_UL=*n*

Specifies the radius of the upper left corner. If specified, this value overrides the initial **Radius=*n*** value, for this corner only.

Radius_UR=*n*

Specifies the radius of the upper right corner. If specified, this value overrides the initial **Radius=*n*** value, for this corner only.

Radius_LL=*n*

Specifies the radius of the lower left corner. If specified, this value overrides the initial **Radius=*n*** value, for this corner only.

Radius_LR=*n*

Specifies the radius of the lower right corner. If specified, this value overrides the initial **Radius=*n*** value, for this corner only.

Layer=*n*

When the parameter is not specified, the transform uses the **Layer** parameter from the options file.

For more information on the **Layer** option parameter, see AFP2PDF Plus Transform Options File, p. 25.

Note

- If no Layer specification is found, the default value is **Layer=*2***.

Example of colored areas in the image map configuration file

```
<IMAGE position:(5.250in,0.613in) size:(0.667in,0.800in)>
SHADED_AREA XPos=5.250 YPos=0.613 XSize=0.667 YSize=0.800
Shade_R=1.0 Shade_G=0.0 Shade_B=0.0
<IMAGE_END>
<IMAGE position:(0.863in,8.483in) size:(2.400in,0.667in)>
<IMAGE_END>
<IMAGE position:(3.596in,8.550in) size:(2.633in,0.700in>
<IMAGE_END>
<IMAGE position:(6.162in,8.483in) size:(2.067in,0.604in)>
<IMAGE_END>
```

When the first image is encountered, it is substituted with a red, 0.667 x 0.8 inch area positioned at 5.25 x 0.613 inches.

You can define abbreviated versions of the image entries to expand the matching capabilities of incoming AFP images. This can simplify and reduce the number of image entries defined in the configuration file.

To define abbreviated versions:

1. Edit the image entry
2. Specify any combination of name, position, and size.
   If the incoming AFP image matches all the characteristics listed for the image entry, the shaded area is substituted.

Example of abbreviated colored areas in the image map configuration file

```
<IMAGE size:(0.667in,0.800in)>
SHADED_AREA
<IMAGE_END>
```

When an incoming AFP image matches the image size information, a colored area with the default color is created.

Adding an image to the transform output

The preprinted forms used during the printing process might have a company logo, a table, or grid that is filled in with the print data. To add an image which emulates the preprinted form to the transform output, AFP2PDF Transform performs these steps:

1. Opens the specified image file, which currently can only have JPEG format.

2. Processed the image data.
3. Converts the data into an image for the PDF output.
   The image can be in color and you can specify which pages it is included on.

To include an image that emulates a preprinted form, add one or more of these static image definitions between the starting <IMAGE> and ending <IMAGE_END> lines of the image information entry in the image map configuration file:

STATICIMG_PAGE

Same image is included on the pages with Type parameter.

STATICIMG_FRONT

Image on the front sheet of AFP data is placed on pages with Type parameter, in the output PDF.

STATICIMG_BACK

Image on the back sheet of AFP data is placed on pages with Type parameter, in the output PDF.

STATIC_IMG

Same image is included on the pages with the Type parameter. Using 72 units per inch, you can set specific image position and size dimensions.

These parameters are used with the static image definitions:

XPos=*n*

Specifies the left edge position of the image relative to the left edge of the page. The units of this parameter are 1440 units per inch.

YPos=*n*

Specifies the top edge position of the image relative to the top edge of the page. The units of this parameter are 1440 units per inch.

XSize=*n*

Specifies the target area width where the image is placed. The units of this parameter are 1440 units per inch.

YSize=*n*

Specifies the target area height where the image is placed. The units of this parameter are 1440 units per inch.

Filename=*path*

Specifies the fully qualified path and name for the image.

Note

- If a blank is used as part of the value, you must enclose the value in double quotes.

Type=All | First | Second | All-First

Specifies the sheets where the image is placed. The values include:

All

The image is included on all sheets.

First (default value)

The image is included on the first sheet only.

Second

The image is included on the second sheet only.

All-First

The image is included on all sheets except for the first page.

Usage note:

- If the Type parameter is not specified, the default value is **First**.

If the page orientation switches between portrait and landscape, these parameters can also be specified to control the position and sizing for the landscape orientation:

XPos_Wide=*n*

Specifies the position of the left edge of the image relative to the left edge of the page. The units of this parameter are 1440 units per inch.

YPos_Wide=*n*

Specifies the position of the top edge of the image relative to the top edge of the page. The units of this parameter are 1440 units per inch.

XSize_Wide=*n*

Specifies the width of the target area where the image is placed. The units of this parameter are 1440 units per inch.

YSize_Wide=*n*

Specifies the height of the target area where the image is placed. The units of this parameter are 1440 units per inch.

Example of an image added to PDF output

```
<IMAGE>STATICIMG_PAGE XPOS=0 YPOS=0 XSIZE=12240 YSIZE=15840 FILENAME
="C:\afp2pdf\
form1.jpg" TYPE=ALL
<IMAGE>
```

Note

- The image definitions do not include position or size information. You must manually add the starting and ending lines to the image map configuration file, in addition to the static image definitions.
- You can define multiple images on a page. You must verify that the size and position do not cause the images to overlap.

Adding a shaded area to the transform output

To include a shaded area, add one or more of the static shaded area definition parameters between the starting <IMAGE> and ending <IMAGE_END> lines of an image information entry in the image map configuration file:

STATIC_SHADED_AREA

Same shaded area is included on the pages with Type parameter.

STATIC_SHADED_AREA_FRONT

Shaded area on the front sheet of AFP data is placed on pages with Type parameter.

STATIC_SHADED_AREA_BACK

Shaded area on the back sheet of AFP data is placed on pages with Type parameter.

Note

- You can also use the list of SHADED_AREA definition parameters. For more information, see Substituting AFP shaded images with colored areas, p. 65.

Adding an image or shaded area to the transform output based on a TLE key-value pair

Adding images or shaded areas to the output PDF file based on a TLE key-value pair is similar to the method of adding images or shaded areas using the STATICIMG or STATIC_SHADED_AREA . The difference is the ability to specify a key-value pair used as a match mechanism. If the specified key-value pair is found in a TLE structure field on page level or on the group page level, the pair is added to the static image.

To include an image that emulates a preprinted form using the key-value pair condition, add one or more of these static image definitions between the starting <IMAGE>and ending <IMAGE_END> lines of the image information entry in the image map configuration file:

INDEX_STATICIMG

Same image is included on the pages with Type parameter, if the Index condition is met.

You can use the same parameters as **STATICIMG_PAGE**. For more information, see Adding an image to the transform output, p. 67.

INDEX_FRONT_STATICIMG

Image on the front sheet of AFP data is placed on pages with Type parameter, in the output PDF, if the Index condition is met.

You can use same parameters as **STATICIMG_FRONT**. For more information, see Adding an image to the transform output, p. 67.

INDEX_BACK_STATICIMG

Image on the back sheet of AFP data is placed on pages with Type parameter, in the output PDF, if the Index condition is met.

You can use the same parameters as **STATICIMG_BACK**. For more information, see Adding an image to the transform output, p. 67.

INDEX_STATICSHDAREA

Same shaded area is included on the pages with Type parameter, if the Index condition is met.

You can use same parameters as **STATIC_SHADED_AREA**. For more information, see Adding an image to the transform output, p. 67.

INDEX_FRONT_STATICSHDAREA

Shaded area on the front sheet of AFP data is placed on pages with Type parameter, in the output PDF, if the Index condition is met.

You can use same parameters as **STATIC_SHADED_AREA_FRONT**. For more information, see Adding an image to the transform output, p. 67.

INDEX_BACK_STATICSHDAREA

Shaded area on the back sheet of AFP data is placed on pages with Type parameter, in the output PDF, if the Index condition is met.

You can use same parameters as **STATIC_SHADED_AREA_BACK**. For more information, see Adding an image to the transform output, p. 67.

To use the index type definitions, add the following parameter:

INDEX="key : value"

Specifies the key-value pair to be matched. If the page contains a TLE with the same key-value pair, the image or shaded area is added as specified by the Type parameter.

Note

- You can have several INDEX parameters. The condition is that all the key-values pairs from all the INDEX parameters are encountered in the TLE of the page.

If there is no match for the key-value pair index, a default value can be specified using the following entries:

INDEX_DEFAULT_STATICIMG

Same image is included on the pages with Type parameter, if no key-value pair match is found in this <IMAGE><IMAGE_END> group.

You can use the same parameters as **STATICIMG_PAGE**.

INDEX_DEFAULT_FRONT_STATICIMG

Image on the front sheet of AFP data is placed on pages with Type parameter, in the output PDF if no key-value pair match is found in this <IMAGE><IMAGE_END> group.

You can use same parameters as **STATICIMG_FRONT**.

INDEX_DEFAULT_BACK_STATICIMG

Image on the back sheet of AFP data is placed on pages with Type parameter, in the output PDF if no key-value pair match is found in this <IMAGE><IMAGE_END> group.

You can use the same parameters as **STATICIMG_BACK**.

8. JAVA Application Programming Interfaces

The AFP2PDF Plus Transform application programming interfaces (APIs) convert AFP documents and resources into files that can be viewed with Adobe Acrobat. These APIs are written to interface with a JAVA application.

The AFP2PDF Plus Transform APIs use data buffers for input and output to the transform. Many times the AFP data is retrieved from a database in a byte array in memory. The reference to this byte array is then passed directly to the conversion code for processing. The output from the API program also uses a reference to a byte array that contains the transformed output. Therefore, the overhead of opening, reading, and closing files is eliminated. Because system performance degrades if very large byte arrays are allocated, this approach assumes that the input and output byte arrays are not extremely large. The command–line interface, which uses files for input and output to the transform, might need to be used for very large byte arrays.

This chapter contains information about the programming functions available for the AFP2PDF Plus Transform API. They mimic the previous AFP2PDF JAVA APIs and are designed to seamlessly integrate into existing AFP2PDF installations with minimal migration effort. For more detailed information, see the `Javadoc` information in the `api` subdirectory.

A2PBufferMessages (boolean *flag*)

Sets up the logging to buffer messages, which are returned via A2PGetMessageBuffer() method. The default message level is **WARNING**.

A2PBufferMessages (boolean *flag*, java.lang.String *level*)

Sets up the logging to buffer messages, which are returned via A2PGetMessageBuffer() method.

Message levels: **INFO**, **WARNING**, **SEVERE**, **ALL**.

A2PDocEnd ()

Terminates the processing for the last AFP document transform.

A2PDocStart2 (byte[ ] *ResIn*, byte[ ] *DataIn*, java.lang.String *InstallDir*)

Initializes the transform processing for an AFP document and resource group file contained in the input byte array. The resulting output PDF must be obtained through the method **getPDFOutBuf()**.

A2PDocStart2 (byte[ ] *DataIn*, java.lang.String *InstallDir*)

Initializes the transform processing for an AFP document contained in the input byte array. The resulting output PDF must be obtained through the method **getPDFOutBuf ()**.

A2PDocStart2 (java.lang.String *inAFPFileName*, java.lang.String *outPDFFileName*, java.lang.String *resGrpFileName*)

Initializes the transform processing for an AFP document, resource group and output file using the input file names. The resulting output PDF is written to the output PDF file name, whether supplied or derived. A call to the method **getPDFOutBuf ()** I returns null, as the method is only useful when other **A2PDocStart2 ()** methods are used.

Note

You must call **setFormDef ()** and **setOptionsFile ()** before calling **A2PDocStart2 ()**.

A2PGenerateMessages (boolean *fGen*)

This command can be specified, but it is not supported in the current implementation. It exists for compatibility purposes only.

A2PGetMessageBuffer ()

Returns the string of log messages since the last XFormDoc.

A2PGetPageCount ()

Returns the number of pages in a specific AFP document. A value greater than 0 indicates a successful completion. A zero value or negative number indicates that an error has occurred.

A2PXFormDoc ()

Transforms the entire AFP document using the given values already set.

A2PXFormPage ()

Transforms one or more pages in an AFP document using the given values already set. The **setPageNumber** method must be used before this method to choose which page to convert. If the last page number is also set, the range of pages is transformed.

A2PXFormPage (int *n*)

Transforms a specific page in an AFP document using the given values already set.

A2PXFormPage (int *StartPageNumber*, int *EndPageNumber*, boolean *LastPage*)

Transforms a range of pages in an AFP document using the given values already set. Both *StartPageNumber* and *EndPageNumber* must be less than or equal to the number of pages in the document. *StartPageNumber* must be less than or equal to *EndPageNumber*.

GetCachedFontMapInfo ()

Returns a string of all font directories currently cached.

getCodeVersion ()

This command can be specified, but it is not supported in the current implementation. It exists for compatibility purposes only.

getCreateTimeStamp ()

This command can be specified, but it is not supported in the current implementation. It exists for compatibility purposes only.

getFontPath ()

Gets the value of the *Font Path* variable.

getFormDef ()

Gets the value of the *FormDef* variable and returns the fully qualified path name of the form definition file. This value can be null.

getImageMapFile ()

Retrieves the location of the name of the image map configuration file.

getLastPageNumber ()

Returns the last page number (1-based) to transform.

getLinearize ()

Returns **True** when the linearized option is set.

For more information, see **setLinearize (boolean *n*)**.

getOptionsFile ()

Gets the value of the *OptionsFile* variable and returns the fully qualified file path of the `a2pxopts.cfg` file. This value can be null.

getOwnerPassWord ()

Gets the value of the *OwnerPW* variable and returns the owner password. This value can be null.

getPageNumber ()

Returns the first page number (1-based) to transform.

getPDFOutBuf ()

Gets the PDF output byte array. This value can be null if an error occurred or files were used for IO.

getPermissions ()

Gets the value of the *Permissions* variable.

getRotation ()

Gets the value of the *Rotation* variable and returns the rotation setting (**0** | **90** | **180** | **270**).

getSignTimeStamp ()

This command can be specified, but it is not supported in the current implementation. It exists for compatibility purposes only.

getUserPassWord ()

Gets the value of the *UserPW* variable and returns the user password. This value can be null.

getXFormDocParameters ()

Returns a list of parameters and their current values.

InitFontMaps (String *fontDir*)

Initializes the font mapping information for the input font directory. This command can be called multiple times and is used to eliminate the initial costs of reading font mapping data when the first file is transformed. The *fontDir* variable is the fully qualified path name of the font map directory.

setCachingSize (int *size*)

Sets the maximum size of the resource cache used across all transforms within a Java Virtual Machine.

setCreateTimeStamp (java.lang.String *n*)

This command can be specified, but it is not supported in the current implementation. It exists for compatibility purposes only.

setFontPath ()

Sets the value of the *Font Path* variable.

setFormDef (java.lang.String *n*)

Sets the fully qualified form definition file name.

Note

This value must be set before calling **A2PDocStart2 ()**.

setImageMapFile (java.lang.String *n*)

Specifies the location and the name of the image map configuration file, different from the default file, `imagemap.cfg`. The file name does not use relative paths and is fully qualified. See Creating the image map configuration file, p. 62 for more information about the image map configuration file.

setLastPageNumber (int *n*)

Sets the last page number (1-based) to transform. If the first page number is not already set using **setPageNumber()**, it is set here to the first page. For example, you can transform the first 33 pages by just setting the last page number to 33.

setLinearize (boolean *n*)

Specifies that the output PDF file is created linearized (or optimized for Fast web viewing).

setOptionsFile (java.lang.String *n*)

Sets the *OptionsFile* variable.

Note

This value must be set before calling **A2PDocStart2 ()**.

setOwnerPassWord (java.lang.String *n*)

Sets the *OwnerPW* variable.

setPageNumber (int *n*)

Sets the first page number (1-based) to transform.

setPermissions (java.lang.String *n*)

Sets the *Permissions* variable.

setRotation (int *n*)

Sets the *Rotation* variable (**0** | **90** | **180** | **270**).

setSignTimeStamp (java.lang.String *n*)

This command can be specified, but it is not supported in the current implementation. It exists for compatibility purposes only.

setUserPassWord (java.lang.String *n*)

Sets the *UserPW* variable.

writeDataBuf (java.lang.String *fName*)

Writes the current AFP data bytes to the output file name. This value cannot be null.

9. Comparison with AFP2PDF Transform

The AFP2PDF Plus Transform has many of the same options as the AFP2PDF Transform. Some options have been added or deleted and some have had the default values changed.

AFP2PDF and AFP2PDF Plus Options and Defaults, p. 77 compares the options that you can specify in the `a2pxopts.cfg` for both products. Abbreviations in this table are:

- Y = Yes
- N = No
- D = Deprecated
- R= Reserved
- T= True
- F= False

AFP2PDF and AFP2PDF Plus Options and Defaults

Transform Options	AFP2PDF (5876–W01)	AFP2PDF Defaults	AFP2PDF Plus (5876-W11)	AFP2PDF Plus Defaults
Append_Log_to_PDF	Y	F	N	
Append_PDF_File	Y		Y	
Append_PDF_File2	N		Y	
Author	Y		Y	
Auto_Rotate	Y	F	Y	F
Auto_Rotate_CountBlanks	N		Y	T
Auto_Rotate_ParseOverlays	N		Y	F
Auto_Rotate_UseCharacterRotation	N		Y	F
AVE_Control_File	N		Y	
Bilevel_Image_Cnvt	Y	F	N	
Bypass_White_GOCA_Lines	Y	F	N	
Cache_AFP_Overlay	Y	F	Y	T
Cache_Font_Image	Y	T	N	
Certificate	Y		Y	
Certificate2	Y		Y	
Color	Y		Y	
Compression_Level	Y		N	
Convert_Text_CMYK	N		Y	F
Creator	Y		Y	
Dash_Pattern	Y		N	
Dash_Pattern_Endcap	Y		N	

Transform Options	AFP2PDF (5876–W01)	AFP2PDF Defaults	AFP2PDF Plus (5876-W11)	AFP2PDF Plus Defaults
Default_Encryption_Permissions	Y		Y	
Default_Linearization	F		F	
Default_Owner_Password	Y		Y	
Default_User_Password	Y		Y	
Disable_Bookmark_Generation	Y	F	Y	F
Disable_Compression	Y	F	Y	F
DotDensity	Y		Y	
Enable_Auto_Font_Image	Y	F	Y	F
Enable_Auto_Image_Cache	Y	F	Y	T
Enable_One_Pass	Y	F	N	
Enable_UDC	Y	F	N	
Expand_Index_Values	Y	F	N	
FIPS_Mode	Y	F	Y	F
Flate_Compression_Level	N		Y	6
FontExt	Y		Y	
Font_Image_Pad_Height	Y		N	
Font_Image_Pad_Width	Y		N	
Font_Path	Y		Y	
Force_GOCA_Area_Boundary	Y	F	N	
Generate_Type3	Y	F	Y	T
Global_Scale	Y	100	N	
GOCA_Pass1	Y		N	
GOCA_Pass2	Y		N	
GOCA_Pattern	Y		Y	
GOCA_Use_Circles	Y	F	Y	F
Honor_Constant_Forms	Y	F	Y	F
Honor_Medium_Orientation	N		Y	F
Honor_Media_Eject_Control	Y	F	N	
Honor_Medium_Colored_Rules	Y	F	N	
Horizontal_Offset	Y		Y	

Transform Options	AFP2PDF (5876–W01)	AFP2PDF Defaults	AFP2PDF Plus (5876-W11)	AFP2PDF Plus Defaults
Ignore_Bar_Code_Object_Area_Size	N		Y	F
Ignore_Data_Font_Height	Y	F	Y	F
Ignore_PTOCA_STC	Y	F	Y	F
Ignore_Transform_Warnings	Y	F	N	
ImageMapEntries_File	Y		Y	
JPEG_Compress_CMYK_Image	Y	F	N	
JPEG_Quality_Level	N		Y	0.95
Keywords	Y		Y	
Layer	N		Y	2
Launch_Preview	Y	F	N	
Locale_Path	Y		Y	
Logging	Y		Y	
Max_Annotes	D		N	
Max_Fonts	D		N	
Max_Images	D		N	
Max_Leaves	D		N	
Max_Objects	D		N	
Max_Overlays	D		N	
Max_Pages	D		N	
Modify_Text_Colors	Y	F	Y	F
Old_Static_Paper	Y	F	N	
Output_IndexInfo	Y	F	Y	F
Output_IndexInfo2	N		Y	F
OverlayExt	Y		Y	
Page_Data_Order_Flags	Y	0	N	
Page_Rotation	Y	0	Y	0
PageSegExt	Y		Y	
PDF/A	Y	F	Y	F
Pdf_extGState_Blend	Y		N	
PfmPfb_Directory	Y		Y	

Transform Options	AFP2PDF (5876–W01)	AFP2PDF Defaults	AFP2PDF Plus (5876-W11)	AFP2PDF Plus Defaults
Preserve_CMYK	Y	F	Y	F
Printer_Resolution	Y		N	
PTX_Drawrule_Use_Line	N		Y	F
ResourceDataPath	Y		Y	
Set_BC_GS1_128_Widths	Y		N	
Shade_RGB	Y		Y	
Show_Outline	Y	T	Y	T
Show_Pageids	Y	T	Y	T
Static_Paper_Center	Y	F	N	
Static_Paper_Length	Y		Y	
Static_Paper_Width	Y		Y	
Subject	Y		Y	
Substitute_Default_Medium_Map_Allowed	N		Y	F
Title	Y		Y	
Trace_Level	Y		N	
Transform_All_Subgroups	Y	F	Y	F
TrueType_Directory	Y		Y	
UDC_File	R		Y	
UDC_Range	Y		N	
Use_AFP_Metrics	Y	F	N	
Use_ICU	Y	F	N	
Use_Intermediate_File	Y	F	N	
Use_Unicode	Y	F	N	
Vertical_Offset	Y		Y	

INDEX

T

U

V

W

X

Data Transforms AFP2PDF Plus Transform User's Guide

EN US EN GB EN AU 3/2020 G5502036410A