



AFP2web Documentation

AFP2web Version 4.3.x.x User Guide and Reference

Date: 2015-10-06

Maas Holding GmbH

Copyright © 2015 Maas Holding GmbH, Germany

All rights reserved. This publication and the concepts, solutions, programs described are copyright protected products of Maas Holding GmbH, Filderstadt, Federal Republic of Germany. This publication is subject to change or correction without prior notice.

Maas Holding GmbH will not be held liable for any damages caused or alleged to be caused either directly or indirectly by this publication.

No part of this documentation may be reproduced, stored in a retrieval system or transmitted in any form or by any means, without the prior written permission of Maas Holding GmbH.

Table of Contents

1	Preface to the AFP2web User Guide 4.x.....	10
1.1	About the AFP2web and Scripting Facility User's Guide	10
1.1.1	Who Should Read this Document.....	10
1.1.2	What's in this Document.....	10
1.1.3	Where to Find Additional Information	11
1.1.4	Conventions	11
1.2	Change History of AFP2web 4.x.....	11
1.2.1	New Since AFP2web Version 3.x.....	11
1.2.2	New Since AFP2web Version 2.x.....	13
1.3	Deprecated Since AFP2web 4.x.....	16
1.3.1	Font Metric Files.....	16
1.4	Important Notices for AFP2web 4.x.....	16
1.4.1	Test thoroughly!	16
1.4.2	Only you know your fonts!.....	16
1.4.3	Make backups of your documents!	16
1.4.4	Backup your previous installation of AFP2web!	17
1.4.5	Use the AFP2web Scripting Facility with care!	17
1.5	Migrating from AFP2web Version 2.x to Version 3.x/4.x.....	17
1.5.1	Changes in the mapping.def.....	17
1.5.2	Changes in the CP Files	18
1.5.3	Defining and Supplying Font Resources	18
1.5.4	Scripting Facility Modifications.....	18
2	Introduction to AFP2web 4.x	19
2.1	Overview of the AFP2web 4.x Functionality	19
2.1.1	Core Components of AFP2web 4.x	19
2.1.2	Methods of Integrating AFP2web 4.x.....	20
2.1.3	Supported Formats in AFP2web 4.x	21
2.2	Application Scenarios for AFP2web 4.x.....	21
2.2.1	Archival of Documents and Index Update	22
2.2.2	On-the-fly Conversion for PDF On-Demand	23
2.2.3	AFP Viewer on Workstations	23

2.2.4	Document Distribution per Post (Print), E-Mail, or Fax	24
3	AFP2web 4.x User Guide	25
3.1	Installing AFP2web 4.x	25
3.1.1	System Prerequisites for AFP2web 4.x	25
3.1.2	Installing AFP2web 4.x on Windows	27
3.1.3	Installing AFP2web 4.x on Unix	27
3.1.4	AFP2web 4.x Installation Results	28
3.1.5	Removing AFP2web 4.x	30
3.2	Configuring AFP2web 4.x.....	30
3.2.1	Overview	30
3.2.2	Setting Parameters in the INI File	30
3.2.3	Converting Sample Documents and Fine Tuning the Configuration.....	31
3.2.4	Providing AFP Resources.....	31
3.2.5	Specifying the Search Path for Shared Objects.....	33
3.3	Using AFP2web 4.x	34
3.3.1	Quick Start	34
3.3.2	Starting/Stopping AFP2web From the Command Line	34
3.3.3	Controlling the Conversion Process.....	35
3.3.4	Selecting Documents for Conversion	36
3.3.5	Selecting Pages for Conversion.....	37
3.3.6	Processing AFP Indexes.....	37
3.3.7	Storing Output Documents in Sub Folders	38
3.3.8	Processing Included Objects (Included Images)	39
3.3.9	Notes on Barcode Support.....	40
3.3.10	Creating PDF Bookmarks for Index Elements.....	41
3.3.11	Adding Text as Watermark Behind the Document Text	42
3.3.12	Stopping Conversion When a Required Resource Is Missing	42
3.3.13	Using the Log File	42
4	Handling Fonts.....	45
4.1	The Configuration Files and the Defaults Used for Font Processing	45
4.1.1	The INI File (afp2web.ini)	45
4.1.2	The Font Mapping File (mapping.def)	46
4.1.3	Code Page Files.....	47
4.1.4	Default Assumptions If Mappings or Font Resources are Missing.....	47

4.2	AFP2web Font Processing Modes	47
4.2.1	Use of the Original Fonts (Default)	47
4.2.2	Font Substitution and Referencing	50
4.2.3	Font Substitution and Embedding	54
5	AFP2web 4.x Scripting Facility User Guide.....	57
5.1	Basic Concepts of the AFP2web Scripting Facility	57
5.1.1	Scripting Facility Applications.....	57
5.1.2	Scripting Facility Interaction With AFP2web	60
5.1.3	Scripting Facility Processing Features.....	64
5.2	Using AFP2web Scripting Facility.....	66
5.2.1	Activating the Scripting Facility	66
5.2.2	Setting the Perl Environment	67
5.2.3	Using the INI Parameter MemoryOutputStream	68
6	AFP2web Reference	70
6.1	Overview of INI Parameters and Command Line Options	70
6.2	INI Parameter Reference	79
6.2.1	Function and Structure	79
6.2.2	Usage Notes.....	80
6.2.3	Section [Settings]	80
6.2.4	<formdef>= <medium map-list>.....	116
6.2.5	Section [AFPColorTable].....	116
6.2.6	Section [SPOOLFILETYPE]	117
6.2.7	User-defined Sections for Logical Locations	118
6.3	AFP2web Command Line Options.....	119
6.3.1	Usage Notes.....	119
6.4	Parameters of the mapping.def	123
6.4.1	Function and Structure	123
6.4.2	Usage Notes.....	124
6.4.3	[CHARSET RENDERING] Section.....	124
6.4.4	[XFONT] Section	126
6.4.5	[CHARSET] Section	127
6.4.6	[FGID] Section	128
6.4.7	[FONTSUFFIX] Section	129
6.4.8	[INPUT FONT ALIASES] Section	130

6.4.9	[PDFFONT] Section	131
6.4.10	[WINFONT] Section.....	132
6.4.11	[UNIXFONT] Section.....	133
6.4.12	[AFPFONT] Section	134
6.4.13	[CODEPG] Section	135
6.4.14	[GCSGID] Section.....	136
6.5	Code Page Files (CP Files)	137
6.6	Output Logging.....	138
6.6.1	Page Logging.....	138
6.6.2	AFP Resource/Data Object Logging.....	138
6.6.3	ColorSpace Logging.....	145
6.6.4	Font Logging	146
6.6.5	Image Logging	151
6.6.6	Line Logging	153
6.6.7	AFP Modca-IS2 Compliance Logging.....	154
6.7	Identifying Fonts Using the IBM Naming Convention.....	154
6.7.1	Overview	154
6.7.2	The AFP Font Naming Convention	154
6.8	Scripting Facility Quick Reference	158
6.8.1	Scripting Facility Interface and Processing Levels	159
6.8.2	Script Routine and Methods.....	159
6.8.3	Methods and Where Used in a Script	166
6.9	Scripting Facility API Reference	178
6.9.1	Overview	178
6.9.2	afp2web.pm	179
6.9.3	a2w::Bookmark.....	184
6.9.4	a2w::Comment [NEW]	193
6.9.5	a2w::Config	195
6.9.6	a2w::ConfigConstants [NEW]	196
6.9.7	a2w::Document [EXTENDED]	201
6.9.8	a2w::DocumentConstants [NEW].....	216
6.9.9	a2w::Font	217
6.9.10	a2w::FontConstants [NEW]	229
6.9.11	a2w::Image [NEW]	230
6.9.12	a2w::ImageConstants [NEW].....	237

6.9.13	a2w::Index	239
6.9.14	a2w::Kernel	244
6.9.15	a2w::Line	247
6.9.16	a2w::MediumMap	253
6.9.17	a2w::NOP	256
6.9.18	a2w::Overlay	258
6.9.19	a2w::Page	264
6.9.20	a2w::PageConstants [NEW]	289
6.9.21	a2w::PSEG	290
6.9.22	a2w::Text [EXTENDED]	291
6.9.23	a2w::UserData	299
6.9.24	a2w::Vector [NEW]	303
6.10	AFP2web Messages	306
6.10.1	General Information About Error Messages, Return Codes, and Error Files	306
6.10.2	AFP2web-specific Messages	307
6.10.3	Maas Image Library Messages	318
6.11	AFP2web Conversion Process	324
6.11.1	Auto image grouping	324
6.11.2	Windows maximum memory usages limitation per conversion process	324
6.11.3	AFP output, MOD:CA IS1 / IS2 extended feature	325
6.11.4	Creating Color, Grayscale and B/W output	325
7	Appendix	326
7.1	Migrating Scripting Facility Modules to Latest Version	326
7.2	Tutorials: Using the Scripting Facility	329
7.2.1	Introduction to the Scripting Facility Tutorials	329
7.2.2	addAnnotations.pm: Adding a Hypertext Annotation	331
7.2.3	addBookmarks.pm: Adding Bookmarks to the PDF output document	332
7.2.4	addUserData.pm: Adding User Data to the Document	335
7.2.5	addWatermark.pm: Adding Watermark Text to Pages	338
7.2.6	afp2xml.pm: Script Used to Produce XML Out of AFP	339
7.2.7	any2html.pm: Module to build HTML output	343
7.2.8	autosplit.pm: Splitting an AFP Spoolfile into Documents and Pages	353
7.2.9	dumpIniParms.pm: Dumping INI Parameters	359
7.2.10	dumpMediumMaps.pm: Output List of Medium Maps to a Dump File	361

7.2.11	dumpNOPs.pm: Output the Contents of NOP Fields Into a Dump File	363
7.2.12	dumpPageTextObjs.pm: Dumping Text and Font Information	365
7.2.13	eyecatcher_lpd.pm: Process and Format LPD text objects.....	369
7.2.14	eyecatcher_mmd.pm: Find and Extract MMD Text Objects.....	378
7.2.15	eyecatcher.pm: Find and Extract Text Objects.....	384
7.2.16	processDocInfo.pm: Access/Process document information.....	390
7.2.17	sortPageTextObjs.pm: Sorting Text Objects in their Print Sequence.....	397
7.3	PDF/A Support in AFP2web.....	401
7.3.1	AFP2web's Commitment to the PDF/A Standard.....	401
7.3.2	Important Notices	401
7.3.3	INI Parameters and Command Line Options	401
7.3.4	Fonts.....	402
7.3.5	Metadata in XMP Format.....	403
7.3.6	Features Not to Be Used for PDF/A	403
	List of Figures.....	405
	List of Tables	406
Index	407	

1 Preface to the AFP2web User Guide 4.x

This preface offers

- a short introduction to the AFP2web User Guide
- a list of the features, which are new in Version 4.x
- notices to be aware of before using AFP2web
- a checklist for migrating from AFP2web 2.x to Version 4.x

1.1 About the AFP2web and Scripting Facility User's Guide

AFP2web is a utility for the intelligent recognition, separation, conversion, indexing, and distribution of AFP mass print data.

AFP2web converts AFP spool files to the standard formats PDF, TIFF, ASCII, and XML and PDF documents to AFP. With AFP2web, it is possible to web-enable AFP data and yet keep the benefits of using the secure and powerful IBM S/390 or Unix systems. AFP2web is used for web-enabling, archiving, indexing, document exchange in workflows, for producing high-quality true fidelity output of conversion results, and as a component in a variety of application scenarios.

The AFP2web Scripting Facility is an enhancement to AFP2web. It provides a scripting interface that is used to intelligently control document recognition, document splitting, index extraction, and much more.

1.1.1 Who Should Read this Document

This document is intended for all users of AFP2web. To use AFP2web, it is sufficient to set up a job environment and the configuration files of AFP2web. Users who need to customize AFP2web will find information on how to do this with the Scripting Facility.

1.1.2 What's in this Document

This document is organized into the following chapters:

- This preface with an overview of the new features of AFP2web Version 4.x, important notices, recommendations for migrating from Version 2.x to Version 4.x
- Introduction to AFP2web, its functionality and possible application scenarios.
- AFP2web User's Guide, which includes information about installing, configuring AFP2web for your application, general information on how to run AFP2web, and how to use basic processing features.
- AFP2web font handling, which points out that AFP2web uses the original AFP font resources without the need for your intervention. However, AFP2web still offers processing modes, which are described here and can be used to substitute fonts with replacement fonts.
- The AFP2web Scripting Facility Guide, which serves as a high level introduction to using the Scripting Facility.
- The AFP2web Reference with detailed information about the syntax and meanings of the AFP2web configuration parameters (INI file, command line options, mapping.def), the AFP font naming convention used for missing font information, the Scripting Facility quick reference and API reference, and finally a list of messages and error codes.
- The Appendix offers additional information, such as a checklist for migrating Scripting Facility modules to Version 4.x, and a tutorial, which introduces the Scripting Facility examples.

AFP2web Version 4.3.x.x User Guide and Reference

1.1.3 Where to Find Additional Information

A source for more in-depth information on AFP is <http://www.printers.ibm.com>.

If you want to learn more about Perl, start with the homepage <http://www.cpan.org/>.

1.1.3.1 Whom to Contact

This manual provides the information you need to get started with AFP2web and the Scripting Facility. Maas Holding GmbH gladly constructs a complete Scripting Facility module for your specific AFP print data format. You can then use this script as a starting point for further customizing. We also recommend using the Scripting Facility examples, which are delivered with AFP2web.

Maas Holding GmbH offers professional services and consulting related to the development and deployment of AFP2web-based solutions.

For more information, please visit our Web site at <http://www.oxseed.com/afp2web>

or contact us via: E-mail: afp2web@oxseed.com

1.1.4 Conventions

1.1.4.1 XML Data

In examples showing XML Documents we break and indent lines to make them more readable. Coding examples, listings, output to the system console are rendered in a fixed-width font.

Highlighting Convention for XML Data:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Index SYSTEM "Index.dtd" >
<Index Doctype="PDF" DocName="pdf/xml/INSURE_s3k.0.pdf"
  PageCount="3" Size="16726">
  <PageGroupIndex PageGroup="00000001">
    <Data>
      <Name>Insured</Name>
      <Value>Geoffrey R Stephens</Value>
    </Data>
    <Data />
  </PageGroupIndex>
</Index>
```

1.2 Change History of AFP2web 4.x

1.2.1 New Since AFP2web Version 3.x

This section summarizes the product features, which are new since V3.x. You can find more up-to-date information about changes to the product in the readme-files, which are delivered with the product.

1.2.1.1 AFP to PDF Conversion

AFP input referring fonts (at Active Environment Group using Map Coded Font) based on Font Global ID (FGID) and Global Character Set Global ID (GCSGID) is now handled by AFP2web. For this, AFP2web mapping.def has been extended with a new section called "[GCSGID]". This section helps to map AFP character set resource equivalent to

given FGID and GCSGID. For more information, please refer Chapter "Parameters of mapping.def" under "AFP2web Reference".

1.2.1.2 LPD/MMD to Any Conversion

LPD/MMD input has been modified with new parameters to specify input format and spool file type.

- "-lpdin" MUST BE used to specify LPD input from now onwards
- "-mmdin" MUST BE used to specify MMD input from now onwards
- "-sft" MUST BE specified with a valid Spool File Type section name
- File creation mode MUST BE set to "DOC_COLD" (either using "FileCreationMode" ini attribute or using "-doc_cold" command line option)

1.2.1.3 AFP Input Features

AFP2web offers major improvements in processing AFP spool files:

- AFP2web is also capable of handling "Code 39" barcode type.

1.2.1.4 New Features of the Scripting Facility

- Below given functions are deprecated from V3.x onwards

- a2w::Config::getIndexFilePath
- a2w::Config::getOutputFilePath
- a2w::Config::getScriptsArgs
- a2w::Config::setAutoSplit
- a2w::Document::addIndex(Name , Value)
- a2w::Page::addIndex(Name , Value)
- a2w::Page::addObject
- a2w::Page::includeObject

- Added new Scripting Facility module "a2w::Comment" with the following APIs

- new
- setValue
- getValue
- remove

- Added new APIs in a2w::Document module

- addComment
- getFirstComment
- getNextComment
- getProperty
- getMetaDataStream

- Added new API in a2w::Kernel module

- getLastErrorMessage

- Organized Scripting Facility modules to use following constants

- a2w::ConfigConstants
- a2w::DocumentConstants
- a2w::FontConstants
- a2w::PageConstants

1.2.1.5 **afp2web.ini and command line options**

New INI parameters and command line options include:

- LoggingLevel with new logging levels: 8, SF, RES, OLY, PSEG, FNTR, CDP, MMAP, FDEF, IOB, PDFINFO, INWARN, OUTWARN. The corresponding command line option is -ll.
- Section [SpoolFileType] to define carriage control type, code page to be used, record format and record length for LPD/MMD input formats
- AFPIs2ComplianceFlags: Flags that specify conditions to be applied for AFP output with IS2 compliance. The corresponding command line option is -is2cf.
- TextPresentationFidelity: To control text presentation for AFP output. The corresponding command line option is -tpf.
- PDFParserConfFile: PDF parser configuration file with path. The corresponding command line option is -ppcf.
- ImageProcessValues: Key to specify default values for missing input image properties like resolution. The corresponding command line option is -ipv.

1.2.1.6 **Additional Enhancements**

Additional enhancements include:

- AFP2web displays its version information in the following format:

AFP2web v<version number> [Built for <OS name> <Machine architecture> on <Date> at <Time>]

- When input spool contain objects that are not supported by AFP2web, AFP2web returns a special return code "2". Applicable now, only for PDF input spools.

1.2.2 **New Since AFP2web Version 2.x**

AFP2web and Scripting Facility is a product bundle, which you use for any level of application complexity. AFP2web offers a core functionality and the Scripting Facility to further customize AFP2web to meet specific demands.

Strong in its font mapping capabilities, PDF document security features, and its powerful Scripting Facility, AFP2web Version 3.x has evolved from a conversion tool to a versatile document conversion and processing framework.

This section summarizes the product features, which are new. You can find more up-to-date information about changes to the product in the readme-files, which are delivered with the product.

1.2.2.1 PDF to AFP Conversion

AFP2web converts PDF documents to AFP format, which is compliant with the MOD:CA P or MOD:CA Interchange Set 2.

New INI parameters are introduced: PrintableLineWidth, CurveSamplingFactor, AFPComplianceLevel.

The INI parameter PageRotation has been enhanced.

The mapping.def [CHARSET RENDERING] section has been extended for finer control of font rasterizing. The new section [AFPFONT] has been added to specify replacement fonts.

When font rasterizing is turned on for the PDF to AFP conversion, AFP2web uses the fonts, which it finds in the input files. AFP2web will also search for external referenced fonts.

1.2.2.2 Font Support Enhancements

AFP2web offers major improvements in its font handling capabilities:

- Font Rasterizing is a new font processing mode, which uses the original AFP fonts, thus producing output documents with 100% true fidelity.
- Font Referencing/Substitution and Font Embedding are font processing modes which replace AFP fonts with their corresponding substitution fonts. The supported font types are: Type 1, TrueType, and OpenType containing TrueType as a single subset.

1.2.2.3 AFP Input Features

AFP2web offers major improvements in processing AFP spool files:

- Support of colored graphics in form of IOCA FS 42 and 45, GOCA and Object Container (TIFF, JPEG,...).
- AFP2web is capable of handling barcodes: Code 128. AFP2web uses Character Set B of Code 128 to draw the barcodes because it includes lowercase letters, uppercase letters, numbers and punctuation characters.
- AFP2web also supports various formats of the bi-dimensional DataMatrix 2D barcode. For more information, please refer to "Notes on Barcode Support" on page 51.
- Support of image compression algorithms is enhanced: Uncompressed, IBM RL4, LZW, G3, G4, Old JPEG, JPEG, IBM MMR, IOCA (FS 10, 11, 42, 45), CCITTRLE, CCITTRLEW, IT8CTPAD, IT8LW, IT8MP, IT8BL, PIXARFILM, PIXARLOG, DEFLATE, ADOBE_DEFLATE, THUNDERSCAN, DCS, NEXT, SGILOG, SGILOG24. (Note: AFP2web does not support CMYK (32Bits/Pixel) JPEGs.)
- FTP access to remote resources. You can configure AFP2web to retrieve AFP resources from any FTP server.

1.2.2.4 PDF and PDF/A Output Features

AFP2web offers major improvements for output to PDF:

- Supports PDF Security Option 40/128 bit, which allows the creator to retain control of the document and associated rights.
- It is possible to define Adobe Acrobat Reader preferences for resulting PDF files.
- PDF documents remain full-text-searchable, even when the original AFP fonts are used.
- To reduce PDF file size, AFP2web has been optimized to use a best-match compression algorithm depending on the image type being either black/white or color.
- Easy document adaptation: changing and amending of text, indexes, annotations, bookmarks and many more.

AFP2web supports the new output format PDF/A, which conforms to the PDF/A-1b standard. The standard is defined in ISO 19005-1. Document management — Electronic document file format for long-term preservation — Part 1: Use of PDF 1.4 (PDF/A-1) Reference number ISO 19005-1:2005(E) and Corrigendum 1 (ISO 19005-1:2005, TECHNICAL CORRIGENDUM 1, Published on 2007-04-01, Corrected version of 2005-12-01).

PDF/A-compliant documents are self contained PDF files, which are used for long-term archival. The standard is based on PDF Version 1.4 and imposes restrictions on the PDF functionality.

1.2.2.5 Features for Raster Formats

AFP2web offers major improvements for raster formats, such as TIFF:

- For raster format input, the support of the compression algorithms is enhanced: Uncompressed, LZW, G3, G4, Old JPEG, JPEG, CCITTRLE, CCITTRLEW, IT8CTPAD, IT8LW, IT8MP, IT8BL, PIXARFILM, PIXARLOG, DEFLATE, ADOBE_DEFLATE, THUNDERSCAN, DCS, NEXT, SGILOG, SGILOG24.
- For output to raster format: Easy document adaptation: changing and amending of text, indexes, and many more.

For output to TIFF, the following AFP2web-specific tags are added to insert metadata from the INI-file:

TIFF Tag	Contents
667	Keyword
668	Subject
669	Title

When opening TIFF files generated by AFP2web, some TIFF viewers may issue a warning "Unknown field/tag" because of these custom tags. This type of warning can be considered insignificant.

1.2.2.6 New Features of the Scripting Facility

The Scripting Facility offers access to printable and non-printable page content before the conversion process. New functions make it possible for you to add, modify, or delete page content.

The Scripting Facility can output to memory streams, thus avoiding time consuming file read and write.

You can find more information on the Scripting Facility in this document. A set of sample Scripting Facility modules are provided, which you can use as a starting point for your own solutions.

1.2.2.7 Supported Operating Systems

The list of supported operating systems has increased. For more information, please refer to the list under [System Prerequisites](#).

1.2.2.8 afp2web.ini and command line options

New INI parameters and command line options include:

- Section [AFPColorTable] to map each color of the AFP standard OCA color table to their corresponding RGB values. The corresponding command line option is available for specifying a list of colors, such as: - clr:<colorname>,<r>,<g>,,<colorname>,...
- Autosplit is an INI-file parameter for automatically splitting AFP spool files into documents and pages. This parameter affects the processing of the Scripting Facility.

- CMYKTORGB to convert IOCA FS45 CMYK color image planes to RGB. The corresponding command line option is used to switch this function off: -nocmyktorgb.
- FilenamePattern to specify the pattern for the names of the output files.
- LoggingLevel with new logging levels: 8, SF, RES, OLY, PSEG, FNTR, CDP, MMAP, FDEF, IOB. The corresponding command line option is -ll.
- PDFSecurity to set rights for accessing resulting PDF documents.
- PDFUIOptions, PDFWinOptions to set preferences for displaying a document in the PDF viewer. The corresponding command line options are -pui and -pwin.
- SkipPage, SkipObjectSize are new parameters that specify pages to ignore in the AFP spool file.
- User-defined sections to specify the FTP location of remote AFP resources.

1.2.2.9 Additional Enhancements

Additional enhancements include:

- Image Handling Enhancement: Faster image handling, better image compression, smaller PDF files.
- Color images to gray scaled images conversion for B&W output
- AFP2web displays its version information in the following format: AFP2web Version <version number> [Built for <OS name> on <Date> at <Time>]

1.3 Deprecated Since AFP2web 4.x

This section lists the product features, which AFP2web Version Version 4.x no longer supports.

1.3.1 Font Metric Files

AFP2web uses the original fonts and does not require additional font formatting hints. Thus, AFP2web no longer supports the use of font metric files.

1.4 Important Notices for AFP2web 4.x

Please use the AFP2web and Scripting Facility with great care. Incorrect settings of AFP2web parameters can affect the conversion results. In the worst case, documents might become useless for archiving purposes.

We therefore recommend:

1.4.1 Test thoroughly!

You should thoroughly test the run time parameters, the resources, and the font settings against each type of AFP document. Especially when you process documents for archival.

1.4.2 Only you know your fonts!

AFP2web cannot automatically detect any customizations you have made to your resources, for example any additions to your code pages. In doubt, please contact Maas Holding GmbH to resolve any problems.

1.4.3 Make backups of your documents!

AFP2web does not replicate data for backup. It is your responsibility to ensure that you have backup copies of the data you pass to AFP2web.

1.4.4 Backup your previous installation of AFP2web!

Save your previous customized versions of your configuration files before you install a new release of AFP2web. Please note that you must migrate your old customizations to the new requirements as described later on in this document.

We recommend preserving your old versions of the files:

- INI file
- afp2web.pm (or your custom scripts for the Scripting Facility)
- mapping.def
- CP files (*.cp)
- fonts for PDF (*.pfm and *.pfb files)

Rename the target folder of your previous A2W installation (Example: A2W_version_date). Create a new target folder for the new installation using the old folder name (Example: A2W). After installing a new release of AFP2web, you can migrate your customizations to the new configuration files.

1.4.5 Use the AFP2web Scripting Facility with care!

Using AFP2web without care can result in loss of data. The appearance of documents can be distorted after conversion or the identification of documents can be lost because of incorrect document splitting.

We therefore recommend:

- You can use different instances of the Perl script afp2web.pm to handle different types of spool files. To do this, you use the -sp option to specify the Scripting Facility module to use.
- Please be sure to test the results of your programming in afp2web.pm. Your scripts are responsible for page skipping, document splitting, and the processing of index data.

1.5 Migrating from AFP2web Version 2.x to Version 3.x/4.x

AFP2web Version 4.x has new enhancements, which require changes in the configuration files and in the scripts used for the Scripting Facility.

Any customizations in your previous version of AFP2web might no longer be necessary or should be adapted to the new requirements described below.

1.5.1 Changes in the mapping.def

AFP2web has improved its font handling functionality. The required font definitions made in the previous version of AFP2web are now obsolete because AFP2web now uses the original AFP fonts. In most cases, the mapping.def entries required for AFP2web Version 2.x may be removed. However, there might be cases, where substitution fonts need to

AFP2web Version 4.3.x.x User Guide and Reference

Preface to the AFP2web User Guide 4.x -Migrating from AFP2web Version 2.x to Version 3.x/4.x

be used (font substitution, font referencing). In this case, the configuration files require definitions in the mapping.def.

AFP2web Version 3.x has the following changes in the definition of the mapping.def:

- In the [PDFFont] section, font names are no longer prefixed.
- [CHARSET RENDERING] section is new and specifies when to override AFP2web's default font processing.
- The [FONTSUFFIX] section is used to specify suffixes to the names of your font files. Because there is no standard convention followed for font file names, this enhancement will give you the freedom of specifying one or more conventions.

AFP2web Version Version 4.x has the following changes in the definition of the mapping.def:

- [GCSGID] section is new and specifies charset name to the corresponding gcsgid/fgid.

1.5.2 Changes in the CP Files

AFP2web Version 3.x has extended the format of the code page files (CP files) for Unicode support. This extension is required for adding text search capability to the PDF files.

1.5.3 Defining and Supplying Font Resources

AFP2web no longer requires a substitution font to be used as a replacement for an AFP font. If the AFP font resource is available either in-line with the AFP spool file or as external AFP font resource, then AFP2web uses this font.

However, there might be reasons for replacing an AFP font and for referencing or embedding a substitution font. AFP2web still offers its basic font replacement functionality. The AFP font resource delivers the basic attributes for a font, which determine which substitution font file to use. In addition, you can specify naming conventions for the suffixes to add to the name of font files.

In the previous version of AFP2web, fonts for the conversion to TIFF had to be installed in the system. AFP2web Version Version 4.x now will install a required font at runtime, if necessary, and will un-install the font upon process completion.

For more information, please refer to [Handling Fonts](#).

1.5.4 Scripting Facility Modifications

- You must modify your custom scripts for the Scripting Facility. The Scripting Facility has changed the name of its packages and methods. For a detailed list, refer to [Migrating Scripting Facility Modules to Version 4.x](#).
- The Scripting Facility has greatly enhanced its functionality. For a quick overview of all available packages and methods, please refer to [Scripting Facility Quick Reference](#). You can find the detailed API reference in this user guide.
- Replaced processPDFBookmarks.pm scripting facility module with processDocInfo.pm scripting facility module.
- Added "eyecatcher_lpd.pm" and "eyecatcher_mmd.pm" scripting facility modules to demonstrate LPD/MMD input format support

AFP2web Version 4.3.x.x User Guide and Reference

Preface to the AFP2web User Guide 4.x -Migrating from AFP2web Version 2.x to Version 3.x/4.x

2 Introduction to AFP2web 4.x

This chapter gives a quick overview of AFP2web:

- Input and output formats supported for document conversion
- AFP2web core components
- Components used to build AFP2web applications
- Typical AFP2web application scenarios

2.1 Overview of the AFP2web 4.x Functionality

In the past, the standard for mass print data was tied to proprietary AFP technology. Today, AFP2web opens the door to a more effective communication. AFP2web makes AFP documents available to any kind of open system by converting them to the de facto standard Internet data formats, such as PDF, JPEG, ASCII, TIFF, XML. AFP2web enables an intelligent recognition, separation, conversion, indexing and distribution of mass print data.

One product focus lies on producing XML output for easy integration with upcoming Internet technology-based print solutions and for powerful data mining from print documents. AFP2web has proved to be inspiring. New applications are evolving. For more information about AFP2web solutions, please refer to the Web site <http://afp2web.com>.

In this section, we describe:

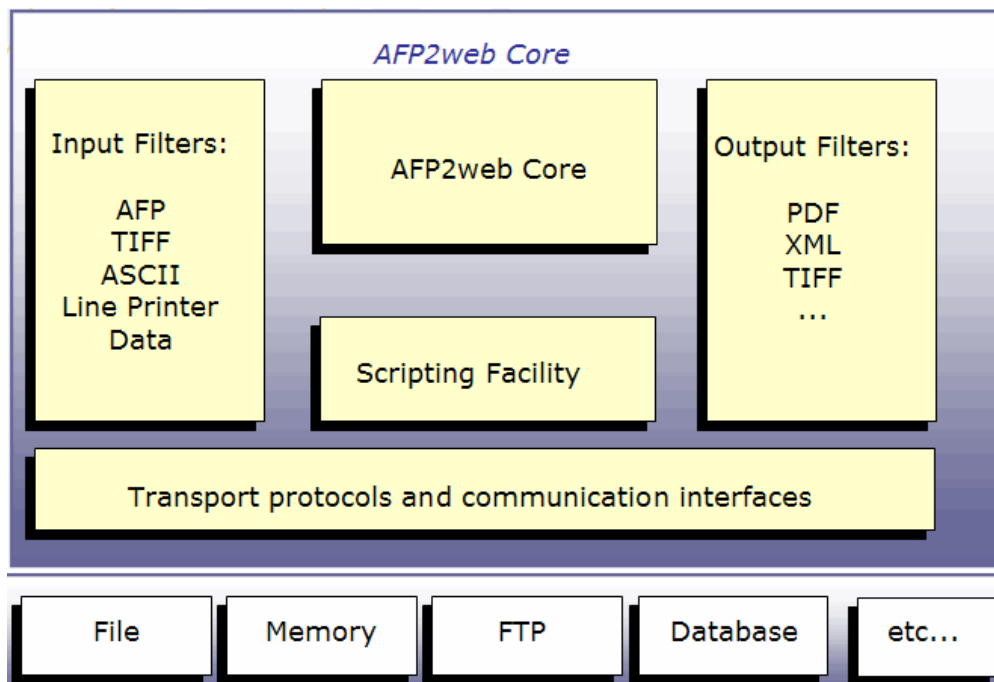
- Supported Formats
- AFP2web core components
- Methods of Integrating AFP2web
- AFP2web application scenarios.

2.1.1 Core Components of AFP2web 4.x

The future is open to AFP2web. The internal structure of this document conversion tool is highly modularized. The internal document model is mapped to internal adapter interfaces, which can accommodate new input and output formats.

One of its major strengths is AFP2web's communication layer and its interface adapters. Daily requirements of many customers of AFP2web include the option of selecting any common channel for reading input and writing output: via File IO, in-memory buffer streams, remote FTP, proprietary databases, and any other channel, if required.

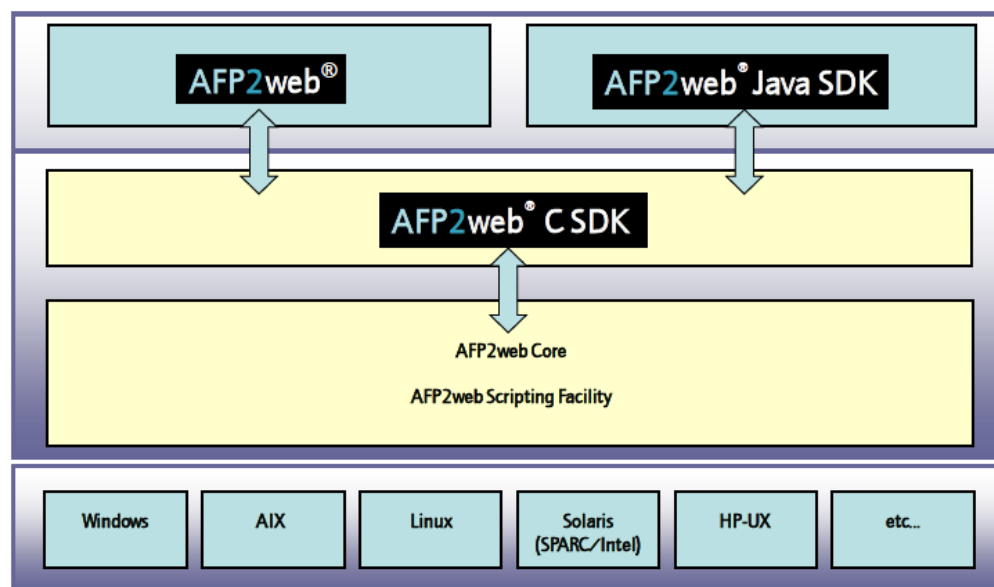
The following figure shows the internal module structure of AFP2web:



2.1.2 Methods of Integrating AFP2web 4.x

AFP2web offers a powerful base functionality, which is configured for the specific application. In its basic version, AFP2web is started as a console application from the command line to process AFP spool files. The AFP2web Scripting Facility is an enhancement, which can be used to formulate processing rules and to trigger external programs. Both AFP2web and the Scripting Facility enable intelligent and flexible document control.

The following figure shows how AFP2web components combine to build AFP2web products:



The optional Software Development Kit (SDK) is available for integrating AFP2web into existing document management or content management solutions for on-demand document conversion. The AFP2web SDK provides











the AFP2web Document Management System (DMS) Application Programming Interface (API). The AFP2web DMS API has functions for input/output via in-memory buffer streams (afp2web) and file IO (afp2web_f). In-memory buffer streams are typically used for on-demand conversion of single documents. File input and output are used to process mass print data in spool files.

The JNI interface ensures easy integration into J2EE-Server environments such as IBM WebSphere.

2.1.3 Supported Formats in AFP2web 4.x


There is a matching solution for every conversion requirement. In addition to the AFP2web basic software, many options are available. Almost any combination of input and output formats can be selected. The AFP2web format options include converting documents from AFP or TIFF to ASCII, JPEG, PDF, TIFF, XML and other common formats.

The following figure shows the format options, which are available.

INPUT	OUTPUT									
										
AFP	➔	AFP	TIFF	PDF	PDF/A	JPEG	PNG	ASCII	XML	HTML
		X	X	X	X	X	X	X	X	X


Notes:

- For AFP2AFP conversion, It is NOT ALLOWED to convert a MOD:CA-P compliant AFP Input.Spool to a MO:DCA-IS2 compliant AFP output.Spool

RASTER	➔	AFP	TIFF	PDF	PDF/A	JPEG	PNG	ASCII	XML	HTML
		X	X	X	X	X	X			X


Notes:

- "RASTER" stands for pixel formats, such as TIFF, JPEG, PNG.
- Pixel data do not transport "content", it is therefore not possible to convert pixel data to meaningful text in ASCII or XML format.


PDF	➔	AFP	TIFF	PDF	PDF/A	JPEG	PNG	ASCII	XML	HTML
		X	X	X	X	X	X	X	X	X

Notes:

- Output to PDF and PDF/A is planned

LPD	➔	AFP	TIFF	PDF	PDF/A	JPEG	PNG	ASCII	XML	HTML
		X	X	X	X	X	X	X	X	X

ASCII	→	AFP	TIFF	PDF	PDF/A	JPEG	PNG	ASCII	XML	HTML
ASCII)		X	X	X	X	X	X	X	X	X

MMD	➡	AFP	TIFF	PDF	PDF/A	JPEG	PNG	ASCII	XML	HTML
		X	X	X	X	X	X	X	X	X

2.2 Application Scenarios for AFP2web 4.x

In the following, we describe typical application scenarios for AFP2web:

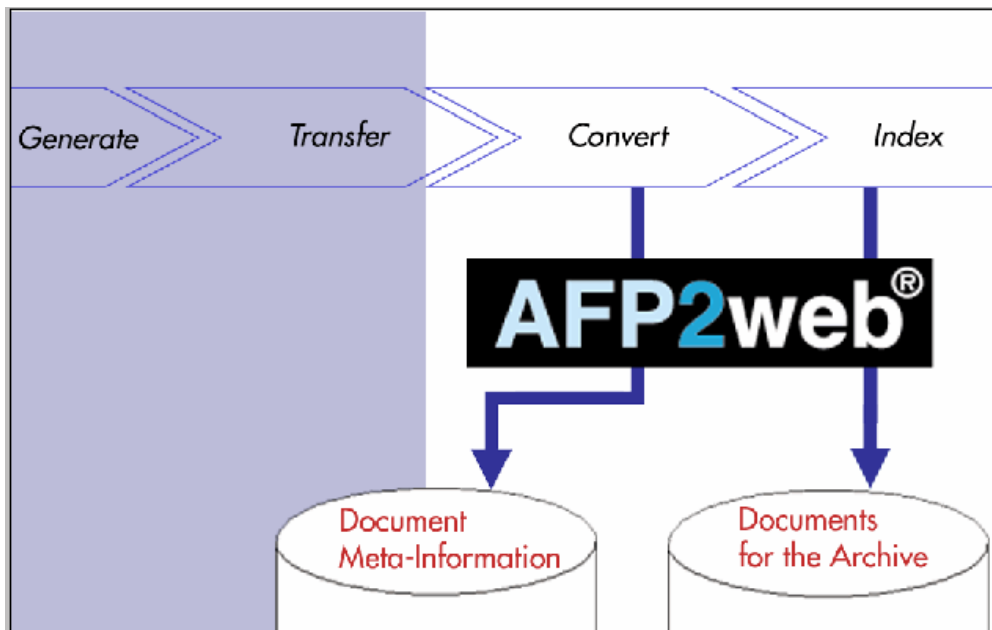
- Document archival and index update
- On-the-fly conversion for PDF on demand

- AFP viewer for workstations
- Document distribution per post (print), e-mail, or fax

2.2.1 Archival of Documents and Index Update

This is the scenario for an automated conversion, indexing and archival of documents. The process runs in batch mode and converts mass print data. The conversion process must run smoothly, efficiently, reliably, and must produce high-quality conversion results.

Figure 1 Scenario: Document Archival and Index Update



The document archival process is a workflow, which interfaces with AFP2web to convert the documents for archival. The steps of this process include:

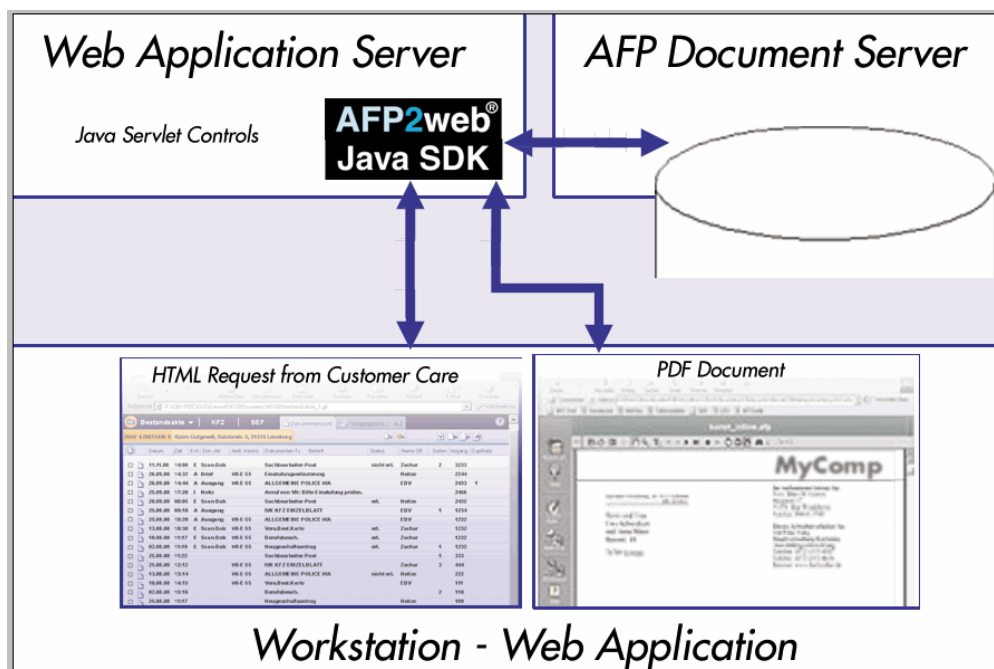
1. AFP data generation
2. AFP data transfer (via FTP)
3. AFP2web document conversion (PDF, one of the raster formats, or XML)
4. Document archival on the document server, delivering a technical document ID
5. Update of the index database with document meta information (delivered by AFP2web) and the technical document ID

AFP2web not only delivers excellent conversion quality. AFP2web can also be used as a central data extraction utility for standard AFP indexes. Moreover, the AFP2web Scripting Facility can be used to fine-tune the data extraction for any type of information in the document. The Scripting Facility can be used as a link for follow-on processing, for example in document-oriented work flows.

2.2.2 On-the-fly Conversion for PDF On-Demand

Display individual AFP documents on request in a front end Web browser. The desired format is PDF delivering a true 1:1 rendition of the original AFP document.

Figure 2 Scenario: On-the-Fly Conversion for PDF On-Demand

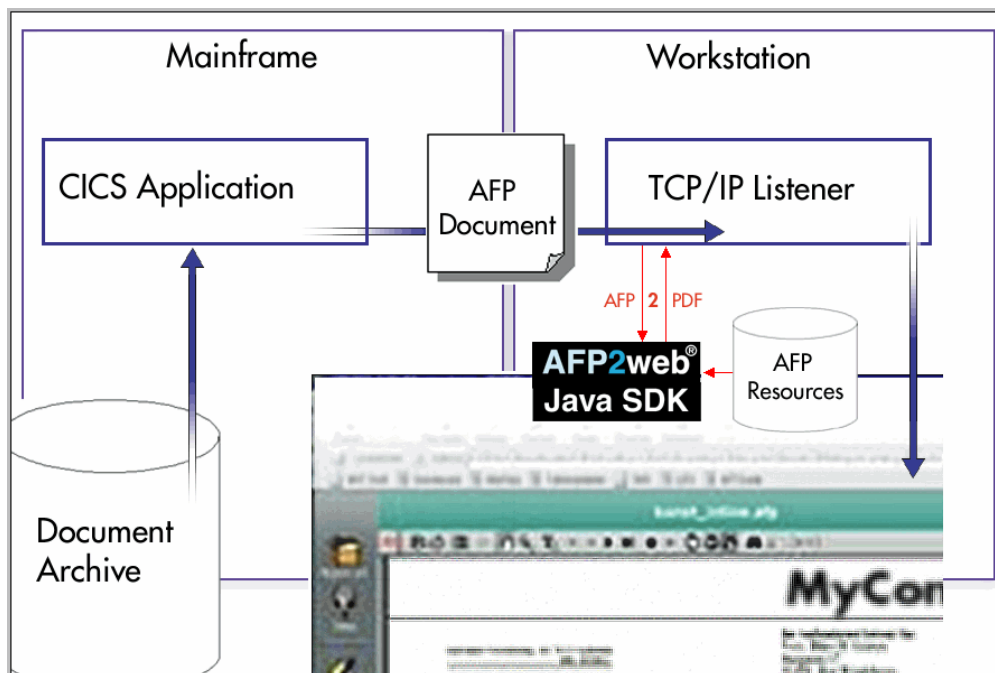


AFP2web is used, for example, as a component in a three-tier application, typically in an Internet application. Controlled by a Java servlet, AFP2web reads the AFP document from the input buffer and writes the PDF document to the output buffer. AFP2web avoids file input and output, works with in-memory data streams, and is thus optimized for quick processing. The efficiency and the high quality conversion make it possible to view and process documents quickly, online, and in distant branch offices.

2.2.3 AFP Viewer on Workstations

A document in proprietary AFP format, which is retrieved from the archive, needs to be delivered on demand and viewed as PDF online in a Web browser.

Figure 3 Scenario: AFP Viewer on Workstations



Using the AFP2web Java Source Development Kit, AFP2web is implemented as a standalone server, satisfying on-demand conversion requests. Combined with any PDF or TIFF viewer, AFP2web is the ideal AFP viewer for displaying AFP documents. AFP2web offers efficiency and high quality, on-the-fly conversion results.

2.2.4 Document Distribution per Post (Print), E-Mail, or Fax

Convert AFP documents to PDF or TIFF. Dispatch documents per post, e-mail, or fax.

No matter whether the output format be PDF or TIFF, produce the format for your purposes and:

- use an e-mail or fax server to quickly deliver AFP data.
- combine AFP2web with a print server, and be able to print AFP in color on non-AFP printers.

3 AFP2web 4.x User Guide

This chapter describes how to:

- Install AFP2web
- Configure the system for AFP2web

You will also find the information you need to work with AFP2web, including:

- A quick start in using the demo files.
- How to start and stop AFP2web processing
- How to use specific features of AFP2web
- How to read the AFP2web error messages and log files to resolve any problems.

3.1 Installing AFP2web 4.x

3.1.1 System Prerequisites for AFP2web 4.x

3.1.1.1 Hardware and Operating System Requirements

AFP2web is an application that runs on these operating systems:

- Windows XP/Server 2003 R2/Server 2008 R2/7 64-bit
- Windows XP/7 32-bit
- Linux with x86 processor or compatible processor (for ex: AMD)
- Linux with x64 processor or compatible processor (for ex: AMD)
- AIX 5.3 32-bit RS/6000 PowerPC
- AIX 5.3 64-bit RS/6000 PowerPC
- Sun Solaris 7 (SunOS 5.7) 32-bit on an Ultra-250 Sparc
- HP-UX 11.i 64-bit C3600a PA-RISC
- zLinux SuSE SLES7

The minimum amount of random memory should be available:

- At least 2 GB RAM.

3.1.1.2 Additional System Requirements

Run time Requirements

- Windows 64-bit must be installed with Microsoft Visual C++ 2010 Redistributable Package, which can be obtained from link "<http://www.microsoft.com/en-us/download/details.aspx?id=14632>"
- HP-UX must have GNU-Compiler V3.4.2 or less
- Linux 32-bit must have GNU libc (GLIBC) version 2.3.3 or more
- Linux 64-bit must have GNU libc(GLIBC) version 2.3.6 or more

- Aix 32-bit/64-bit must have LibC library (bos.rte.libc) level 5.3.0.61 or more
- Solaris 32-bit must have LibC library version SUNW_1.1, SUNWprivate_1.1 or more

PERL

For the following list of operating systems (processors), AFP2web has the Perl engine included. If you need to use your own Perl environment, please install Perl version information as given below

The Perl engines included in AFP2web are:

- Windows (Intel) 32-bit: Perl 5.8.7
- Windows (Intel) 64-bit: Perl 5.8.9
- Aix 5.1 (RISC): Perl 5.8.7
- Solaris 2.7 (SPARC): Perl 5.8.7
- Suse Linux 9.1 (Intel): Perl 5.8.7
- Debian Linux 4.0 (AMD) 64-bit: Perl 5.8.8
- HP UX 64: Perl 5.8.7

3.1.1.3 Hard Disk Capacity

Plan for enough storage to hold your input and output files. To estimate the amount of storage required, the following figures might help:

- resulting TIFF file is approximately 10 times larger than the respective document in the AFP data stream. This applies when using "G4" as the compression option. The factor is about 40 when using the option "packed". The factor is 100 when using no compression.
- A resulting PDF file is about 1 to 2 times larger than the respective document in the AFP data stream. This value depends on the structure of the AFP document.
- Use the PDFDocLimits parameter of the INI file to specify or increase the limits of memory resources. Using PDFDocLimits might have an impact on performance. When used with DOC_COLD (Scripting Facility) memory allocation will happen for each resulting PDF document. PDFDocLimits should therefore be set to the lowest possible value to increase performance. When used with -DOC option, memory allocation will only happen once, thus there is no side effect on performance.
- For each font embedded in the PDF file, the size of the PDF file will increase by an average of about 30 KB to 50 KB.
- If you use the option -sod for included objects, AFP2web writes these objects to disk to speed up processing and to highly optimize the object data for conversion. In this case, plan for extra hard disk capacity.
- Log files produced by AFP2web can be very large. AFP2web will also suffer a degrade in performance when producing log files during the conversion. Log files are normally used only during development and integration not in a production environment.

3.1.1.4 Fonts for the Conversion

If you specify this in the mapping.def as a font processing mode or if AFP2web cannot find the AFP font, AFP2web substitutes AFP fonts by those fonts specified in the configuration files.

The requirements are:

For a raster format (TIFF, JPEG, PNG, AFP): The required fonts should be installed in the Windows.

In the previous version of AFP2web, fonts for the conversion to TIFF had to be installed in the system. AFP2web Version 4.x now will install a required font at runtime, if necessary, and will un-install the font upon process completion.

For PDF: AFP2web either references or embeds the fonts in the PDF file.

3.1.2 Installing AFP2web 4.x on Windows

3.1.2.1 Preparation

It is recommended that you save the configuration files of your previous AFP2web installation. You might have customizations, which you want to redefine in the new configuration files.

If you followed our installation recommendations you already have a backup copy of the following files:

- INI file
- afp2web.pm (or your custom scripts for the Scripting Facility)
- mapping.def
- code page (*.cp) files

3.1.2.2 Installation

To install AFP2web on Windows:

1. Start the setup routine afp2web*.exe The setup routine is a self extracting file, which stores all AFP2web installation files in the selected target directory.
2. Select the target directory for installing all files.
Click the button [...] to select a directory
Select the option **Confirm overwrites** if you want to prevent overwriting existing files.
Click **OK** to start the installation process.
The dialog Unpacking files shows the progress indicator: The setup routine unpacks all files to the target directory and creates default sub directories in a few seconds. You can stop installation by clicking **Cancel**.
3. If the dialog **File exists, overwrite? ...** appears, confirm with:
Yes if you want to overwrite an existing file
No to prevent overwriting this file
Cancel if you want to stop installing at this point.

3.1.3 Installing AFP2web 4.x on Unix

To install AFP2web on UNIX you unpack the file afp2web.tgz.

- If "tar" command support "z" option (to decompress gz) issue following command

```
tar xvzf <afp2web release package filename>.tgz
```

- Else issue following commands

```
gzip -d <afp2web release package filename>.tgz
tar xvf <afp2web release package filename>.tar
```

3.1.4 AFP2web 4.x Installation Results

3.1.4.1 AFP2web (common for all products)

The following files are installed for AFP2web (common for all products):

Directory	Files	Description
(target directory)	afp2web.ini	Configuration file with general settings for AFP2web
	demo.bat (Windows) demo (Unix)	Batch file as example of calling the program
	history.txt	Change history
	license*.txt	Licensing information "_en" file name suffix means English "_de" file name suffix means German
	readme.txt	Last-minute information
	afp2web.pm	AFP2web Scripting Facility script template
	Perl*.dll	PERL DLL offered by AFP2web for WINDOWS OS ONLY
a2w	Perl modules (*.pm)	PERL packages of the AFP2web Scripting Facility
afpcp	codepages (*.cp)	Code page mapping files
	mapping.def	Configuration file, used to map AFP-specific fonts to substitution fonts
	gcid2unicode.def	Default Unicode mapping file.
doc	afp2web_*.pdf	AFP2web User's Guide, When file name contains "en" means English User Guide "de" means German User Guide
log		Target directory for log files
pdf		Target directory for converted documents (Note: DTDs are included for the afp2xml.pm example)
extfont	*.pfb/*.pfm, *.ttf files	Directory for additional Adobe Type 1 and TrueType fonts used for the conversion to TIFF and PDF

sfsamples	*.pm	Scripting Facility examples
sfsamples\XML	Writer.pm	PERL module required by "afp2xml.pm" Scripting Facility sample module
samples		Sample files used to demonstrate AFP2web
samples\resource		Resources for the sample files

3.1.4.2 AFP2web batch product

The following files are installed for AFP2web batch product:

Directory	Files	Description
(target directory)	afp2web.exe (Windows) afp2web (Unix)	The AFP2web program
	quickstart.txt	Quick start instructions on using demo.bat (Windows), or demo (for UNIX)

3.1.4.3 AFP2web C SDK product

The following files are installed for AFP2web C SDK product:

Directory	Files	Description
(target directory)	a2wcsdk.exe (Windows) a2wcsdk (Unix)	The AFP2web C SDK sample program
doc\csdk	index.html, *.*	Documentation for AFP2web C SDK
liba2w\lib	*.h	C SDK API header files
	*.lib (Windows) *.a (UNIX)	C SDK static library files
liba2w\samples\src	*.c, *.h	C sample source files to demonstrate various AFP2web C SDK APIs
liba2w\samples\a2wCSDK	a2wCSDK.dsw (Windows) a2wCSDK*a2wCSDK.dsp (Windows)	Microsoft VC++ 6.0 project workspace for sample source files
	build (UNIX) clean (UNIX) configure.in (UNIX) Makefile.al (UNIX)	Shell script and build tool files to compile sample source files

3.1.4.4 AFP2web Java SDK product

The following files are installed for AFP2web Java SDK product:

Directory	Files	Description
(target directory)	a2wjdk<version>.dll (Windows) liba2wjdk<version>.so (Unix)	The AFP2web Java SDK native program
	a2wjdk<version>.jar	Java SDK jar file
	addUserData.pm	AFP2web Scripting Facility script sample to demonstrate adding user data through Scripting Facility and to access them in Java
doc\jsdk	index.html, *.*	Documentation for AFP2web Java SDK
javasamples	*.java	Java sample source files to demonstrate various AFP2web Java SDK APIs

https://info.oxseed.com/a2w_uguide-en-40/User_Guide/1494-mht_DSY/g1/1595-mht_DSY.html

3.1.5 Removing AFP2web 4.x

To remove AFP2web from your system, delete all files and directories created by the installation routine of AFP2web.

3.2 Configuring AFP2web 4.x

3.2.1 Overview

Configuring AFP2web involves setting parameters, providing AFP resources and specifying search path for shared Objects used by AFPweb.

- Setting Parameters in the INI File
- Converting Sample Documents and Fine Tuning the Configuration
- Providing AFP Resources
- Specify Search Path for Shared Objects

3.2.2 Setting Parameters in the INI File

The configuration file (INI file) defines global parameters of AFP2web, such as conversion options, output paths, paths to AFP resources, and other processing options. For a complete description, please refer to [INI Parameter Reference](#).

By default, AFP2web searches for an INI file with the name `afp2web.ini` in the current working directory. You can access a particular INI file with one of the following command line options:

-if	Path and filename of the INI file
-ip	Path to <code>afp2web.ini</code>

You can override most of the parameters by entering the appropriate command line option when starting AFP2web. Command line options are described in detail [AFP2web Command Line Options](#).

3.2.3 Converting Sample Documents and Fine Tuning the Configuration

Run-testing AFP2web by converting sample documents is the first step in the process of fine tuning the AFP2web configuration. The conversion results and the messages in the LOG file indicate what remains to be done.

In the following description, we give a few hints on what to take care of.

3.2.3.1 Download AFP Spool File (Host Documents) in Binary Format

Please take care to download any AFP spool file from the host in binary format. (A common mistake is to download AFP documents and resources from the host to the PC and, when doing this, to perform an ASCII conversion.)

3.2.3.2 Convert the AFP Spool File and Check the LOG File

Run the afp2web.exe program to convert your sample documents in the AFP spool file.

To quickly find out how to start AFP2web, you can follow the instructions and use the demo batch command file as described under [Quick Start](#).

AFP2web creates the following files as output:

- The converted spool file, documents, or a text file with error messages.
- A log file, if logging is activated.

When you start AFP2web, activate the logging option. Logging will slow down AFP2web, but is necessary at this stage. For example, to investigate font processing, use the INI parameter `LoggingLevel=FONT` (or the command line option `-lf`).

Check the converted document and any system messages to identify problems. For example, a missing resource or font might be the reason for a non-satisfactory result.

Examples include:

- A resource is missing. In this case, please check if the resource is located at the position specified in the INI file (or as specified in the command line options when calling AFP2web).
- If font substitution/referencing is used, the log file might indicate that a substitution font is missing. Check to see if the font is correctly specified in the mapping.def. In addition, this font must be either installed (for output to a raster format, such as TIFF) or must be found in the specified folders (for PDF). For more information, please refer to [Using the Log File](#) or to [Font Logging](#).

3.2.4 Providing AFP Resources

AFP resources (such as Fonts, Formdefs, Overlays, Page Segments) can be defined in-line in the AFP data stream. If not, then AFP2web searches for an external resource in the folders you have specified.

The INI parameter `CodePage` (or command line option `-dcp`) defines the default code page to use for translating resource names, NOPs and indexes of the AFP spool to ASCII.

Normally, no special configuration is required for AFP2web's font handling. If the AFP fonts are available either in-line or as external resources, AFP2web will take the actual fonts and either embed them (in the PDF output) or rasterize them (for output to a raster format such as TIFF). Thus, the actual font is used and the conversion results reproduce the original documents in true fidelity.

For more details about using fonts, associated code pages, and the configuration files, please refer to [Handling Fonts](#).

3.2.4.1 Path to External AFP Resources

There are different possibilities of providing external AFP resources. For example, you enter the command line option `-rf` when calling the program to use a particular resource file, or you use the settings of the INI file to specify paths for specific resource types.

For all parameters except for ResPath (`-rp`) and the option `-rf`, you can specify a comma-separated list of paths. The parameters, which you can use, are:

INI file	Command line option	Type of AFP resource
ResPath	<code>-rp</code>	All resource types
	<code>-rf</code>	All resource types in a particular resource file
AFPFontPath	<code>-ap</code>	AFP font resources
OverlayPath	<code>-ovp</code>	AFP Overlay resources
PageSegmentPath	<code>-psp</code>	Page Segments
FormDefPath	<code>-fdp</code>	Formdefs

The Standard Search Path Used to Find a Resource

AFP2web searches for external AFP resources in the following order:

1. The folder specified for the resource type (default is `samples/resource`)
2. Folder specified in ResPath (default is `samples/resource`)
3. Folder of the input files
4. In the default folder "`samples/resource`".

The Search Path for Non-AFP Resources

When AFP2web searches for non-AFP resources, for example, for TIFF/JPEG files requested in IOBs, AFP2web looks in the following paths in this order:

1. Resource path (given by ResPath INI attribute)
2. Input file path
3. Current path

Optional Feature: File Extensions for AFP Resources

It is possible to use the INI file to specify file extensions for the following resource types:

FormdefExt	File extension for files with Formdefs
OverlayExt	File extension for Overlays
PageSegExt	File extension for Page Segments
CharSetExt	File extension for Character Sets
CodePageExt	File extension for Code Pages
CodedFontExt	File extension for Coded Fonts

The Processing Flow of AFP2web When Searching for a Resource

AFP2web doesn't need an extension to the resource name. AFP2web searches for resources as follows:

1. AFP2web follows the INI file setting (default or the user-specified extension) to search for the resource. AFP2web goes through the standard search path to find the resource.
2. If the resource cannot be found in step 1 and the option Strict is on, AFP2web will stop the process.
3. If the resource is not found in step 1 and the option Strict is off, AFP2web will search for the required resource again in the standard search path. AFP2web takes the first resource found with any type of extension.

Important: AFP2web searches for resources, with file name beginnings that match the name of the resource. The first file found is taken. This might not be the resource you want.

Examples:

C1H20000 is required. Per default A2W will look for "C1H20000*". All the following resources will be valid: C1H20000, C1H20000.chs, C1H20000.240, C1H20000_ABC.def.

If more than one resource matches the search pattern, A2W will take the first one found.

If the INI file specifies "chs" as the valid file type extension, then AFP2web will find the resource in step 1: "C1H20000.chs".

If the INI file has "*" or no settings, then the default applies and AFP2web will find the resource in step 3: "C1H20000".

Specifying FORMDEFs

As mentioned at the beginning of this section, AFP resources (such as Formdefs, Overlays, Page Segments) can be defined in-line in the AFP data stream. If not, then AFP2web searches for an external resource in the folders you have specified. For example, you can use the command line option -fd or -rf to reference a resource file containing a Formdef.

A Formdef is a special AFP resource that contains medium maps and/or copy groups, but which is not referenced in the AFP data stream. The [FORMDEF] section of the INI file can be used to map Medium Maps to Formdef resource names. This feature is useful, for example, if the command line option -fd is not used. The Formdef itself must be available as external resource.

3.2.5 Specifying the Search Path for Shared Objects

AFP2web Linux/Aix 64 bit versions, need following dynamic libraries as run time dependency

- Perl (libperl.so.5.x)
- Standard C++ (libstdc++.so.x)

These libraries are given as part of AFP2web Release package. In order that AFP2web find these libraries during run time, current directory must be added to the system load library path as given below

3.2.5.1 Linux:

```
#---- Export the Library Path
export LD_LIBRARY_PATH=.:LD_LIBRARY_PATH
```

3.2.5.2 Aix:

```
#---- Export the Library Path
export LIBPATH=.:LIBPATH
```

3.3 Using AFP2web 4.x

3.3.1 Quick Start

To convert AFP documents with AFP2web:

1. Make sure that you download your AFP documents and resources in binary format from the host.
2. Copy your AFP spool file to the directory **samples/**.
3. Copy your AFP resources to the directory **samples/resource/**.
4. Run AFP2web "**demo.bat**" (Windows) or "**./demo**" (Unix) respectively
5. Specify input sample file name when prompted "ENTER SAMPLE FILENAME [<default sample filename>]:"
NOTE: If not specified, the default sample file name within "[...]" will be used
6. Check the output in the directory **pdf/**.
7. Repeat steps 1 to 6 for each and every AFP spool file.

If you encounter any problems, please contact support@oxseed.com.

3.3.2 Starting/Stopping AFP2web From the Command Line

3.3.2.1 Starting AFP2web

You start the program `afp2web` as a console application with the following command:

On Windows: `afp2web.exe [-options ...] inputfile(s)`

On UNIX: `./afp2web [-options ...] inputfile(s)`

Wild cards are allowed for input file(s), for example: `"*.afp"` for all files having "afp" as extension.

For a detailed description of the command line options you can enter, please refer to [AFP2web Command Line Options](#).

3.3.2.2 Stopping AFP2web

On Windows: To stop AFP2web press CTRL+C on the system console or stop the task in the Windows Task Manager.

On UNIX: Kill the process.

3.3.3 Controlling the Conversion Process

In this section, we list the parameters of the INI file, which you use to control the formats and the types of output documents to create.

The parameter **InputFormat** specifies the format of the input documents to convert. This parameter can have one of the following values:

TIFFIN	TIFF document
LPDIN	LPD format
MMDIN	MMD format

In general, AFP2web can identify input formats automatically, to overwrite this behavior in above options can be used.

The parameter **OutputFormat** specifies the target format for the conversion. This can be one of the following:

AFP	AFP format
ASCII	ASCII format with adjustments to line/column positioning
JPEG	JPEG format
PDF	PDF format
PNG	Portable Network Graphic
TIFF	TIFF format

The parameter **FileCreationMode** controls splitting or merging documents, creating index data or activating the Scripting Facility. One of the following options is possible:

ALL	One output file for each input file.
DOC	Multiple output files: One file per AFP document.
DOC_MERGE	Merge all files into one output file. For converting to a raster format, such as TIFF, to PDF or for converting AFP.

DOC_INDEX	Create one index file and one output document per indexing information unit.
DOC_COLD	Split documents and process document elements using the AFP2web Scripting Facility.
PAGE	One file per page in an AFP document.

3.3.4 Selecting Documents for Conversion

Processing AFP spool files normally involves converting individual AFP documents and processing AFP index data. AFP document boundaries are detected through document index elements. Or, when using the Scripting Facility, according to the rules defined in a script. To let AFP2web process standard index elements, specify **DOC_INDEX** (See also [Processing AFP Indexes](#)). To process index data and document splitting with the Scripting Facility, specify **DOC_COLD** (See also [Basic Concepts of the AFP2web Scripting Facility](#)).

You can instruct AFP2web to either start or end processing the spool file at a particular AFP document. AFP2web will parse the complete spool file to find the specified document. This option is used to handle special cases, such as emergency restarts or for debugging.

Using the option **-p** creates processing log which we call a „protocol“. This protocol is a useful reference for selecting documents for the conversion. In the protocol, you will find a list of the documents processed and their document IDs. The protocol is written to the folder for output files (INI file parameter **OutputFilePath**, or the **-op** command line option). Filename of the protocol is: „protocol“ + timestamp + „.txt“.

3.3.4.1 Example of a Protocol Showing Document IDs

Running AFP2web v4.0 [Built for Windows 2003/2008 32-bit on Oct 1 2010 at 11:43:13] at 10:17:22 on 14 October 2010 using : -q -c -p -doc_index samples\insure.afp

```
Processing Spool File: samples/insure.afp
[12:13:51] Processing Document Id : 1, ./pdf/insure_s3j8.1.pdf : Ok
[12:13:52] Processing Document Id : 2, ./pdf/insure_s3j8.2.pdf : Ok
[12:13:52] Processing Document Id : 3, ./pdf/insure_s3j8.3.pdf : Ok
[12:13:52] Processing Document Id : 4, ./pdf/insure_s3j8.4.pdf : Ok
[12:13:52] Processing Document Id : 5, ./pdf/insure_s3j8.5.pdf : Ok
```

AFP2web v3.4 [Built for Windows 2003/2008 32-bit on Oct 1 2010 at 11:43:13] ended at 10:17:23 on 14 October 2010, Elapsed Time 00:00:01

A case for using this option: When AFP2web stops processing due to a failure (for example memory or file space exhausted) you can refer to the protocol to find out where AFP2web stopped.

After solving the problem you do not have to process the complete spool file again. Instead you use the parameters **StartingDocument** (or the option **-sd**) and **EndingDocument** (or the option **-ed**) to select a range of documents for processing. For example, to start converting at particular point in the spool file, you start AFP2web with the option **-sd:docid**, where docid is the running document number which you can find as document ID in the „protocol“.

Parameters for processing spool files:

INI file	Command line option
----------	---------------------

FileCreationMode=DOC_INDEX or FileCreationMode=DOC_COLD	-DOC_INDEX or -DOC_COLD
Protocol=on	-p
StartingDocument	-sd
EndingDocument	-ed

3.3.5 Selecting Pages for Conversion

You can select a range of pages within the AFP documents to be processed. The parameters to use:

INI file	Command line option
StartingPage EndingPage	-pp:[fp][-tp] (Process from/to pages)

3.3.6 Processing AFP Indexes

In the following, we describe the standard processing of AFP index data with AFP2web. If you want to customize AFP2web standard processing, please refer to the description of the Scripting Facility.

To invoke the AFP2web standard index processing, you use the option **FileCreationMode=DOC_INDEX**.

By default, AFP2web reads standard AFP index elements and writes the index data either to the path for output documents as specified by the parameter **OutputFilePath** or to the path specified by **IndexPath**.

If the AFP index is not within the AFP data stream, but instead delivered as an external file, you use the command line option **-xf** to specify the external file.

With the parameter **IndexFormat**, you create index files formatted either as XML or as comma-separated values (CSV).

The following shows an example of an index file in XML format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Index SYSTEM "Index.dtd">
<Index Doctype="PDF" DocName="pdf/xml/INSURE_s3k.0.pdf" PageCount="3" Size="16726">
  <PageGroupIndex PageGroup="00000001">
    <Data>
      <Name>Insured</Name>
      <Value>Geoffrey R Stephens</Value>
    </Data>
    <Data>
      <Name>Policy</Name>
      <Value>324-1443255-11</Value>
    </Data>
  </PageGroupIndex>
  <PageIndex Page="00000001">
    <Data>
      <Name>Contents</Name>
      <Value>Disability Income Policy</Value>
    </Data>
  </PageIndex>(additional lines not shown here...)<PageIndex Page="00000003">
    <Data>
      <Name>Contents</Name>
      <Value>Definitions</Value>
    </Data>
```

```
</PageIndex>
</Index>
```

If you select CSV as the format for index files, you can use the `IndexRecord` parameter to add meta information to each index entry.

For example, `IndexRecord=PAGE_GROUP_NAME, PAGE_NAME,PAGE_COUNT,FILE_NAME,FILE_TYPE,INDEX,FILE_SIZE` will add these fields as meta information to each index entry in this order.

The following figure shows an example of index file in CSV format:

```
PAGE_GROUP_NAME,PAGE_NAME,PAGE_COUNT,FILE_NAME,FILE_TYPE,INDEX,FILE_SIZE
324-1443255-1100000001,,3,./pdf/insure_sluo.1.pdf,PDF,Insured=Geoffrey R Stephens,112776
324-1443255-1100000001,,3,./pdf/insure_sluo.1.pdf,PDF,Policy=324-1443255-11,112776
,00000001,3,./pdf/insure_sluo.1.pdf,PDF,Contents=Disability Income Policy,112776
,00000002,3,./pdf/insure_sluo.1.pdf,PDF,Contents=Policy Schedule,112776
,00000002,3,./pdf/insure_sluo.1.pdf,PDF,Contents=Policy Schedule,112776
,00000002,3,./pdf/insure_sluo.1.pdf,PDF,Contents=Table of Benefits,112776
,00000002,3,./pdf/insure_sluo.1.pdf,PDF,Contents=Additional Benefits,112776
,00000002,3,./pdf/insure_sluo.1.pdf,PDF,Contents=Premium Summary,112776
,00000003,3,./pdf/insure_sluo.1.pdf,PDF,Contents=Definitions,112776
```

The parameters used for creating index files:

INI file	Command line option
InputFormat=AFP	
OutputFormat=PDF	-PDF
FileCreationMode=DOC_INDEX	-DOC_INDEX
IndexPath	-xp
	-xf
IndexFormat=CSV XML	
IndexRecord=fieldlist	

3.3.7 Storing Output Documents in Sub Folders

The number of resulting output documents might be too large for your file system to handle efficiently when all output documents are stored into one folder (directory). For this case, AFP2web provides the command line option **-dc** (resp. the INI parameter **DocumentCount**), which you can use together with the option **DOC_COLD** or **DOC_INDEX**. You use the command line option **-dc** to create new sub folders for your output documents and to limit the number of documents per sub folder.

Example: The command line option **-dc:500** together with **-doc_cold:xml** limits the number of output files to 1000 per sub folder: You receive per sub folder 500 documents plus 500 index files in XML format (This is the conversion results of 500 documents).

AFP2web neither prevents overwriting documents, which already exist, nor does AFP2web check the number of documents, which already exist in the output folder.

We therefore recommend:

- Please use AFP2web with the option **-dc** only to process one single large input spool file.
- Use **-dc** only to create new sub folders for your output. Do not use **-dc** to produce output in existing folders.

Example of output for the option -dc:500 and -doc_index:

```
Folder ... \pdf
<DIR> 00
<DIR> 01
Subfolder ... \pdf\01
insure_s1eo.1.pdf
insure_s1eo.1.xml
insure_s1eo.2.pdf
insure_s1eo.2.xml
insure_s1eo.3.pdf
insure_s1eo.3.xml
...
insure_s1eo.500.pdf
insure_s1eo.500.xml
Subfolder ... \pdf\01
insure_s1eo.501.pdf
insure_s1eo.501.xml
...
```

3.3.8 Processing Included Objects (Included Images)

Non-AFP data (mostly images such as logos and colored images) are either embedded in or referenced by AFP object containers, called Included Objects.

AFP2web strives to optimize conversion results and the output size of the object:

- Color objects are stored in JPEG format (JPEG is a lossy format). You use the INI parameter JPEGQuality (or the command line option -jq) to increase or decrease the compression.
- Black/white objects are stored TIFF G4 format.

3.3.8.1 How AFP2web Finds the Object Data?

The object data can be either in-line or in external files. To load and process an external object file, AFP2web must know its file name and file type. AFP2web derives this information from the "Include Object - IOB" structured field of AFP data stream.

Supported Object Data Types:

The object types supported formats and the file extensions AFP2web expects are,

Object Type (Supported Formats)	File Extension
TIFF (G4, G3, LZW, Packbits, Uncompressed, JPG, CCITTRLE, CCITTRLEW, IT8CTPAD, IT8LW, IT8MP, IT8BL, PIXARFILM, PIXARLOG, DEFLATE, ADOBE_DEFLATE, THUNDERSCAN, DCS, NEXT, SGILOG, SGILOG24)	tif
JPEG	jpg
Portable Network Graphics	png

For an object container referencing an external object, the file name is a combination of: <Object Name in SFI>.<File Extension>

Example: If the AFP stream has an included Object MOC1 of type TIFF, AFP2web searches for the file name: MOC1.tif

After building the object container file name, AFP2web searches for external AFP resources in the following order:

1. Folder specified in ResPath (default is samples/resource)
2. Folder of the input files
3. In the current path.

3.3.8.2 Optimizing the Conversion

By default, AFP2web loads and processes the object data in memory. If the AFP spool file contains especially large images, which are only occasionally used, there is no reason for holding this object data in memory.

For this case, AFP2web offers the INI file parameter **SaveOCDataOnDisk** (respectively the command line option **-sod**). This option specifies that an included object (the graphic file) be temporarily stored on disk instead of being loaded and processed in memory.

The temporary files for object container data are stored in the path specified by **TempPath** (the default path is the temporary path defined in the system, the corresponding command line option is **-tp**). AFP2web removes these temporary files when cleaning up.

3.3.9 Notes on Barcode Support

AFP2web supports following barcode types

- Code 128 barcode (Character Set B of Code 128)
- Code 39
- Bi-dimensional Data Matrix 2D

Bi-dimensional Data Matrix 2D barcode type, supported with following constraints

1. All square sizes having one data region are supported. Square sizes having more than one data region are not supported.
2. All rectangular symbol sizes are supported irrespective of the number of data regions. The following table gives more details about the support for the DataMatrix 2D barcode. The number in parantheses stands for the number of data regions for that size.

1a. Supported rectangular symbol sizes	8x18 (1) 8x32 (2) 12x26 (1) 12x36 (2) 16x36 (2) 16x48 (2)
2a. Supported square sizes (These sizes have one data region.)	10x10 12x12 14x14 16x16 18x18 20x20 22x22

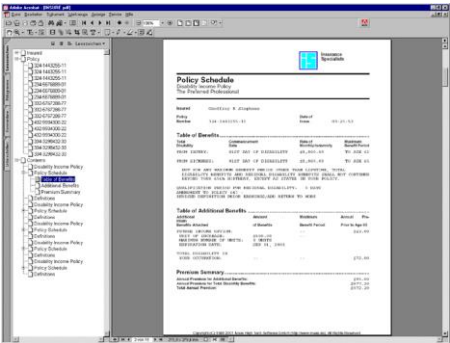
	24x24 26x26
2b. Square sizes not supported (These sizes have more than one data region.)	32x32 (4) 36x36 (4) 40x40 (4) 44x44 (4) 48x48 (4) 52x52 (4) 64x64 (16) 72x72 (16) 80x80 (16) 88x88 (16) 96x96 (16) 104x104 (16) 120x120 (32) 132x132 (32) 144x144 (32)

3.3.10 Creating PDF Bookmarks for Index Elements

Using PDFBookmark (or by using the command line option `-bm`), you create PDF bookmarks for index elements in the resulting PDF files.

The following figure shows an example of the bookmarks as they are displayed in the Acrobat Reader.

Figure 4 PDF Bookmarks for Index Elements



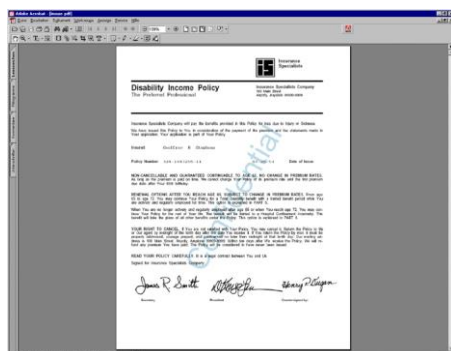
The parameters for creating PDF bookmarks:

INI file	Command line option
InputFormat=AFP	
OutputFormat=PDF	-PDF
PDFBookmark	-bm

3.3.11 Adding Text as Watermark Behind the Document Text

Using the INI file parameter **Watermark** or the command line option **-wm**, you can embed a text as watermark behind the document text, as shown in the following example:

Figure 5 PDF with watermark



Text Added as Watermark on a Document Page

3.3.12 Stopping Conversion When a Required Resource Is Missing

Using the command line option **-Strict**, AFP2web will stop processing if a required AFP resource cannot be found. The resources AFP2web checks for included Page Segments, Overlays, and Formdefs, AFP font resources and non-AFP resources. This function is useful if you cannot allow such failures during conversion.

The parameter:

INI file	Command line option
Strict=on	-strict

3.3.13 Using the Log File

3.3.13.1 Creating the Log File

Name of the log file

If logging is activated, AFP2web creates a log file for each AFP file. The name of log file is the filename and extension of input file with the additional extension ".log" added to it. For example, the input file **INSURE.AFP** will be converted to **INSURE.PDF** with the respective log file **INSURE.AFP.log**.

The log file path

Use the parameter **LogPath** in the INI file to set the target folder for log files. You override the INI file if you use the command line option **-lp** when calling the program **afp2web**.

Using the log file for font information: We recommend using the log file to investigate font information. You do this either:

- Globally in the INI file with the parameter **LoggingFont=on**,
- or once when calling the program using the command line option **-lf**.

Additional options

You can add more information to the log file, which is helpful for the AFP expert and AFP2web support incidents. To add more information to the log file, use:

- The afp2web.ini parameter **Logging=on**, or **LoggingLevel**, or **ExceptionLoggingLevel**
- The command line option **-I**, or **-II**.

3.3.13.2 Setting the ExceptionLoggingLevel

By default, AFP2web will issue exception messages with no further details. You can add more information to exception messages with the parameter **ExceptionLoggingLevel**.

3.3.13.3 Structure of the Log File

The LOG file has two parts:

Parsing section	showing the AFP parameters found while parsing the AFP data stream. Here you might find messages indicating whether resource files were found or not. The parsing section ends with an end-of-file message. This part is intended for, and useful for the AFP expert and AFP2web Support.
Builder section	which traces font substitution. If you want to use the log file, we recommend that you use the builder section to identify problems with font substitution.

The following table indicates which information is displayed in the log file:

Parsing-Section Record Structure	
Column 1 - 10:	File offset of the Structured Field Introducer (SFI) in Hex format. The SFI can be taken as the MO:DCA-command.
Column 14 – 21:	The SFI in Hex format.
After Column 23:	The parameters found in the SFI.

Builder-Section Record Structure	
PDF page number	Example: Writing Page 1
Font (Character Set, Code Page)	Example: ==>CS=C1H400B0, TF=HELVETICA LATIN1, W=B, S=12.00, RF, CP=T1GI0361(MD:2065.cp)
Font used and the text with text position	Example: -->UsedFont=C1H400B0, TF=F1, S=12.00, T3

	@ (8496,862)>Insurance< @ (8496,1120)>Specialists<
--	---

4 Handling Fonts

In this chapter, we describe each font processing mode and show a few examples. But before we do this, we give a short introduction to the configuration files which are used to control font processing.

For the sake of simplicity, we say input fonts when we talk about fonts found in the input documents, and output fonts to refer to the fonts used by the output documents.

Fonts are either embedded or available in line in the documents or are referenced as external fonts. Fonts, which are used, must be available to AFP2web.

4.1 The Configuration Files and the Defaults Used for Font Processing

The configuration files used for fonts are:

- the INI file (afp2web.ini)
- the font mapping file (mapping.def)
- code page files (CP files)

If AFP2web cannot find a specified font, then AFP2web will use a fallback font to produce conversion results.

4.1.1 The INI File (afp2web.ini)

The INI file is the configuration file for AFP2web processing. If a command line option is entered, it will replace the corresponding INI parameter.

You can find a detailed description of each parameter in the INI file in [Overview of INI Parameters and Command Line Options](#) and [INI Parameter Reference](#).

The following table lists the INI parameters which are relevant to font processing.

INI Parameter	Command line option
Parameters to print a log with font information:	
LogLevel=FONT	-ll:FONT or -lf
LogLevel=ALL	-ll:ALL
LogLevel=FNTR	-ll:FNTR
Processing flag used to stop processing when resources are missing:	
Strict=on	-strict

Parameters used to specify locations of AFP font resources and substitution fonts:	
AFPFontPath=<path>	-ap
<named sections for font locations>	
ResPath=<path>	-rp
ExtFontPath= <path>	-fp
Parameters used to specify naming conventions for AFP resource members:	
CharSetExt=<file extension>	
CodedFontExt=<file extension>	
CodePageExt=<file extension>	
Character mappings for code pages:	
CodePage= <defaultcodepage>	-dcp
CpPath=<path>	-cp
CodePageDefaultChar = " "	

4.1.2 The Font Mapping File (mapping.def)

The mapping.def is the font processing configuration file.

AFP2web offers a standard processing, which uses the original fonts whenever possible. It is therefore basically possible for you to run AFP2web without modifying the mapping.def. You normally use the mapping.def to fine-tune font processing for your type of documents.

You use the mapping.def to:

- define the font processing mode for each character set or font.
- specify substitution fonts for a target format such as AFP, PDF, a raster format such as TIFF on either Windows or Unix systems.
- map character sets to code page files.
- specify file name suffixes to help AFP2web find the font files. For example, if you use the font "ARIAL, Italic" and your font file is named Arial-Italic.ttf, then you can specify that italic fonts use font file names with the suffix "-Italic".

- describe AFP fonts (character sets, code pages, font global IDs). For example, character sets and encoding in the AFP data stream might be out-dated, or no longer visible in your current AFP resource pool. In this case, you can define AFP font attributes manually in the AFP-specific sections of the mapping.def.

For detailed information on the syntax and parameters of this configuration file, please refer to [The Font Mapping File \(mapping.def\)](#).

4.1.3 Code Page Files

Code page files (CP files) define the font encoding to use. They map EBCDIC to ASCII and Unicode code points. ASCII is the encoding, which the AFP2web Scripting Facility uses to identify objects in the AFP data stream.

AFP2web delivers a set of predefined code page files, which are commonly used and which can be found in the directory of the mapping.def file.

For more details about code page files, please refer to [Code Page Files \(CP Files\)](#).

4.1.4 Default Assumptions If Mappings or Font Resources are Missing

If neither the input files nor any of the above configuration files deliver the font resources and mapping information, AFP2web adheres to a default processing.

Please note that in this mode, AFP2web cannot guarantee true fidelity of the conversion results.

For example, font characteristics can be derived from the naming conventions for these resources, if these follow the IBM font naming convention. Thus, AFP2web will still be able to map AFP fonts to substitution fonts. The rules for this automatic font mapping is described in detail in [Identifying Fonts Using the IBM Naming Convention](#).

4.2 AFP2web Font Processing Modes

AFP2web offers the the following font processing modes:

- Use of the original fonts (the default processing)
- Font substitution and referencing
- Font substitution and embedding

4.2.1 Use of the Original Fonts (Default)

Normally, no special configuration is required for AFP2web's font handling. If the font resources are available either in-line or as external resources, the default behavior of AFP2web is to take the original fonts to produce the conversion results.

4.2.1.1 The Default Conversion of AFP Documents

General Information

When converting AFP documents, AFP2web uses the original fonts:

- For PDF output, AFP2web converts AFP fonts as given below
An AFP outline font is converted to a Type 1 font.
An AFP raster font is converted to a Type 3 font.
In this mode, AFP2web always embeds the fonts in PDF.
- For RASTER outputs (like TIFF, JPEG, PNG etc), AFP rasterizes both AFP outline and raster fonts

If the AFP fonts are not provided inline in the AFP spool, they must be available as external font resources.

AFP2web currently does not support AFP Outline Fonts of type "CID-Keyed".

Example: AFP2PDF With Default Font Processing(Font Rasterizing)

The input format AFP is to be converted to PDF. The original font of the AFP Character Set C1D0GT10 is to be directly converted and embedded in the PDF document. The configuration file afpcp\mapping.def specifies that all fonts be rasterized.

This is the default processing of AFP2web:

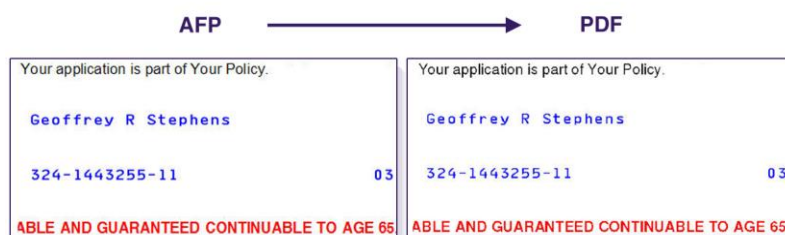
```
[CHARSET RENDERING]
*=0
```

The following command starts the conversion:

```
afp2web.exe -lf -lp:pdf -c -q samples\insure.afp
```

The conversion result is a PDF document. Our example embeds fonts and therefore exceeds 100 kB in file size.

Figure 6 fonts example



The information in the LOG-file:

```
==>CS=C1D0GT10, FGID=40, TF=GOTHIC, W=M, S=10.00, RF, CP=T1D0BASE
-->UsedFont=C1D0GT10, TF=F7, W=M, S=10.00, RF, ACP=MD:2063.cp, ENC=T1D0BASE
  @(2448.00,5353.00)>Geoffrey R Stephens<
```

LOG Text	Description
GOTHIC	The name of the font according to the AFP definition is "GOTHIC"
C1D0GT10	The input font in AFP, the name of the AFP Character Set. AFP2web uses the original AFP font for the font rasterizing.
F7	For the PDF document, a PC font is generated and embedded with the name F7. In the PDF document, it can be identified as a Type 3 font.
W=M	Weight is "Medium"
RF	Raster Font. The font has been embedded into the PDF document as a raster font.

4.2.1.2 The Default Conversion of PDF Documents

General Information

When converting PDF documents to AFP, the standard processing of AFP2web is:

- The fonts in PDF are rasterized and inserted as IOCA images in the target AFP document.
- A Type 3 font in the PDF file will be converted to an AFP raster font and will be delivered as a font resource in the conversion results.

When rasterizing fonts from PDF to AFP, the following applies:

- Embedded input fonts are used as is.
- If input fonts are not embedded, they must be available as external fonts. AFP2web searches for external fonts, as described below.
 - Using defined mapping of PDF font typeface name to external fonts in [INPUT FONT ALIASES] section.
 - Finding PDF font typeface name.
 - Run AFP2web with the PDF input sample
 - `afp2web.exe -q -c -afp -lf -lp:pdf samples\insure.pdf`
 - PDF font typeface name is logged as typeface name(TF) in AFP2web log file as shown below. Use this name to map external fonts in [INPUT FONT ALIASES] section.
 - Writing Page 1
==>TF=FreeMono, W=M, S=10.00, Type1, ENC=WinAnsi, Emb
-->CS=CZ4200, TF=FreeMono, W=M, S=10.00, OF, CP=T1001141, ACP=MD:1141.cp
@(510,1115)>Geoffrey R Stephens<
 - If there is no mapping entry, AFP2web searches for the full name of the font. For example: If PDF font name is "Arial", then AFP2web will search for a PC font "Arial.*" which will match with the file name "Arial.pfb" or "Arial.ttf"
 - If a font name has special character like ',' then the font file name must also include this character. For example, if the the font name is "Arial,Bold" then font file name should be "Arial,Bold.ttf"
 - To identify a font subset, AFP2web ignores the first 7 characters of the font name. For example: If PDF font name is "NHAHHE+Arial", then AFP2web will search for "Arial.*" which will match "Arial.pfb" or "Arial.ttf".

When PDF input has a font with custom encoding, not all text objects using that font will be rasterized. Only text objects that have custom encoded code points will be rasterized.

Example: PDF2AFP With Default Font Processing(Font Rasterizing)

We want to convert PDF to AFP and rely on the default processing. Default processing is to rasterize PDF fonts and inserted these as images into the resulting AFP document. No AFP font resources are produced.

This processing is equivalent to the settings in the mapping.def:

```
[CHARSET RENDERING]
*=0,1
```

The following command in the demo.bat starts the conversion:

```
afp2web.exe -q -c -afp -lf -lp:pdf samples\insure.pdf
```

The following LOG-file information shows that an embedded font is rasterized:

```
==>TF=FreeMono, W=M, S=10.00, Type1, ENC=WinAnsi, Emb
-->UsedFont=C:/Users/john/AppData/Local/Temp/FreeMono_27.fnt, TF=FreeMono, W=M, S=10.00, RI
  @(510,1115)>Geoffrey R Stephens<
```

LOG Text	Description
FreeMono	The type1 font in the PDF file
UsedFont	Indicates that the font is unpacked to a temporary folder
RI	Font is inserted as Raster Image

4.2.2 Font Substitution and Referencing

In cases, where it is not preferable to use the original font, you can specify that a substitution font replace the original font. Substitution fonts might be available and produce satisfactory results. You use font referencing to reduce the file size of the output document.

4.2.2.1 Converting AFP Documents with Substitution Fonts

General Information

You use the mapping.def to specify that an AFP font be replaced by a PC font:

- The section [CHARSET RENDERING] specifies the processing mode for each AFP Character Set.
- For the conversion to PDF, the section [PDFFONT] maps AFP Character Sets to the substitution font.
- For the conversion to a raster format (such as TIFF, JPEG), the section for the system platform, either [WINFONT] or [UNIXFONT], maps AFP Character Sets to substitution fonts.

If the AFP fonts are not defined in-line in the input spool file, you must make the external font resources available to AFP2web.

Additional entries in the mapping.def determine which substitution font to use:

1. The AFP data stream delivers the base name for the font.
2. The mapping.def maps the font name to the typeface name.
3. The mapping.def section [FONTSUFFIX] determines the suffixes to add for the font attributes (bold, italic).
4. The font attributes are either taken from the AFP font resource / definition or from the settings in mapping.def section [FGID].

AFP2web supports the following formats for substitution fonts:

- TrueType,
- OpenType (ONLY as Single True Type Font with the file extension ".ttf"),
- Type 1

Example: AFP2PDF With Font Substitution and Referencing

The input format is AFP and shall be converted to PDF.

The configuration file `afpcp\mapping.def` specifies that all fonts be replaced and referenced. If AFP2web cannot find the PC font in the system, then AFP2web will use a default substitution font. It is very important that the PC fonts are available to AFP2web.

The font processing mode in the `mapping.def` (replace and reference):

```
[CHARSET RENDERING]
*=1
```

The following command starts the conversion:

```
afp2web.exe -lf -lp:pdf -c -q samples\insure.afp
```

The result is a PDF document, which is significantly reduced in file size, because fonts are only referenced. Our example is just about 45 kB in file size.

The information in the LOG-file shows that HELVETICA LATIN1 has been mapped to the PC font Helvetica-Bold:

```
==>CS=C1H400B0, FGID=2305, TF=HELVETICA LATIN1, W=B, S=12.00, RF,
CP=T1GI0361
-->UsedFont=Helvetica-Bold, TF=Helvetica-Bold, W=B, S=12.00, SF,
ACP=MD:2065.cp, ENC=ANSI
@(8496,862)>Insurance<
@(8496,1120)>Specialists<
```

LOG Text	Description
C1H400B0	AFP character set with the typeface name HELVETICA LATIN1
UsedFont	The font used is Helvetica-Bold
SF	Standard font

The following information in the LOG-file shows that AFP2web could not find a PC font file in the system. A default font must therefore be used. A warning is issued and the default font used is indicated by the code "D":

```
==>CS=C1D0GT10 TF=GOTHIC Warning! External font file not found.
==>CS=C1D0GT10, FGID=40, TF=GOTHIC, W=M, S=10.00, RF, CP=T1D0BASE
-->UsedFont=Helvetica, TF=Helvetica(D), W=M, S=10.00, SF,
ACP=MD:2063.cp, ENC=ANSI
@(2448,5353)>Geoffrey R Stephens<
```

LOG Text	Description
Warning!	For the character set C1D0GT10 (Typeface GOTHIC), an external font file could not be found.
UsedFont	The font used is Helvetica
D	Default font

When using substitution fonts, it is advisable to check the conversion results. The original text string "Geoffrey R Stephens", for example, uses a mono-spaced font in AFP:

Figure 7 AFP font, monospaced

Geoffrey R Stephens

The conversion result in PDF uses a proportional font because the GOTHIC typeface has not been mapped in the mapping.def:

Figure 8 Mapped to Gothic in PDF

Geoffrey R Stephens

We see that a conversion result might not be acceptable in all cases and that adjustments might be necessary.

Example: AFP2PDF With Specifying Substitution Font for specific AFP Fonts

The input format AFP is to be converted to PDF. If AFP2web cannot find a PC font in the system, AFP2web then uses a default. This is what we want to prevent.

The following definition in the configuration file afpcp\mapping.def specifies that the AFP Character Set C1D0GT10 be replaced by a substitution font and that this font shall be referenced in PDF. For all the other fonts, AFP2web is to use the original fonts:

```
[CHARSET RENDERING]
C1D0GT10=1
*=0
```

In the appropriate section of the mapping.def, we also specify which substitution font to use for the target format PDF. The file name of the font is lucon.ttf for Lucida Console. Because there is no global naming convention for fonts, AFP2web uses file names to find a PC font. We therefore assume that the typeface name is part of the file name and we specify the base part of the file name in the mapping.def:

```
[PDFFONT]
;
; user defined
GOTHIC=lucon
```

Our example specifies the font path as **ExtFontPath=c:\WINDOWS\Fonts** and uses the font file lucon.ttf.

The following command starts the conversion:

```
afp2web.exe -lf -lp:pdf -c -q samples\insure.afp
```

The information in the LOG-file shows that the font mapping has taken place. The AFP Character Set C1D0GT10 has been mapped to the typeface "Lucida Console":

```
==>CS=C1D0GT10, FGID=40, TF=GOTHIC, W=M, S=10.00, RF, CP=T1D0BASE
-->UsedFont=C:/WINDOWS/Fonts/lucon.ttf, TF=LucidaConsole, W=M,
```

S=10.00, TT, Ref, ACP=MD:2063.cp, ENC=ANSI
@(2448,5353)>Geoffrey R Stephens<

LOG Text	Description
TF=GOTHIC	The AFP font has the typeface name "GOTHIC"
UsedFont	The font file used as substitution font. Its typeface name "Lucida-Console" is taken from the font file "lucon.ttf"
Ref	Indicates that the font is referenced in the PDF file

The conversion result has improved . The text string "Geoffrey R Stephens" looks like this in the original AFP:

Figure 9 The original AFP monospaced font

Geoffrey R Stephens

and is now mapped to a more appropriate mono-spaced font in the resulting PDF:

Figure 10 Mapped to Lucida Console in PDF

Geoffrey R Stephens

4.2.2.2 Converting PDF Documents with Substitution Fonts

General Information

You use the mapping.def to specify that a PDF font be substituted by an AFP font:

- The section **[CHARSET RENDERING]** specifies the processing mode for each PDF font typeface.
- For the conversion to AFP, the section **[AFPFONT]** maps PDF font typeface to AFP Character Sets and Code Pages.
- For the conversion to a raster format (such as TIFF, JPEG), the section for the system platform, either **[WINFONT]** or **[UNIXFONT]**, maps fonts to substitution fonts.

To configure substitution font for PDF input Fonts, PDF font typeface name need to be known. To get PDF font typeface name for PDF input file, run AFP2web with log option.

```
afp2web.exe -q -c -afp -lf -lp:pdf samples\insure.pdf
```

PDF font typeface name is logged as typeface name(TF) in AFP2web log file as shown below. Use this name in **[CHARSET RENDERING]** and **[AFPFONT]** sections to map AFP Character Sets and Code Pages for PDF Fonts.

```
==>TF=FreeMono, W=M, S=10.00, Type1, ENC=WinAnsi, Emb  
-->CS=CZ4200, TF=FreeMono, W=M, S=10.00, OF, CP=T1001141, ACP=MD:1141.cp  
@(510,1115)>Geoffrey R Stephens<
```

Example: PDF2AFP With Substitution Fonts

The following example specifies that all fonts with typeface names beginning with "FreeMono" are to be replaced with their appropriate substitution fonts:

```
[CHARSET RENDERING]
FreeMono*=1
```

If the RenderingFlag 1 is set, substitution fonts used must be available in [AFPFONT] section. If necessary, add entries to the section [AFPFONT].

The following shows the entry for the "FreeMono":

```
[AFPFONT]
FreeMono=CZ4200,T1001141
```

The AFP substitution fonts, which you use, must be available as font resources (either as Coded Font or as AFP Character Set and Code Page). Font resources must be found in the predefined resource paths. You define the resource paths in the INI-file, either as the general path for resources (ResPath) or as a path to AFP font resources (AFPFontPath).

The following command in the demo.bat starts the conversion:

```
afp2web.exe -q -c -afp -lf -lp:pdf samples\insure.pdf
```

The following lines in the LOG-file indicate that the font substitution has taken place. FreeMono is mapped to the character set CZ4200:

```
==>TF=FreeMono, W=M, S=10.00, Type1, ENC=WinAnsi, Emb
-->CS=CZ4200, TF=FreeMono, W=M, S=10.00, OF, CP=T1001141, ACP=MD:1141.cp
@(510,1115)>Geoffrey R Stephens<
```

LOG Text	Description
FreeMono	FreeMono is input PDF font name
CZ4200	The character set used in the AFP document
OF	Outline Font

4.2.3 Font Substitution and Embedding

In cases, where it is not preferable to use the original font, you can specify that a substitution font replace the original font. You choose to embed the font to make the target document selfcontained, for example for archival.

4.2.3.1 Converting AFP Documents to PDF with Substitution Fonts

Example: AFP2PDF and CharSetRendering=2 (Mapping AFP GOTHIC to Windows Gothic)

The input format is AFP and the output format is PDF. AFP2web automatically associates the AFP font with the typeface name GOTHIC to the Windows font with the file name "Gothic.ttf".

The configuration file afpcp\mapping.def specifies for the character set that a substitution font is to be used and embedded in PDF. All other fonts are to be rasterized:

```
[CHARSET RENDERING]
C1D0GT10=2
*=0
```

We do not need to specify a mapping in the mapping.def, if we can expect AFP2web to find the font file in the system. Therefore, we do not add definitions to the following section:

```
[PDFFONT]
;
; user defined
```

AFP2web will find the font if the following is true:

- The parameters of the INI-file (ExtFontPath) specifies the path to font files.
- The font file adheres to the naming convention (typefacename=filename.*).

If the font has a weight of "Bold" or a style of "Italic" then the filename must have an appropriate suffix. For example, when searching for a font "GOTHIC,Bold", AFP2web builds the file name to look for using the typeface name (GOTHIC) concatenated with the first suffix defined in the [FONTSUFFIX] section of the mapping.def. If no matching font file name (in our case, GOTHICb.*) is found, AFP2web builds a new file name using the second suffix (GOTHIC-Bold.*) and so on.

In the mapping.def we define file name suffixes for bold and italic fonts as follows:

```
[FONTSUFFIX]
BoldSuffix=b,bd,-Bold
ItalicSuffix=i,-Italic,-Oblique
BoldItalicSuffix=bi,-BoldItalic,-BoldOblique
```

Our example uses ExtFontPath=c:\WINDOWS\Fonts and the font file GOTHIC.ttf.

The following command starts the conversion:

```
afp2web.exe -lf -lp:pdf -c -q samples\insure.afp
```

The information in the LOG-file:

```
==>CS=C1D0GT10, FGID=40, TF=GOTHIC, W=M, S=10.00, RF, CP=T1D0BASE
-->UsedFont=C:/WINDOWS/Fonts/GOTHIC.TTF, TF=CenturyGothic, W=M,
S=10.00, TT, Emb, ACP=MD:2063.cp, ENC=ANSI
@(2448,5353)>Geoffrey R Stephens<
```

LOG Text	Description
TF=GOTHIC	The AFP document character set C1D0GT10 has the typeface name GOTHIC.
UsedFont	The font is correctly mapped to the specified font file
Emb	Indicates that the font is embedded in the PDF file

The text string "Geoffrey R Stephens" in the original AFP is:

Figure 11 AFP original font monospaced

Geoffrey R Stephens

The conversion result in PDF is:

Figure 12 Automatically embedded Gothic

Geoffrey R Stephens

A font substitution cannot guarantee true fidelity of the conversion results, as can be seen in this example. We also assume that a mono-space font would be more appropriate as a substitution font.

In this example, we see that it is better to explicitly specify the substitution font and not to rely on an automatic search for font files. We can improve the conversion results, by specifying a mono-spaced substitution font:

```
[CHARSET RENDERING]
C1D0GT10=2
*=0
[PDFFONT]
;
; user defined
GOTHIC=COUR
```

The information in the LOG-file:

```
==>CS=C1D0GT10, FGID=40, TF=GOTHIC, W=M, S=10.00, RF, CP=T1D0BASE
-->UsedFont=c:/WINDOWS/Fonts/cour.ttf, TF=CourierNewPSMT, W=M, S=10.00,
TT, Emb, ACP=MD:2063.cp, ENC=ANSI
@(2448,5353)>Geoffrey R Stephens<
```

The text string in the PDF file:

Figure 13 Courier Font Selected For Embedding

Geoffrey R Stephens

5 AFP2web 4.x Scripting Facility User Guide

This chapter explains how to use the AFP2web Scripting Facility to customize standard AFP2web processing. The information includes:

- Basic concepts and suggestions for what you can do,
- An introduction to the parsing events, packages, and methods

For a detailed information on syntax and commands, please refer to [Scripting Facility API Reference](#) and [Scripting Facility Quick Reference](#).

Script examples are delivered with the AFP2web product. You can use these examples as a starting point for your own custom scripts. For a description of these examples, please refer to the appendix.

5.1 Basic Concepts of the AFP2web Scripting Facility

5.1.1 Scripting Facility Applications

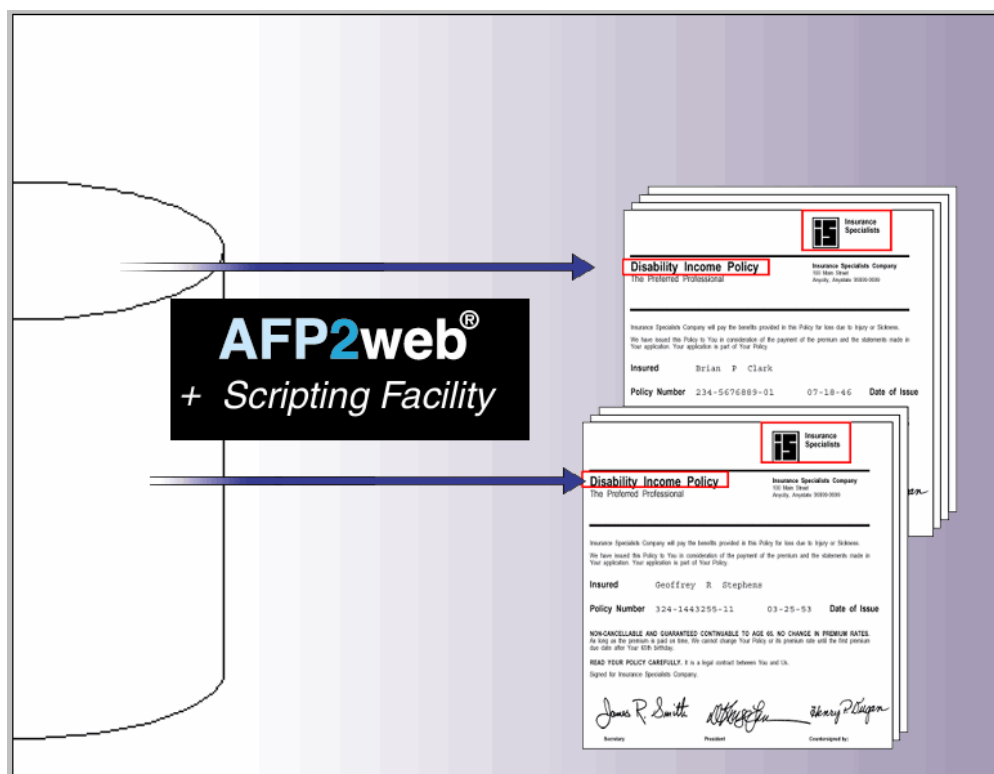
The Scripting Facility is the AFP2web interface used to customize AFP2web document parsing and conversion. When the Scripting Facility is activated, AFP2web passes control to a user-defined script at particular processing events. The Scripting Facility enables intelligent document processing. Examples of tasks performed by a script include:

- Splitting an AFP spool file into individual output documents and pages according to rules, which you define and which are not limited to any standard AFP structure. If your AFP data contains the standard AFP indexes as described elsewhere in this chapter, AFP2web splits AFP spool files automatically. If you are coping with different requirements, the Scripting Facility gives you all the flexibility you need to control this process.
- Extracting data from AFP document content to create detailed document indexes. Here again, AFP2web offers a standard method of extracting data from standard AFP indexes and delivers index data in basic formats. However, if the AFP format itself comes in application-specific variations or if you need more information, you use the Scripting Facility to extract index data from any object in the AFP data stream and to deliver index data in any required format.
- Adding, changing or removing document content, such as text, watermarks, bookmarks, background forms, images, indexes, annotations, and other elements. If you need to process, re-purpose, or enhance your documents, you use the Scripting Facility to do the job.
- Trigger any pre- or post-processing action for the current page or document. For example, you might want to forward document indexes to the archive system. Instead of having the archive polling for any incoming data, you can thus actively trigger an archive update.
- Acting as a plug-In mechanism for easy integration into existing applications. AFP2web works as a functional unit in document processing chain. Together with the Scripting Facility, AFP2web produces results for follow-on processes (converted documents, index data) in one single job step and thus helps streamline the workflow. The possibilities of using the AFP2web Scripting Facility is basically unlimited. In the following sections, we give a few examples of typical application scenarios using AFP2web and the Scripting Facility.

5.1.1.1 Document Recognition and Separation

Define rules for splitting an AFP spool file into individual documents and document pages.

Figure 14 Scripting Facility - Document Splitting



A script formulates rules for recognizing documents and pages. These rules can be based on printable and non-printable information in the AFP spool file, for example, printed text, the occurrence of overlays and other objects, and encoding values passed through AFP-specific NOP fields.

Document recognition and separation is run as a fully automated process.

Example: An AFP data stream contains a spool file with insurance bills for many customers. To split this spool file for each customer, we can look for a particular text string (eye catcher) on that page. Finding this eye catcher, we mark this page as the beginning of a new document. In addition, we capture the name of the customer for our index.

5.1.1.2 Data Extraction and Document Indexing

Extract visible and non-visible information from AFP documents. Deliver index data in any format, for example, CSV or XML.

Figure 15 Scripting Facility - Data Extraction and Indexing

AFP2web[®] + Scripting Facility

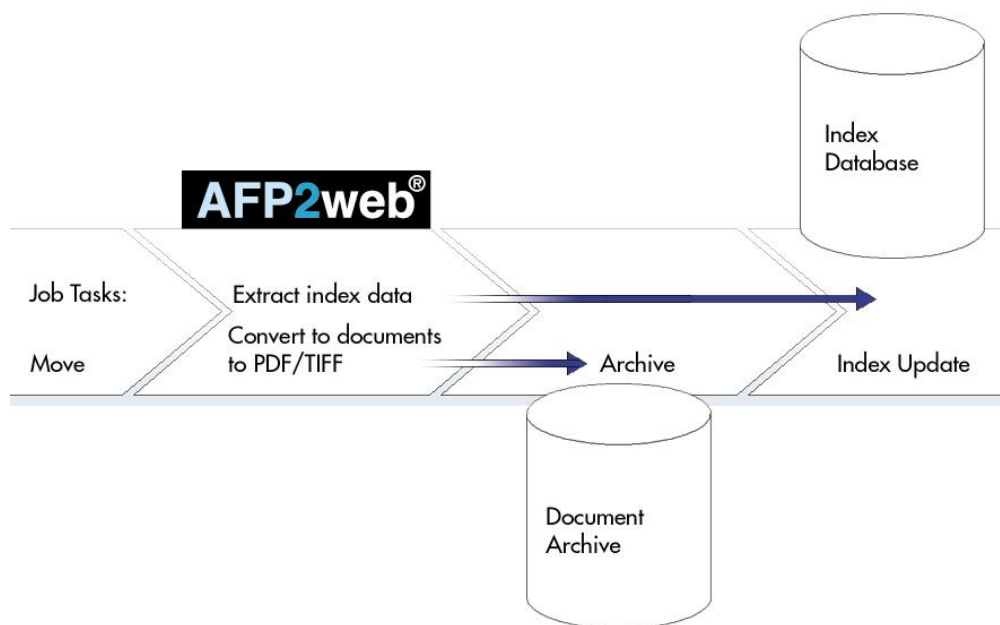
document 1 index file	document 2 index file
Filename=INSURE_s7q.1.pdf	Filename=INSURE_s7q.2.pdf
File Size=16141	File Size=20432
Page Count=3	Page Count=4
Insured=Geoffrey R. Stephens	Insured=Brian P. Clark
Policy=324-1443255-11	Policy=234-5676889-01
Date of Issue=03-25-53	Date of Issue=07-18-46

Using Scripting Facility functions, a custom script is designed to access visible text and non-visible data anywhere within a document. An intelligent extraction of index data is possible. The Scripting Facility runs as a full automated process extracting index data from mass print documents. Index data can be delivered in any required format

5.1.1.3 Document Conversion and Archival

The tasks to solve include converting incoming documents, extracting data, delivering converted documents to the archival system, obtaining the resulting document IDs, and finally delivering both document IDs and extracted data to the index database.

Figure 16 Scripting Facility - Archival Flow

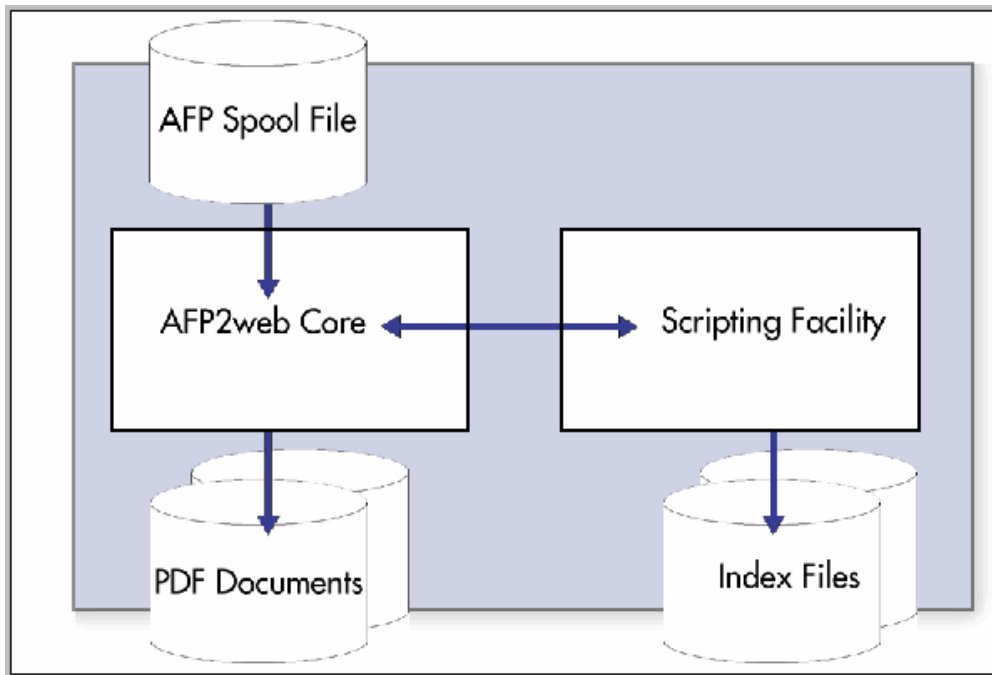


Thanks to the AFP2web core functionality, documents are quickly converted to PDF or TIFF for the archives. Using the AFP2web Scripting Facility, a custom script is designed to access visible text and non-visible data anywhere within a document. The custom script extracts index data of any required level of detail and delivers the index data in any required format.

After document conversion, the script finally intercepts a "finalizeDoc()" event to trigger the next process. This process forwards the converted document to the archives, obtains the document ID, and passes document ID and extracted index data to the index database.

5.1.2 Scripting Facility Interaction With AFP2web

AFP2web interacts with the AFP2web Scripting Facility, giving you the opportunity to intercept particular events, such as initializing, processing, and post-processing either the current document or page.



The AFP2web Scripting Facility is based on Perl. An embedded Perl engine (perl dll) is delivered with the AFP2web installation. To provide additional functionality or to add new modules, you must install Perl.

AFP2web also provides script examples showing how to solve basic tasks. These are described in the Appendix.

5.1.2.1 The Objects of the AFP2web Scripting Facility

To access all document objects required for your scripting, you include those Perl packages which define the methods for the objects you need.

The following table gives a short overview of the packages, which you can use to read or write object properties:

Package	Objects Which Are Accessed
a2w::Bookmark	PDF bookmark
a2w::Comment	Comment object
a2w::Config	AFP2web processing parameters
a2w::Document	The document
a2w::Font	Font settings
a2w::Index	Set attributes to be used as index data
a2w::Kernel	Exposes additional AFP2web functionality
a2w::Line	AFP line object
a2w::MediumMap	AFP medium map

a2w::NOP	Invisible AFP “No Operation Fields” used to carry data or processing instructions in the AFP data stream
a2w::Overlay	AFP overlay
a2w::Page	Document page
a2w::PSEG	AFP page segment
a2w::Text	Text
a2w::UserData	User defined data added from Scripting Facility

5.1.2.2 The Parsing Events of the AFP2web Scripting Facility

At specific parsing events, AFP2web activates a subroutine in your main script module (the default is `afp2web.pm`). The following chart gives you a quick overview of these parsing events and the correlating subroutines.

AFP2web Kernel Event	Subroutine in <code>afp2web.pm</code>	Scope
Process begin (Only once)	<code>initialize()</code>	global
Document begin detected	<code>initializeDoc()</code>	global
Document processed	<code>finalizeDoc()</code>	global
Process ending (Only once)	<code>finalize()</code>	global
The following subroutines are called in sequence as one logical step for a document page:		
Page begin detected	<code>initializePage()</code>	document
Page parsed	<code>afp2web()</code>	document
Page processed	<code>finalizePage()</code>	document

5.1.2.3 The Required Subroutines in a Script

Each subroutine takes particular objects as input parameters. In a subroutine, you save a reference to the object so that the methods of this object becomes available at a later event. In the following description, we indicate what can be done in a subroutine. For more details, please refer to the reference and to the script examples.

initialize()

This routine is performed only once at the beginning of the AFP2web process. It is used to access and configure run time parameters (INI file or command line options) and to read information about the spool file.

Usage examples:

- Reading the scripting arguments which are entered with the command line option `-sa`.
- Set the `AutoSplit` property to `TRUE` if you want to control document splitting using the standard AFP index.

initializeDoc()

This routine is performed at the beginning of each process preparing to write the output document. It is called at the beginning of each output document process. You can use this routine to initialize output document dependent processes.

initializePage()

This routine is performed at the beginning of each new page. From this point on, you can access the methods of the page.

afp2web()

The main entry point in the AFP2web Scripting Facility.

This routine is performed for each and every page. You use this routine to access any object of the page, for example to:

- retrieve references to all child objects
- use the object's methods to access the object's properties.
- formulate the parsing rules for identifying the first page of a new document, for skipping the page, for appending the page to the current document.
- collect index data for the page.

finalizePage()

This routine is performed for each and every page. Called after processing each output page, You can do page finalizing processes.

finalizeDoc()

This routine called after the document has been physically written. At this point, the document conversion is complete (PDF or a raster format, such as TIFF).

Normally, you use this routine to build, format and output the indexes for the document. You can also set the document name, for example with a time-stamp.

Because the output document is available, you can do any post-processing for the document like passing it to your archive.

finalize()

End of AFP2web process and after processing the input AFP spool file. You can use this routine for post processing actions.

5.1.3 Scripting Facility Processing Features

5.1.3.1 General Features

5.1.3.2 The Default Encoding

Please note that the Scripting Facility is ASCII-based. ASCII is the encoding used when parsing AFP resource names, AFP index elements, and is used as the base encoding for the Scripting Facility.

The mapping.def determines which code page file to use for an AFP code page. The code page file (CP file) lists the ASCII code points to use for each character.

5.1.3.3 The Coordinate System Used by the Scripting Facility

The top left is the origin of coordinate system used by the Scripting Facility.

The X value increases the direction to the right.

The Y value increases the direction downwards.

5.1.3.4 Global vs. Document Scope (Handling Fonts Within a Script)

Please note that a script operates in two different scopes:

- **Global scope** which begins with the initialize() routine and ends with the finalize() routine.
- **Document scope** which starts with initializeDoc() and ends with finalizeDoc().

Fonts, which are allocated in the initialize() function, will have a global scope.

Fonts, which are allocated in one of the functions initializeDoc(), initializePage(), afp2web(), or finalizePage(), will have document scope.

The following is an issue in Perl - When you declare a font variable, try to avoid using the "my" keyword if you want to access the variable within a global scope.

5.1.3.5 The Processing of Document Index Data

From the viewpoint of AFP2web, the index data found in an AFP file is divided into two categories:

- Standard AFP indexes
- Non-standard indexes.

Standard AFP Indexes

We assume that you generally understand that elements in the AFP data stream are encoded as structured fields. For more information on AFP and MO:DCA, please refer to the appropriate documentation of IBM. In the following, we name the two elements, which are recognized as standard AFP indexes.

Standard AFP indexes are objects known as Index elements (IEL). Within the AFP data stream, an Index Element (IEL) is located within a document index – within the structured fields Begin Document Index (BDI) and End Document Index (EDI):.

The Tag Logical Element (TLE) provides the attribute name and attribute value, which can be used as an index entry for the page or page group. The TLE may contain an explicit reference to the page or document. A document or a page is generally referred to as an indexed object.

If the AFP spool file contains standard AFP indexes, the AFP2web Scripting Facility is not in all cases necessary to extract index data. AFP2web provides a standard method for processing standard AFP indexes. You activate this standard method if you set the INI-file parameter **FileCreationMode=DOC_INDEX**. Using this option, you create files with index information in one of the standard formats available.

When AFP standard indexes are available, the Scripting Facility is normally used to enhance these with additional data taken from the document, and to deliver index data in a custom format.

Non-Standard Indexes

Your application might require a custom solution for document indexing. Solutions, which are quite common, include:

- Passing document index information via NOP (no operation) structured fields.
- Capturing text (eye catchers) within the page content.
- Detecting specific overlays and capturing text marked by these overlays.

Using the AFP2web Scripting Facility, you can process these elements to collect index data from the document content. You can build new indexes or add information to existing indexes. You can format and output the results according to your own standards.

For example, using the `addIndex()` function of the `a2w::Document` or `a2w::Page` object, you can insert additional indexes at both document and page level. When used together with the command line option `-bm`, AFP2web creates PDF bookmarks for these indexes.

5.1.3.6 Splitting an AFP Spool File Into Documents and Pages

If the data stream contains standard AFP indexes, AFP2web will be capable of splitting documents automatically.

Automatic Splitting with Autosplit

If you want AFP2web to handle document splitting automatically while running the AFP2web Scripting Facility, you must explicitly set the `autosplit` option to `TRUE`. This is done when initializing in the script for the process.

During automatic document splitting, AFP2web actively triggers the parsing events – the script you have programmed for these events are thus executed automatically.

A small amount of control is, however, still possible, that is, you can set flags to tell AFP2web to:

- Append the page to the current document, or
- Skip the current page.

Script-Controlled Splitting without Autosplit

There are cases, in which it is preferable to use a script to control splitting an AFP spool file into documents and pages. Examples include AFP data without the standard AFP indexes or application specific rules for splitting.

To handle document recognition in your scripts, you turn off the autosplit property. In your script, you access and investigate objects in the AFP data stream, such as:

- Text objects in the presentation data (eye catchers),
- Overlays and their text objects,
- NOP fields in the AFP data.

These characteristics are used to identify the type of page being parsed. The script must explicitly set a flag to tell AFP2web that the current page

- Marks the beginning of a new document.

In addition, flags are also set to tell AFP2web to:

- Append the page to the current document, or
- Skip the page.

With autosplit turned off, you must set these flags if you want to AFP2web to trigger parsing events and thus execute the script subroutines.

5.1.3.7 Exception Handling

In a custom script procedure, you can set any exception by returning a negative value (negative return code, message text). The syntax is:

```
return (rc, "errortext");
```

where rc is a negative value.

When AFP2web encounters a negative return value, processing will stop and the exception is issued as error message.

Example:

```
return ( -123, "afp2web(): missing..." );
```

AFP2web returns a message to the console as follows:

```
E088: Scripting Facility Error (rc=-123): afp2web(): missing...
```

5.2 Using AFP2web Scripting Facility

AFP2web delivers several scripting examples, which you can use as a starting point for your own script procedures.

afp2web.pm is the master template to use. It has all required subroutines implemented and placeholders for basic functions.

5.2.1 Activating the Scripting Facility

You configure AFP2web to run with the Scripting Facility by setting the appropriate parameters in the AFP2web configuration file (INI-file) or you enter the appropriate command line options when you start AFP2web.

The following table summarizes parameters and options for the Scripting Facility:

INI Parameter	Command Line Option	Description
FileCreationMode=DOC_COLD	-doc_cold	<p>Using this option, you activate the AFP2web Scripting Facility.</p> <p>-doc_cold AFP2web will not write any index files. This has to be done by the custom scripting module.</p> <p>You can extend the command line option for AFP standard index elements - or if you used the addIndex method of the Scripting Facility to add standard index elements. In this case, AFP2web will handle formatting and output of index data for you. Use one of the following extensions:</p> <p>-doc_cold:csv AFP2web writes index files in CSV format</p> <p>-doc_cold:xml AFP2web writes index files in XML format</p> <p>Note: If you do not extend doc_cold option, your custom Perl routine is responsible for formatting and output of index data.</p>
ScriptProcedure=<scriptFile>	-sp (scriptFile)	<p>The file name of the scripting routine to use. If not specified, AFP2web Scripting Facility will load "afp2web.pm" by default.</p> <p>Example of using the command line option:</p> <p>-sp:c:\afp2web\mysp.pm</p> <p>Note: If you call AFP2web.exe from a directory other than the AFP2web installation base, you must specify the absolute path to the main scripting module using the option -sp.</p>
ScriptArgument=parm1,parm2, ...	-sa:parm1,parm2, ...	<p>Using this option, you can pass any number of arguments to the AFP2web Scripting Facility.</p> <p>In the scripting routine, you use the method getScriptArgs() of the a2w::Config object to access these arguments as one single string and you parse this string according to your own rules (delimiter, keywords, etc.).</p>
Autosplit		<p>AFP2web will look for standard AFP index elements (TLEs) to recognize document boundaries. If Autosplit is set to TRUE the \$NEWDOC flag will have no effect.</p>
SkipPage, SkipObjectSize		<p>These INI parameters specify pages to ignore in the AFP spool file. The criteria you can set is a maximum byte size of a page's data stream. All pages which are smaller or equal to this size will be ignored. You can use these parameters to skip blank pages or pages with a standard content (for example: separator pages).</p>

5.2.2 Setting the Perl Environment

The AFP2web Scripting Facility has a Perl engine embedded (and delivered as Perl DLL).

If you call AFP2web from the root directory of the AFP2web installation and if you do not use additional external Perl modules, you don't have to set up the Perl environment.

5.2.2.1 Case 1: Different working directory

If you do not use external modules, but you call AFP2web from another path than the root directory of the AFP2web installation, then you must do the following:

- Tell AFP2web where to find the AFP2web Perl packages (a2w*.pm): You can set the PERLLIB environment variable. Instead, the simplest method is to modify the internal Perl array @INC as described below under "Case 2: External modules" on page 95.
- Tell the AFP2web Perl engine where to find the main module (afp2web.pm). This is done by passing the fully qualified name of the main module by using the -sp option.

Example:

```
-sp:c:\AFP2web\MyModule.pm
```

5.2.2.2 Case 2: External modules

If you use external modules, you must tell the AFP2web Perl engine where to find these external modules. You can set the PERLLIB environment variable. However, the simplest method is to modify the Perl internal array @INC.

Modifying @INC must come as the very first statement in the main script module, which is loaded by AFP2web.

The following example shows how to add runtime directories to the Perl search path. For this example, we assume:

- Your Scripting Facility module (afp2web.pm) requires standard Perl modules (XML::Writer) and your Perl installation is located in C:\Perl\.
- Your Scripting Facility module is not in AFP2web root installation directory, but instead in C:\temp\xxx
- AFP2web's Scripting Facility Package (a2w:*) is not located in the AFP2web root installation directory, but instead in c:\temp.

You add all directories to the Perl search path with the following statement, which comes as the first statement of the main script module loaded by AFP2web:

```
BEGIN {  
  unshift(@INC, ('C:\Perl\lib', 'C:\Perl\site\lib', 'C:\temp'));  
}
```

All that is left to be done is to inform AFP2web, which Scripting Facility module to load. You do this

- Either with the INI file parameter ScriptProcedure=C:\temp\xxx\afp2web.pm
- Or by using the command line option -sp:C:\temp\xxx\afp2web.pm.

5.2.3 Using the INI Parameter MemoryOutputStream

When running AFP2web with the INI Parameter MemoryOutputStream set to ON, the AFP2web Kernel writes the output document to a buffer in memory instead of to file. The purpose of this INI parameter is to make it possible for a scripting routine to access the output document using the appropriate methods of the a2w::Document package (getOutputBuffer, getOutputBufferLength).

A possible application of this is to post-process document content (for example by archiving the document buffer, modifying document content, etc.).

The appropriate event for accessing a buffered document in a script would be the finalizeDoc() sub routine.

An example of the coding statements used to access an output document in the memory buffer:

```
#---- Get PDF buffer length
$PDFBufLenTmp = $a2wDocumentPar->getOutputBufferLength();
#---- Get PDF buffer
$PDFBufTmp = $a2wDocumentPar->getOutputBuffer();
#---- From here on you can write the PDF buffer to a PIPE or
a database or a file or ...
#---- Sample to write to a file
my $OPFileNameTmp = "pdf/sf_" . $a2wDocumentPar->getOutputFilename();
if ( open( OPFILE, ">$OPFileNameTmp" ) ){
    return (-1, "Unable to open $0 : $!" );
}
binmode( OPFILE );
syswrite( OPFILE, $PDFBufTmp, $PDFBufLenTmp );
close( OPFILE );
```

6 AFP2web Reference

This chapter begins with a reference of the parameters controlling AFP2web core functionality:

- INI parameters (afp2web.ini)
- command line options
- mapping.def parameters
- IBM specific font naming conventions

Included is also the Scripting Facility Reference:

- Cross-reference overview of methods and where used in a script
- Scripting Facility API Reference

This chapter also includes:

- A list of error messages and codes.

6.1 Overview of INI Parameters and Command Line Options

In the following, the INI file parameters and their corresponding command line options are grouped by category and listed side by side.

Licensing information		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
Licensee		Licensee
SerialNr		Serial number
	-usid	Generation of a system key

Input / output formats		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
InputFormat	<inputformat>	Format of input documents
OutputFormat	<outputformat>	Format to convert to

Read from stdin, write to stdout / buffer		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
	-std	Both options: -si and -so
	-si	Read from stdin
	-so	Write to stdout
MemoryOutputStream	-mos	Write output document to a buffer in memory

Type of output (document splitting/merging and creation of index files)		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
FileCreationMode	-DOC / ALL / -DOC_INDEX / -DOC_COLD / -DOC_MERGE / -PAGE	Documents to create
PageOutput	-po	Create one file per page of the output document (only for TIFF as output format)

Processing index data		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
IndexFormat	-DOC_INDEX: CSV XML -DOC_COLD: XML CSV	Format of index data
IndexRecord		Additional fields for index data in CSV format
PDFBookmark	-bm	PDF bookmarks for index entries

Generating file names and output folders
--

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
FilenamePattern	-fnpt	Pattern for output file names
GenerateUniqueFile	-u	Generate unique file names
DocumentCount	-dc	Create sub folders for output and limit the number of output documents per sub folder.

Specifying folders (directories)		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
	-ip	Path to afp2web.ini
	-if	Path and file name of INI file
LogPath	-lp	Folder for LOG output
ResPath	-rp	Standard folder for AFP resources
CpPath	-cp	Folder with code pages
OutputFilePath	-op	Folder to write output documents
ExtFontPath	-fp	Font path to additional Type1 and TrueType fonts
IndexPath	-xp	Folder for writing index files.
OverlayPath	-ovp	Folder for AFP Overlay Resources
PageSegmentPath	-psp	Folder for AFP Page Segment resources
FormDefPath	-fdp	Folder for AFP FORMDEF Resources
AFPFontPath	-ap	Folder for AFP Font resources
TempPath	-tp	Temporary path is used to cache AFP2web temporary files. Default is the temporary path defined in the system. This temporary path must exist and AFP2web must have read/write permissions to this directory.

Code page defaults		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
CodePageDefaultChar		Replacement character
CodePage	-dcp	Default code page to use when translating strings and indexes of the AFP spool file to ASCII

Locating AFP resources		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
	-rf	AFP resource file
	-xf	The input file with the AFP index to be processed for the AFP document
	-fd	AFP FORMDEF
FormdefExt		File extension for AFP FORMDEFs
OverlayExt		File extension for AFP Overlays
PageSegExt		File extension for AFP Page Segments
CharSetExt		File extension for AFP Character Sets
CodePageExt		File extension for AFP code pages
CodedFontExt		File extension for AFP coded fonts

The section [FORMDEF]		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>

<Formdef Name>=<Medium Map Name 1>[,<Medium Map Name 2>]...[,<Medium Map Name n>]		Lists AFP FORMDEF and for each FORMDEF the medium maps, which apply
---	--	---

LOG files, messages, statistics		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
Quietmode	-q	Suppress or display processing information on the console
	-ncm	Redirect messages to file
Logging	-l	LOG output
LoggingFont	-lf	LOG output: font infos
LoggingLevel	-ll	Restrict LOG output to information categories.
ExceptionLoggingLevel	-ell	Print error messages in a selected level of detail
Statistic	-s	Write a statistic report (stat*.txt)

Spool file processing		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
Strict	-strict	Stop processing if AFP resource cannot be found.
Protocol	-p	Write „protocol“ with Doc IDs
StartingDocument	-sd	Doc ID, start converting at this position in spool file.
EndingDocument	-ed	Doc ID, stop converting after this document in spool file.
StartingPage	-pp:[fp][-tp]	Page range to convert (Starting page)
EndingPage	-pp:[fp][-tp]	Page range to convert (Ending page)

SkipPage		Ignore standard page
SkipObjectSize		Size of standard page

Additional program flags

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
	-h or -?	Help info about AFP2web (list of command line options)
	-v	Display version of AFP2web
	-vall	Display versions of all AFP2web components
LaunchPreview	-pv	Display converted document in viewer

PDF document properties, security settings, viewer preferences, conversion parameters

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
Creator		Author
Title		Title
Subject		Subject
Keywords		Keywords
PDFSecurity	-ps	PDF Security
PDFUIOptions	-pui	Viewer settings
PDFWinOptions	-pwn	Document display settings
PDFDocLimits		Memory allocation for PDF

Color	-c	Color
-------	----	-------

PDF output or input		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
Flush	-flush	Caching of output in memory or in file.
CodedOutput	-nco	Retain text encoding or not

TIFF (raster format) output		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
FormatType	-TIFF: compressionformat or: -compressionformat	TIFF format
Resolution	-pr	Resolution
Color	-c	Color
	-os	Size of output document in pixels or STANDARD paper size

ASCII output		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
MaxCols	-mc	Maximum columns per row
MaxRows	-mr	Maximum rows per page
	-iASCII	Display optimal values for the options–mc and –mr

AFP output		
------------	--	--

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
AFPOutputFlags	-aof	AFP output flags
AFPIIS2ComplianceFlag	-acl	AFP Compliance level
CurveSamplingFactor	-csf	Curve sampling factor
PageRotation	-r	Page rotation
PrintableLineWidth	-plw	Minimal line width for printing
ResourceGroup	-rgp	Controls the generation of resource groups
	-BW	Convert images to 1-bit pixel images (black/white)

Optimize included objects (graphics)

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
SaveOCDataOnDisk	-sod	Save Object Container Data on disk
TempPath	-tp	Temporary path is used to cache AFP2web temporary files. Default is the temporary path defined in the system. Temporary Path should exist.
JPEGQuality	-jq:n	Jpeg Quality where n can be 1 to 100
ImageResizeFilter	-isf	Filters to be used for image stretching and shrinking
ImageRotationFilter	-irf	Filter to use for rotating images

Color, IOCA Images to PDF

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
CMYKTORGb	-nocmyktorgb	Convert CMYK to RGB

Colorspace		Colorspace for converting IOCA to JPEG
ColorTolerance	-ct	Tolerance for mapping to AFP SOCA equivalent colors
Section [AFPColorTable]	-clr	RGB values for standard AFP OCA colors

Fonts, Vector Objects, GOCA, Lines, etc..

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
FontAliasing		Font anti-aliasing
FontResolution		Font resolution
MaxVectorObjects	-mvo	Maximum number of vector objects
ImageMaxCacheSize	-imcs	Maximum cache size for images
PDFFontHeightFactor	-pfh	PDF Font Height Factor

Document adjustments.

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
PageRotation	-r	Page rotation
Watermark	-wm	Watermark
OutputSize	-os	Page size or STANDARD paper size

AFP2web Scripting Facility

<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
------------------------------	----------------------------	-----------------

FileCreationMode=DOC_COLD	-doc_cold	Activates the AFP2web Scripting Facility. Also used to specify output format of index data.
ScriptProcedure	-sp	Name of the script file.
ScriptArgument	-sa	Arguments to pass to the script
ScriptUnitBase		Base unit of measurement
Autosplit		Split documents automatically using standard AFP Indexes (valid only for -doc_cold)

Raster Input Format		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
ImageProcessValues	-ipv	Option to specify default values for missing input image properties like resolution.

LPD/MMD input formats		
<i>Parameter of INI file</i>	<i>Command line option</i>	<i>Function</i>
custom named section	-sft	Option to specify the name of spool type section that has to be used while processing given LPD/MMD input spool
FileCreationMode=DOC_COLD	-doc_cold	Mandatory for LPD/MMD , Activates the AFP2web Scripting Facility to format LPD input and also process indexes

6.2 INI Parameter Reference

The following description lists the parameters of the INI file in alphabetical order. For an overview of parameters grouped by category, please refer to [Overview of INI Parameters and Command Line Options](#).

6.2.1 Function and Structure

afp2web.ini is the configuration file used to set global parameters of AFP2web. This file is divided into the sections:

- [Settings] For conversion parameters
- [FORMDEF] To map FORMDEFs to their Medium Maps.
- [AFPColorTable] To map standard AFP OCA colors to RGB values
- [SPOOLFILETYPE] To describe the carriage control type, record length like processing factors for LPD/MMD input file format
- [user-defined section] To define FTP locations of remote AFP resources

6.2.2 Usage Notes

When you make changes to afp2web.ini, please note that:

- A semicolon (";") at the beginning of a line marks a comment line.
- A path without a drive will be interpreted as relative to the directory with the INI file.
- The parameters are not case-sensitive.
- An entry within the INI file must not exceed 1024 (1 Kb) characters.

6.2.3 Section [Settings]

6.2.3.1 AFPFontPath=<path>

Path to AFP font resources.

It is also possible to specify a comma-separated list of paths.

Example: -ap:samples/resource,c:\afp\fonts

Default: samples/resource

6.2.3.2 AFPI2ComplianceFlags=<AllowSOCAColorFlag>, <ForcePTValueControlFlag>

AFPI2ComplianceFlags specify conditions to be applied for AFP output with IS2 compliance:

Syntax: AFPI2ComplianceFlags=<AllowSOCAColorFlag>, <ForcePTValueControlFlag>

AllowSOCAColorFlag	<p>For Modca-IS2 compliance, PT1 subset must be used. As per PT1 subset only color allowed in PTOCA is Device Default color(0xFF07).</p> <p>This flag enables to use other colors specified in SOCA color table</p> <p>Value= on off</p> <p>on: Enables to use colors defined in SOCA Color table for PTOCA object.</p> <p>off: Restrict to use only Device Default color(0xFF07) not any other colors defined in SOCA Color table.</p>
ForcePTValueControlFlag	<p>For Modca-IS2, PT1 subset must be used. PT1 subset defines a valid range of values for PTOCA control sequences</p> <p>Value: on off</p> <p>on: Strictly checks PTOCA control sequence values are within valid range defined by PT1 subset. Otherwise exception is thrown and process is terminated.</p>

	off: Allows PTOCA control sequences to have out of range values also.
--	---

Example: AFPIIS2ComplianceFlags=on,off

Default: AFPIIS2ComplianceFlags=on,off

6.2.3.3 AFPOutputFlags=<Write5AFlag>

AFPOutputFlags specifies conditions to be applied for AFP output

Syntax: AFPOutputFlags=<Write5AFlag>

Write5AFlag	<p>For AFP output, AFP2web by default writes Structured Fields (SFIs) with 0x5A introducer. This flag will allow to change this default behavior, thus 0x5A introducer can be skipped.</p> <p>Value= on off</p> <p>on: Instructs to write 5A introducer for AFP SFIs on output</p> <p>off: Instructs NOT to write 5A introducer for AFP SFIs on output</p> <p>Default is on</p>
--------------------	---

Example: AFPOutputFlags=off

Default: AFPOutputFlags=on

6.2.3.4 AutoRotateImage=<on/off>

Rotates input images having non zero degree rotation to zero degrees.

on	Rotate image and place rotated image on output and set rotation as zero degree on output
off	Do not rotate the image and place as is on output and specify given rotation on output.

Default: off

Applies only for PDF2AFP conversion

6.2.3.5 Autosplit=<on/off>

Automatic Document splitting based on standard AFP Indexes. Only for FileCreationMode=DOC_COLD.

If Autosplit=on, then the document begin/end events of the Scripting Facility are automatically triggered. This

means, passing a return code of 2 from afp2web() will have no effect.
Default: off

6.2.3.6 BWSampler=<Dither|Threshold:Adaptive|Threshold:1..100>

B/W sampler for 1 bpp RASTER outputs (like TIFF:G4, TIFF:G3 so on). Input page is bit demoted to 1 bpp using configured sampler technique.

where value could be any one of the following

Value	Description
Dither	Use dithering technique to bit demote input page, DEFAULT
Threshold:Adaptive	Use adaptive threshold technique to bit demote input page
Threshold:1..100	Use percentage based threshold technique to bit demote input page

Default: Dither

6.2.3.7 CharSetExt=<file extension>

File extension for AFP Character Sets.
Default: *

Normally AFP2web looks for <resource name>*, so it is normally not necessary to use this parameter. But if -strict is set and a file extension is used, then AFP2web looks for <resource name>.<file extension>

6.2.3.8 CMYKTORGB=<on/off>

Only for IOCA FS45 color images output to PDF (only for AFP2PDF conversion).

on	Combine IOCA FS45 CMYK color image planes as RGB.
off	Pass through IOCA FS45 CMYK color image planes.

We recommend using this option only for print documents. CMYK for PDF results in large PDF files. You should apply the standard conversion to RGB if you need documents for online viewing.

Default: CMYKTORGB=on

By default, the INI file parameter CMYKTORGB is on - that is, CMYK color planes are converted to RGB for PDF. By using **-nocmyktorgb** as command line option it can be turned off.

6.2.3.9 CodedFontExt=<file extension>

File extension for AFP Coded Fonts

Default: *

Normally a2w looks for <resource name>*, so it is normally not necessary to use this parameter. But if -strict is set and a file extension is used, then AFP2web looks for <resource name>.<file extension>

6.2.3.10 CodedOutput=<on/off>

Retain text encoding or not (on/off) .

on	Output will be rendered as text or image as given in the input
off	Output will be rendered completely as image and text searching will not be not possible.

Default: on

AFP2web will retain the text encoding. Using **-nco** as the the command line option explicitly turns off text encoding. This option applicable only for PDF input.

6.2.3.11 CodePage=<defaultcodepage>

Defines the default code page to use for translating resource names, NOPs and indexes of the AFP spool to ASCII. (Examples include: index names and values, SFI names such as Doc Name, Page Name, Resource Name, NOPs.)

The mapping.def maps this AFP code page to the code page file (CP file). The default is also set in the mapping.def.

Default: T1GIG500

6.2.3.12 CodePageDefaultChar=" "

Code page default character.

This is the character that will be used to display a non-mapped character.

Default: " "

6.2.3.13 CodePageExt=<file extension>

File extension for AFP code pages.

Default: *

Normally AFP2web looks for <resource name>*, so it is normally not necessary to use this parameter. But if -strict is set and a file extension is used, then AFP2web looks for <resource name>.<file extension>

6.2.3.14 Color=<on/off>

Applies to output formats supporting colors. Leave colors as is or convert these to black/white (text and image):

off	Black/white
on	Preserve colors (same as command line option -c)

Default: off

6.2.3.15 Colorspace=<colorspace>

Colorspace to be used when compressing IOCA images as JPEG. Possible is a value out of:

RGB	(Default)
YCbCr	

Notes:

- CMYKTORGB must be set to on to activate the Colorspace option.
- Colorspace=YCbCr overwrites the CMYKTORGB=on option.
- Colorspace is not used while passing JPEG as is to PDF output.

Default: RGB

6.2.3.16 ColorTolerance=<n>

Color tolerance value, which is applied to find the AFP SOCA equivalent color for any input objects using the following formula:

$(\text{AFP SOCA Color} - \text{Color Tolerance}) < \text{Input object color} < (\text{AFP SOCA Color} + \text{Color Tolerance})$

If the input color falls in any one of these ranges, it is treated as an AFP SOCA color.

Default: 0

6.2.3.17 Creator=<doc-info>

Document property: Author

Default: See default value in the INI file

6.2.3.18 CurveSamplingFactor=<n>

Factor used to smooth curves. A value from 1 to 65535. Modify this factor to get smooth curves on AFP output. The lower the factor the smoother the curves. Applies only when converting to AFP.

Examples:

CurveSamplingFactor=64 (as INI parameter)

-csf:64 (as command line option)

Default: 64

6.2.3.19 DocumentCount=<n>

Create sub folders for output and limit the number of documents per sub folder.

This parameter is used together with FileCreationMode=DOC_INDEX

Example:

DocumentCount=500 together with FileCreationMode=DOC_INDEX limits the number of output documents to 1000 per sub folder: You receive per sub folder 500 documents plus 500 index files.

When used as a command line parameter:

Create sub folders for output and limit the number of documents per sub folder.

This option is used together with the command line parameters

-DOC_INDEX or with

-DOC_COLD:XML or DOC_COLD:CSV.

Example:

The command line option `-dc:500` together with `-doc_cold:xml` limits the number of output documents to 1000 per sub folder: You receive per sub folder 500 documents plus 500 index files in XML format.

6.2.3.20 EndingDocument=<n>

Last document to convert in a spool file. For EndingDocument, enter the document ID as determined in the "protocol".

This option is only possible if DOC_INDEX or DOC_COLD is selected

Default: 0

6.2.3.21 EndingPage=<endpage>

Page Range to convert (Ending page)

Default: -1 (last page)

6.2.3.22 ExceptionLoggingLevel=<n>

Print error messages in a selected level of detail.

You specify one of the following:

1	Log exception message alone
2	Log file name and line number with exception message

Default: 1

6.2.3.23 ExtFontPath=<path>

Font path to additional Type1 and TrueType fonts.

Default: extfont

6.2.3.24 FileCreationMode=<option>

Output files to produce:

ALL	One output file for all AFP documents
DOC	One output file per AFP document
DOC_INDEX	Create output documents with index files.
DOC_COLD	Activate the AFP2web Scripting Facility.
DOC_MERGE	Merge all files into one output file. (For converting TIFF to PDF or for converting AFP.)
PAGE	One file per page in an AFP document.

Default: ALL

The following applies, when entered as command line option -<filecreationmode>:

DOC_INDEX with additional extension specifies the format of the index file to create. Possible is one of:

-DOC_INDEX:CSV	Index file in CSV format
-DOC_INDEX:XML	Index file in XML format

When used together with the AutoSplit function of the AFP2web Scripting Facility, DOC_COLD specifies processing and, in addition, the format of the index file to create:

-DOC_COLD	AFP2web will not write any index files. This has to be done by the custom scripting module.
-DOC_COLD:CSV	AFP2web writes index file in CSV format.

-DOC_COLD:XML	AFP2web writes index file in XML format.
---------------	--

6.2.3.25 FilenamePattern=<pattern>,Keyword1,Keyword2...

Filename pattern (without extension) used to define any pattern for filenames by using constant text and/or placeholders for reserved keywords.

The syntax is:

<pattern>[,Keyword1,Keyword2...

The syntax for the command line is:

-fnpt:<pattern>[,Keyword1,Keyword2...

<pattern> specifies a formatting pattern in the syntax of a C printf command.

When used as pattern for the TIMESTAMP value (see below), you use the syntax of a C strftime command. The keywords you specify are used to create the filename according to the pattern. It is important to ensure that data types of the keyword match with the formatting pattern.

Possible as keywords:

DOCID	Document ID (data type: Integer)
PAGEID	Page ID (data type: Integer)
PID	Process ID (data type: String)
TIMESTAMP[:<pattern>]	Current time stamp (data type: String) The default TIMESTAMP format is "%y%m%d%H%M%S"
SPOOLNAME	Spool file name (data type: String)

Which default is used as FilenamePattern depends on your settings for **FileCreationMode** and **GenerateUniqueFile**, as indicated in the following table:

Condition	Default for FilenamePattern
FileCreationMode is either FILE_MERGE, DOC_INDEX, DOC_COLD, DOC, or PAGE	"%s_%s.%d",SPOOLNAME,PID,DOCID
GenerateUniqueFile=on	"%s_%s.%d",SPOOLNAME,PID,DOCID
FileCreationMode=ALL and GenerateUniqueFile=off	"%s",SPOOLNAME
PageOutput=on (or when using the command line option -po)	"%s_%s.%ld.%ld",SPOOLNAME,PID,DOCID,PAGEID

6.2.3.26 Flush=<on/off>

Caching of output to either memory or file (on/off) .

on	Cache output page as file in temporary path (which is set using TempPath)
off	Cache output page in memory.

Default: off

Applicable only for PDF output

6.2.3.27 FontAliasing=<on/off>

Font anti-aliasing flag (on/off) .

on	Apply anti-aliasing when rastering text
off	Do not apply anti-aliasing when rastering text.

Default: off

6.2.3.28 FontResolution=<n>

AFP Font Resolution 240/300 dpi.

Default: 300

6.2.3.29 FormatType=<tiffoutputformat>

Applies only to TIFF: Compression option for the resulting TIFF file.

For the compression format, specify a value out of:

G3	ITU Group III
----	---------------

G4	ITU Group IV (Default)
JPG	TIFF with JPEG compression
LZW	TIFF with LZW compression
PACK	TIFF, packed
UNCOMPRESSED	TIFF without compression

The following applies, when entered as a command line format:

When entered as a command line option, enter the option **-TIFF** and specify one of the compression formats as a separate option:

-G3
-G4
-JPG
-LZW
-PACK
-UNCOMPRESSED

It is also possible to extend the option **-TIFF** as follows:

-TIFF:G3
-TIFF:G4
-TIFF:JPG
-TIFF:LZW
-TIFF:PACK
-TIFF:UNCOMPRESSED

It is basically possible to enter a specific subformat either as an extension to the output format option (example: **-AFP:BW**) or as a separate option or (example: **-AFP** and **-BW**).

6.2.3.30 **FormdefExt=<file extension>**

File extension for FORMDEF resources.

Default: *

Normally AFP2web looks for <resource name>*, so it is normally not necessary to use this parameter. But if -strict is set and a file extension is used, then AFP2web looks for <resource name>.<file extension>

6.2.3.31 **FormdefFilename=<formdef filename>**

Standard FORMDEF to be used for the AFP document.

6.2.3.32 **FormDefPath=<path>**

Path to FORMDEF resources.

It is also possible to specify a comma-separated list of paths.

Example: samples/resource,c:\afp\formdef

Default: samples/resource

6.2.3.33 **GenerateUniqueFile=<on/off>**

This parameter tells AFP2web to generate unique file names for output files.

on	Generate unique file names
off	Do not generate.

Default: off

This parameter is maintained for backward compatibility. For this version of AFP2web it is no longer required and we even recommend not to use it.

6.2.3.34 ImageMaxCacheSize=<n>

Maximum memory cache size for image, specified in Megabytes.

AFP2web forces image caching from memory to temporary file when the image size exceeds this limit.

Default: 5.0 (5 Megabytes)

6.2.3.35 ImageProcessValues=[<DefaultInputResolution>]

ImageProcessValues specify default values for missing input image properties like resolution.

Syntax:ImageProcessValues=[<DefaultInputResolution>]

DefaultInputResolution	It is the used as resolution for input images when input image resolution is zero or resolution is not given in input image. Value is specified in dots/inch (DPI)
-------------------------------	--

Example:ImageProcessValues=72

Default:ImageProcessValues=72

6.2.3.36 ImageResizeFilter=<Stretch filtername>,<Shrink filtername>

Applies only to PDF2AFP conversion.

Specifies filter names to be used for image resizing. The specified filter is used whenever an image has to be resized before writing to the output file.

Allowed are the following filter names:

BiCubic	
BiCubic2	
BicubicSpline	For an accurate image

Bilinear	Slow Algorithm
Lanczos	Best for shrinking images
NearestPixel	Fastest Algorithm
Mitchell	Best for stretching images

Default: Stretch filtername=Mitchell and shrink filtername=Lanczos

6.2.3.37 ImageRotationFilter=<filtername>

Only allowed for output to raster formats.

Specifies the filter to use for image rotation. The specified filter is used whenever an image has to be rotated before writing to the output file.

Default: ImageRotationFilter=None

Allowed are one the following filter names:

None	Use simple pixel by pixel rotation filter
Bessel	
Bicubic	
BiCubic2	
Bilinear	
Blackman	
Box	
Bspline	
Catrom	
Gaussian	
Hamming	
Hermite	

Lanczos	
Mitchell	
Nearest Neighbour	
Quadratic	
Sinc	

6.2.3.38 IndexFormat=<CSV|XML>

Format of the index file to create.

Only if FileCreationMode=DOC_INDEX is specified.

CSV	Index file in CSV format
XML	Index file in XML format (Default)

Default: XML

6.2.3.39 IndexPath=<path>

Path to output index files. If this path is not specified, AFP2web will write the index files to the path specified in OutputFilePath.

Default: pdf

6.2.3.40 IndexRecord=<fieldlist>

Only for IndexFormat=CSV.

A list of keywords for meta information to be added to each index entry. Given in the order, in which each meta field must occur. Each meta field specified must have a delimiter (for example ",").

The following keywords are possible for <fieldlist>:

PAGE_GROUP_NAME	AFP Group name
PAGE_NAME	AFP Page name

PAGE_COUNT	Page count
FILE_NAME	Name of output file
FILE_TYPE	ASCII/JPEG/PDF/TIFF
FILE_SIZE	Size of output files
INDEX	Index as: <IndexName>=<indexValue>

Default:

IndexRecord=PAGE_GROUP_NAME,PAGE_NAME,PAGE_COUNT,FILE_NAME,FILE_TYPE,INDEX

6.2.3.41 **InputFormat=<input format>**

Input format. One of the following is possible:

TIFF	TIFF document
LPD	LPD document
MMD	MMD document

For LPD/MMD input format, File creation mode must be set to "DOC_COLD"

6.2.3.42 **JPEGQuality=<n>**

Compression quality for colored graphics and applies only to the following cases:

- Any format to JPEG conversion
- AFP (IOCA color images) to PDF
- PDF (color images) to AFP (IOCA FS 11)

This value is used while saving object container to disk. Quality can be any value between 1 and 100:

1	Best compression but lowest quality
100	Lowest compression but best quality
50	is a recommended value.

Default: 50

Example: JPEGQuality=50

JPEGQuality is used whenever the image is placed as JPEG in PDF output, for example: For B/W output, color images in the spool file are converted to gray scale JPEG images. For color output, when "CMYKTORGB" is set to "on", IOCA FS45 CMYK JPEG planes are merged as RGB JPEG and place in PDF output.

6.2.3.43 **Keywords=<doc-info>**

Keywords to be stored in the document

Default: See default value in the INI file

6.2.3.44 **LaunchPreview=<option>**

Preview option. Launches Acrobat Reader (for PDF) or the appropriate ASCII, TIFF or JPEG viewer to display the resulting output file.

on	Display resulting output file.
off	Do not display.

Default: off

Works only under Windows if some application (viewer) has been mapped to the file format in the system.

6.2.3.45 **Licensee=<name>**

The name of the licensee as determined by Maas Holding GmbH. Please set the value within quotes.

Example: Licensee="Maas Holding GmbH"

Both SerialNr and Licensee will activate the input/output formats and the platform, which have been purchased as AFP2web options.

Default: See default value in the INI file

6.2.3.46 **Logging=<on/off>**

Log file (Part 1: Parsing-section, Part 2: Builder-section with font information):

on	Create complete log file. Same as LoggingLevel=1,2,3,4,FONT
off	Do not create.

Default: off

If LoggingLevel=0 follows, it will switch off Logging.

6.2.3.47 **LoggingFont=<on/off>**

Log file (Part 2: Builder-section with font information):

on	Create log file, part 2 Same as LoggingLevel=FONT
off	Do not create (Default).

Default: off

If LoggingLevel=0 follows, it will switch off LoggingFont.

6.2.3.48 **LoggingLevel=<n>**

Logging level.

Syntax:

LoggingLevel=1[,2][,3][,4][,5][,6][,8][,SF][,RES|(ResType)+][,INPUT]

Example:

LoggingLevel=1,2, FONT

For <n>, specify one or more values out of:

0	No log (Default)
1	Structured Fields (SFI)
2	Data information of SFI
3	Repeating Groups
4	Patterns (like image)
5	FOCA information
6	Mapping.def entries added to the end of the log file. These entries can be used as template to modify the mapping.def
8	FTP Logging
ALL	All information including FOCA
FONT	Builder part, font information (same as LoggingFont=on)

Resource Logging Levels:

Specify either

RES	for all resource types.
------------	-------------------------

or any of the following resource types:

CDP	Code Pages
FDEF	Formdefs
FNTR	Fonts (detailed list of all fonts used for current spool file)
IOB	Object containers
MMAP	Medium Maps
OLY	Overlays
PSEG	Page Segments

Image Logging Levels

INFO	Log image processing informational messages
WARN	Log image processing warning and informational messages
DEBUG	Log warning, informational and debug messages during image processing

Input Specific Logging Levels

PDFINFO	<p>Logs additional image processing info while parsing PDF input. Additionally logged image processing info will look like as given below</p> <p>Memory Source. Buffer length=>3888, Type=>DIB</p> <p>Source Image Information:</p> <p>=====</p> <p>Type : DIB</p> <p>Width : 160</p> <p>Height : 192</p> <p>Compression : Uncompressed</p> <p>BitsPerPixel : 1</p> <p>BitsPerSample : 1</p> <p>SamplesPerPixel : 1</p> <p>XResolution : 2880</p> <p>YResolution : 2880</p>
INWARN	<p>Logs warning occurred while parsing input spool. Applicable now, only for PDF input spools</p> <p>Warning! Unsupported object occurred in page. Type=Annotation SubType=Screen, A=Rendition, MFN="KlingelSound.mp3", MimeType=audio/mpeg</p> <p>Warning! Unsupported object occurred in page. Type=Annotation SubType=PolygonMemory Source. Buffer length=>3888, Type=>DIB</p>

Output Specific Logging Levels

OUTWARN	<p>Logs warning occurred while writing objects for specified output . Applicable now, only for warnings occurred while writing text objects to AFP output with IS2 Compliance</p> <p>Warning! Draw I Axis Line(PTOCA DIR) Width value "88" is not within valid range(0-40) for Modca-IS2 compliance.</p>
---------	---

Default: 0

If LoggingFont follows LoggingLevel=0, LoggingFont will take effect

6.2.3.49 **LogPath=<path>**

Path to write the log file.

Default: log

6.2.3.50 **MaxCols=<max columns> MaxRows=<max rows>**

Maximum number of columns and rows. For ASCII only. Limits the page size.

Default: If either no value or zero is specified, AFP2web calculates appropriate values.

6.2.3.51 **MaxVectorObjects=<n>**

Maximum vector objects per page. If the number vector objects of page exceeds this value, vector objects are rasterized and then written as image objects on output. A good value to start with is 200, which you modify, if necessary.

The following values, of which one can be entered, have a special meaning:

-1	Write vector objects using graphic primitives on output
0	Write all vector objects as image objects in output.

Default: -1

6.2.3.52 **MemoryOutputStream=<on|off>**

AFP2web will be configured to write the output document to a buffer in memory instead of to file.

For Java SDK/C SDK variants of product, this option is useful to get back the transformed output document.

For Batch variant of product, this option is valid only when used together with the file creation mode "DOC_COLD".

For usage notes, please refer to the Scripting Facility guide and the API reference.

Default: Off (write to file).

6.2.3.53 **ObjectContainerPath=<path>**

Path to object container resources.

It is also possible to specify a comma-separated list of paths.

Example: samples/resource,c:\afp\objects

Default: samples/resource

6.2.3.54 **OutputFilePath=<path>**

Path to write output files.

Default: pdf

6.2.3.55 **OutputFormat=<output format>**

Output format:

AFP	AFP format
ASCII	ASCII format with adjustments to line/column positioning.
JPEG	JPEG format
PDF	PDF format (Default)
PDFA	PDF/A-compliant output.
PNG	Portable Network Graphics
TIFF	TIFF format

When entering -TIFF as command line option:

Adding an extension to the option –TIFF specifies the TIFF compression to use. Possible is one of:

-TIFF:G3

-TIFF:G4

-TIFF:JPG

-TIFF:LZW

-TIFF:PACK

-TIFF:UNCOMPRESSED

In general, specific subformats for an output format can be specified - as for TIFF - either as an extension to the output option separated by colon (for example: -AFP:BW) or as a separate command option (for example, -AFP and -BW).

6.2.3.56 **OutputScale=<Scale to Size>[,<Applying Rule>]**

Re-size the output page size based on "Scale to Size" and applying Rule

Applicable only for RASTER output

Syntax:

OutputScale=<ScaleToSize>[,<ApplyingRule>]

or

-oscale:<ScaleToSize>[,<ApplyingRule>]

where

Scale to Size	Expected page size specified in STANDARD paper sizes like A4, A3 etc. Possible values are A0 to A10, B0 to B10, C0 to C10, 4A0, 2A0, Letter and Ledger . Default is to use input page size AS IS	
Applying Rule	Can be any of following value	
	Value	Description
	gt	Scale input page, when its size is greater than 'ScaleToSize'
	lt	Scale input page, when its size is lesser than 'ScaleToSize'
	ne	Scale input page, when its size is not equal to 'ScaleToSize'
Default is "gt"		

Example:

OutputScale=A4,lt

AFP2web will scale up to A4 if the size of input page is less than A4 size

6.2.3.57 **OutputSize=<width>,<height>**

Size of the resulting output document specified as width and height in pixels. For example, to produce thumbnail images for document pages.

Default: No default - the document original size is used.

6.2.3.58 **OverlayExt=<file extension>**

File extension for Overlay resources.

Default: *

Normally AFP2web looks for <resource name>*, so it is normally not necessary to use this parameter. But if -strict is set and a file extension is used, then AFP2web looks for <resource name>.<file extension>

6.2.3.59 OverlayPath=<path>

Path to overlay resources.

It is also possible to specify a comma-separated list of paths.

Example: samples/resource,c:\afp\overlays

Default: samples/resource

6.2.3.60 PagedefFilename=<Pagedef Filename>

External pagedef filename to be used for the MMD document.

6.2.3.61 PageOutput=<on/off>

Create one output file for each page of an output document. Applies only to TIFF as output format.

Default: off

6.2.3.62 PageRotation=<rotation>

Allowed only for output to AFP, PDF or a raster format. The options "portrait" or "landscape" are only possible for output to AFP.

Page rotation clockwise in degrees or the page orientation for all output pages:

0

90

180

270
portrait
landscape

6.2.3.63 PageSegExt=<file extension>

File extension for Page Segment resources.

Default: *

Normally AFP2web looks for <resource name>*, so it is normally not necessary to use this parameter. But if -strict is set and a file extension is used, then AFP2web looks for <resource name>.<file extension>

6.2.3.64 PageSegmentPath=<path>

Path to Page Segment resources.

It is also possible to specify a comma-separated list of paths.

Example: samples/resource,c:\afp\pagesegments

Default: samples/resource

6.2.3.65 PDFBookmark=<option>

For index elements, inserts bookmarks in output PDF documents.

on	Insert bookmarks
off	Do not insert bookmarks

Default: off

6.2.3.66 PDFDocLimits=<limit>

Applies only to PDF as output format: This parameter is used as a factor for allocating memory (fonts, images, other PDF objects).

We recommend that you increase this limit if you encounter the error message:

Too many images in this PDF: nnn. Increase limits by `mpdf_open()`.

The formula for the amount of memory allocated:

Memory allocated(in bytes) = $18200 * \text{PDFDocLimits} + 6892$

Memory allocated(in Kbytes) = $(18200 * \text{PDFDocLimits} + 6892) / 1024$

The following table gives a few examples of settings and the resulting allocation of memory:

PDFDocLimits	Memory allocated (Kbytes)
20	362
200	3561
500	8893
1000	17780

Default: 200

6.2.3.67 PDFFontHeightFactor=<n.n>

PDF Font Height Factor. Must be greater than or equal to 1.0.

Applies only for type3 fonts embedded in PDF input file.

Default: 1.0

6.2.3.68 PDFParserConfFile=<PDFParserConfFile>

PDF parser configuration filename, used to control/enhance parsing. For example, useful to configure CMAP tables that are used while parsing PDF texts

Default: pdfparser.conf

6.2.3.69 PDFSecurity=[<Owner Password>],[<User Password>],[<Key Length>],[<Permission Flags>]]

PDF Security options.

The parameters are fixed-positioned and have the following meanings:

Owner Password	Optional, Owner Password. Used to change security options
User Password	Optional, User password, required when opening the document.

Key Length	Optional, Encryption Level Key Length. Possible values are 40 and 128 40 => Normal Encryption 128 => Better Encryption Default is 40.
Permission Flags	Optional, One or more flags separated by " " specified below. For Key Length 40, Flags 5,6,7,8 are not allowed. NOTE: To enable, high quality printing, both low quality and high quality flags must be given. PDFSecurity=afp2web,,40,1 ==> Turn on printing @ Low quality PDFSecurity=afp2web,,128,1 8 ==> Turn on printing @ High quality

The following values are allowed as Permission Flags:

0	None, everything is disabled (Default)
1	Allow printing and also set print resolution to low level
2	Allow Changing the document
3	Allow Content copying or extraction
4	Allow Authoring comments and form fields
5	Allow Form field fill-in or signing
6	Allow Content accessibility
7	Allow Document assembly
8	Allow High Quality Printing

Examples:

1. Neither Owner nor User Password (this is the default)

PDFSecurity=

2. Owner Password is MHT, Encryption Key Length is 128 bits, no Permission:

PDFSecurity=MHT,,128,0

6.2.3.70 PDFUIOptions=[<HideMenubar>][,<HideToolbar>][, <HideWindowUI>]

PDF viewer preferences for the "User Interface".

For each parameter, specify either "on" or "off" (without the quotes). By default, all parameters are off.

HideMenubar	on = Hide menu bar.
HideToolbar	on = Hide tool bar.

HideWindowUI	on = Hide window controls.
--------------	----------------------------

An empty value is allowed and default values are assumed for empty values. For example, PDFUIOptions=,on,on is equivalent to PDFUIOptions=off,on,on. OR as a command line parameter: -pui:,on,on is equivalent to -pui:off,on,on.

Examples:

To hide only the menu bar, use PDFUIOptions=on.

To hide menu bar and window controls, use PDFUIOptions=on,,on.

6.2.3.71 PDFWinOptions=[FitWindow][,<CenterWindow>][,<FullScreenWindow>][,<DisplayDocTitle>]

PDF viewer preferences "Window" options

For each parameter, specify either "on" or "off" (without the quotes). By default, all parameters are off.

FitWindow	on = Resize the document's window to fit the size of the first displayed page.
CenterWindow	on = Position the document's window in the center of the screen.
FullScreenWindow	on = Display the document in full-screen mode, with no menu bar, window controls, or any other window visible.
DisplayDocTitle	on = Display the document description property Title in the window's title bar. off = Display PDF filename in the window's title bar.

An empty value is allowed and the default is assumed for empty values. For example, PDFWinOptions=,on,on,on is equivalent to PDFWinOptions=off,on,on,on. Or as command line parameter: -pwin:,on,on,on is equivalent to -pwin:off,on,on,on.

Examples:

To resize the window, use PDFWinOptions=on.

To center window and display in full screen mode, use PDFWinOptions=,on,on.

To display the document title, use PDFWinOptions=,,,on.

6.2.3.72 **PrintableLineWidth=<n>**

Minimum line width that has to be used on output to ensure that lines are printable. Specified in a unit of pixels. Applies when converting to AFP.

Example: PrintableLineWidth=1

Default: 1

6.2.3.73 **Protocol=<on/off>**

The "protocol" is a processing log listing documents converted from spool file. Documents are identified by document ID. This option is only possible if DOC_INDEX or DOC_COLD is selected.

on	Write protocol
off	Do not write.

Default: off

6.2.3.74 **Quiet2Mode=<on/off>**

Suppress messages to system console:

on	No messages, also suppress "Running AFP2web ..." message
off	Show message

Default: off

6.2.3.75 **Quietmode=<on/off>**

Suppress messages to system console:

on	No messages
-----------	-------------

off	Show message
------------	--------------

Default: off

6.2.3.76 **Resolution=<dpi resolution>**

Resolution in dpi. We recommend a value out of:

200	200 dpi
240	240 dpi
300	300 dpi

Allowed only for AFP and RASTER outputs.

Default : 240

6.2.3.77 **ResourceGroup=<option>**

Controls resource group generation for AFP output. Possible is one of the following options:

0	Generate no resource group
1	Generate 1 resource group nevertheless of FileCreationMode
n	Generate n resource group for "DOC", "DOC_INDEX" and "DOC_COLD" file, creation modes and 1 resource group for other file creation modes.

Default: n

6.2.3.78 **ResPath=<path>**

Path to AFP resources.

Default: samples/resource

6.2.3.79 **SaveOCDataOnDisk=<on/off>**

Save Object Container Data on disk.

Flag to cache the object container data temporarily in to file path given by "TempPath". This option helps to save memory and to optimize the output file size.

This option is used for large inline container object (to store it in some temporary file and use this while writing the object) and also for raster formats, such as tiff, jpeg, and bmp.

Default: off

6.2.3.80 **ScaleToSize=<Size>[,<Condition>[,<Tolerance>[,<Aspect Ratio>]]]**

APPLICABLE ONLY FOR PDF INPUT

Specifies expected output page size to which the input page must be scaled when condition match. Tolerance provides flexibility on condition range and Aspect ratio flag allows to retain or not to retain the aspect ratio during scaling.

Syntax:

Table 1 INI file

```
ScaleToSize=<Size>[,<Condition>[,<Tolerance>[,<Aspect Ratio>]]]
```

OR

Table 2 Command line

```
-s2s:<Size>[,<Condition>[,<Tolerance>[,<Aspect Ratio>]]]
```

where

Size	Expected page size specified in standard page sizes like A4, A3 etc. Possible values are A0 to A10, B0 to B10, C0 to C10, 4A0, 2A0, Letter and Ledger. Default is to use input page size AS IS								
Condition	Can be any of following value <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>gt</td><td>Scale down the page if the page size is greater than the specified Size</td></tr> <tr> <td>lt</td><td>Scale up the page if page size is lesser than the specified Size</td></tr> <tr> <td>ne</td><td>Scale the page to the specified Size if the page size is not equal to the the specified one</td></tr> </tbody> </table> Default is "gt"	Value	Description	gt	Scale down the page if the page size is greater than the specified Size	lt	Scale up the page if page size is lesser than the specified Size	ne	Scale the page to the specified Size if the page size is not equal to the the specified one
Value	Description								
gt	Scale down the page if the page size is greater than the specified Size								
lt	Scale up the page if page size is lesser than the specified Size								
ne	Scale the page to the specified Size if the page size is not equal to the the specified one								
Tolerance	Tolerance in percentage that is used to evaluate whether the page is greater, lesser or not equal to the specified size Possible values are 0 to 100. Default is 0								
Aspect Ratio	Aspect ratio flag allows to retain or not to retain the aspect ratio during scaling. Possible values are "on", "off". "on" means keep the aspect ratio. Default is "on"								

Example:

Case 1: ScaleToSize="A4","gt", will force any input page greater than A4 size to A4 size

Case 2: ScaleToSize="A4","lt", will force any input page lesser than A4 size to A4 size

Case 3: ScaleToSize="A4","ne", will force any input page not equal to A4 size to A4 size

6.2.3.81 ScriptArgument=<arguments>

Arguments to pass to the scripting procedure. Used for AFP2web Scripting Facility (only if FileCreationMode=DOC_COLD).

Default: null (no arguments)

6.2.3.82 ScriptProcedure=<scriptfile>

The scripting procedure file name. Used for AFP2web Scripting Facility (only if FileCreationMode=DOC_COLD).

Default: afp2web.pm

6.2.3.83 **ScriptUnitBase=<baseunit>**

Base unit of measurement for the Scripting Facility used to specify/receive position and size parameters. (only if FileCreationMode=DOC_COLD).

Possible one out of:

pixel	Pixels at page resolution
mm	Millimeters

Default: pixel

6.2.3.84 **SerialNr=<serialnr>**

The serial number as determined by Maas Holding GmbH.

Both SerialNr and Licensee will activate the input/output formats and the platform, which have been purchased as AFP2web options.

6.2.3.85 **SkipObjectSize=<n>**

The maximum byte size of an image resulting from a scanned empty page, which can be ignored. Even though a page (for example the back side) can be empty, the scanned image still might contain a few pixels. This parameter specifies the maximum size of this type of image.

Default: 2K (=2048 Bytes)

6.2.3.86 **SkipPage=<on|off>**

Skip empty pages (for example, a scanned empty page). This flag tells AFP2web to ignore a page if it has no content or if it contains one image, which is less or equal to the size specified in **SkipObjectSize**.

Default: off

6.2.3.87 **StartingDocument=<n>**

First document to convert in a spool file. For StartingDocument, enter the document ID as determined in the "protocol".

This option is allowed only if FileCreationMode is DOC_INDEX or DOC_COLD is selected.

Default: 0

6.2.3.88 StartingPage=<startpage>

Page Range to convert (Starting page)

Default: 1

6.2.3.89 Statistic=<on/off>

Write a report with performance statistics (stat*.txt).

on	Write report.
off	No report

Default: on

6.2.3.90 Strict=<on/off>

Stop processing if a required AFP resource is not found.

on	Stop processing if resource not found
off	Continue processing

Default: off

6.2.3.91 Subject=<doc-info>

Document property: Subject

Default: See default value in the INI file

6.2.3.92 TempPath=<path>

Temporary path is used to cache AFP2web temporary files. The temporary path must exist.

AFP2web must have read/write access to the directory specified in this path. Otherwise, AFP2web will fail when trying to create and delete temporary files.

Default: the temporary path defined in the system.

6.2.3.93 TextPresentationFidelity=<char|text>

Controls AFP output text presentation for better fidelity

char	Text is written as character by character when character width table is available, else written as a single text object
text	Text is written as single text object

Example: TextPresentationFidelity=char

Default: char

6.2.3.94 Title=<doc-info>

Document property: Title

Default: See default value in the INI file

6.2.3.95 TransformationProfile=<section name>

Transformation Profile describes options for a specific transformation. Transformation Profiles must be defined in the INI file as sections. Any option that can be specified under the [Settings] section can be used in the Transformation Profile.

Default: None

Command Line Options/SDK API properties will override transformation profile values.

Sample Transformation Profiles

Example 1: TIFF:G4 output generated by applying adaptive threshold

```
[TPR_TIFFG4_BWS_ADAPTIVE]
OutputFormat=TIFF
FormatType=G4
Color=Off
QuietMode=on
CodedOutput=off
BWSampler=Threshold:Adaptive
```

- Set "**TransformationProfile=TPR_TIFFG4_BWS_ADAPTIVE**" on INI file to select this transformation profile during transformation
OR
- Pass "**-tpr:TPR_TIFFG4_BWS_ADAPTIVE**" on command line to select this transformation profile during transformation

Example 2: TIFF:G4 output scaled to A4 size when input page is greater than A4

```
[TPR_TIFFG4_SCALETOA4]
OutputFormat=TIFF
FormatType=G4
Color=Off
QuietMode=on
ScaleToSize=A4,gt
```

- Set "**TransformationProfile=TPR_TIFFG4_SCALETOA4**" on INI file to select this transformation profile during transformation
OR
- Pass "**-tpr:TPR_TIFFG4_SCALETOA4**" on command line to select this transformation profile during transformation

6.2.3.96 **Watermark=<text, typeface,height,rotation,redmask,greenmask,bluemask>**

Displays a text as a watermark behind the document text. The parameters that can be specified are:

text	text to display
typeface	Font typeface
height	Font size in tenths of points. Specify 90 for 9 points.
rotation	Rotation. A value from 0 to 360.
redmask	RGB Red mask value from 0 to 255
greenmask	RGB Green mask value from 0 to 255
bluemask	RGB Blue mask value from 0 to 255

Example:

As an INI parameter: Watermark=Confidential,Helvetica,600,60,189,219,231

or as a command line parameter: -wm:"Confidential,Helvetica,600,60,189,219,231".

Default: no watermark

6.2.4 <formdef>= <medium map-list>

This section is required if the AFP spool file requires a Formdef. The Formdef can be specified using the command line option **-fd** or **-rf**.

The [FORMDEF] section of the INI file can be used to map Medium Maps to Formdef Resource names. This feature is useful, for example, if the command line option -fd is not used. The Formdef itself must be available as external resource.

You use this section to list the medium maps, which are contained in a Formdef. A medium map must be unique to the Formdef. AFP2web will load the first Formdef it can find for a given medium map.

Example:

F1hqsd=HOCH,HOCHD,QUER,QUERD,HFACH1

6.2.5 Section [AFPColorTable]

The section [AFPColorTable] is used to map each color of the AFP standard OCA color table to their corresponding RGB values. You use this section to override the default values (For more information on the default values for each color, please refer to the documentation of IBM, document number SC31-6802-06, page 473/501).

Syntax:

Colorname=<Red>,<Green>,<Blue>[,<Minimum Shading Value>[,<Maximum Shading Value>]]

Where:

<Red>,<Green>,<Blue>	A combination of RGB values. Each RGB value should be in the range 0 to 255. Values outside this range are treated as zero.
<Minimum Shading Value>	Minimum shading value is the limit from which the value is treated as given color, default is 20.
<Maximum Shading Value>	Maximum shading value is the limit up to which the value is treated as given color, default is 235.

Important notes:

1. Minimum and Maximum shading values are used only for GRAY color and for others it is ignored.
2. Minimum and Maximum shading values are used only for PDF2AFP (MODCA IS2 compliance) conversion, when finding nearest AFP color for given PDF object color value

Example:

Blue=0,0,255

Color names and their default RGB values are:

Black	0,0,0
Blue	0,0,255
Brown	144,48,0
Cyan	0,255,255
Darkblue	0,0,170
Darkcyan	0,146,170
Darkgreen	0,146,0
Default	0,0,0
Gray	131,131,131
Green	0,255,0
Magenta	255,0,255
Medium	255,255,255
Mustard	196,160,32
Orange	255,128,0
Purple	170,0,170
Red	255,0,0
White	255,255,255
Yellow	255,255,0

6.2.6 Section [SPOOLFILETYPE]

The section [SPOOLFILETYPE] is used to specify carriage control type, code page to be used, record type and length factors for LPD & MMD input formats

Syntax:

```
DEF=<CCType>,<Codepage>,<LineFormat>,<LineDelimiter>
BODY=_
```

Where:

CCType	ASA (mean American Standards Association carriage control is used to format content) MACHINECODE (mean IBM Machine Code carriage control is used to format content)
Codepage	<Codepage name> ASCII
LineFormat	FIXED (mean record length is fixed and "LineDelimiter" will used as length of fixed records) VAR (mean record could have variable length and one of following is used as record delimiter, also "LineDelimiter" will be used as Max Record Length)

	Record Delimiter	Description
	CR	Carriage Return character will be used as record delimiter that occur within max record length
	LF	Line Feed character will be used as record delimiter that occur within max record length
	CRLF	Consecutive Carriage Return & Line Feed characters will be used as record delimiter that occur within max record length
LineDelimiter	<Length>	
BODY	Not used, reserved for future use	

Example Spool File Types defined in ini file:

```
[ASA_EBCDIC_FIXED_133]
DEF=ASA,1141,FIXED,133
BODY=_
```

```
[ASA_EBCDIC_FIXED_148]
DEF=ASA,1141,FIXED,148
BODY=_
```

The name of the Spool Filetype must be specified either using "SpoolFileType" parameter (C/Java SDK) or be passed through the "-sft" command line parameter (Batch).

Example:

```
afp2web.exe -q -c -lpdin -doc_cold -sp:eyecatcher_lpd.pm -sft:ASA_EBCDIC_FIXED_148
samples\lpdsample.lpd
```

6.2.7 User-defined Sections for Logical Locations

Any arbitrarily named section can be added to the INI file to define the location of remote AFP resources. The following parameters are possible:

[mysectionname]	Use the "[]" Brackets to define your FTP Connection. Any name can be used (for example, [FONT300]). To tell AFP2web to use that FTP connection for a particular AFP font resource, specify the FTP connection name prefixed with an @ in your INI parameters. Example: AFPFontPath=@FONT300
Type=FTP	Communication protocol
HostName=<hostname or IP address>	Host name or IP address
UserName=<username>	FTP user name

Password=<password>	Password
StartDir=<starting directory>	Starting directory Examples: StartDir='SYS1.FONT300' ==> For a mainframe PO data set (Single quotes have to be used) StartDir=c:/afp/font3000 ==> For a Windows FTP server StartDir=/opt/afp/font3000 ==> For a Unix FTP server

You define one section for each location, for example:

```
[MYFONT300]
Type=FTP
HostName=192.168.1.1
UserName=afp2web
Password=*****
StartDir='SYS1.FONT300'
```

6.3 AFP2web Command Line Options

The following description lists the command line options in alphabetic order. For an overview of options grouped by function, please refer to [Overview of INI Parameters and Command Line Options](#).

6.3.1 Usage Notes

When you call afp2web, begin each option with a minus sign or hyphen ("-").

If you do not specify a command line option, AFP2web will look for the setting in the INI file or assume a default option. The sequence is:

- Command line option entered when calling AFP2web
- Parameter in the INI file
- Default value (as described for the INI file)

For options with multiple arguments, it is recommended to enclose the arguments in quotes. This ensures that AFP2web will not terminate the argument string at the first blank character.

Example:

```
-wm:"Confidential Text",Helvetica,600,60,189,219,231
```

Command option	Purpose	See also INI Parameter
-<inputformat>	Format of input documents	InputFormat
-<outputformat>	Target format of output documents	OutputFormat

-acl	Conditions to be applied for AFP output with IS2 compliance	AFPIs2ComplianceFlag
-aof	Conditions to be applied for AFP output	AFPOutputFlags
-ap	Path to AFP font resources.	AFPFontPath
-bm	For index elements, inserts bookmarks in output PDF documents	PDFBookmark
-BW	Convert colors to black/white (text and image)	(none)
-c	Color on or off	Color
-clr	Mapping of AFP standard OCA colors to their corresponding RGB values	Section [AFPCOLORTable]
-cp		CpPath
-csf	Factor used to smooth curves. A value from 1 to 65535	CurveSamplingFactor
-ct	Color tolerance value	ColorTolerance
-dc	Create sub folders for output and limit the number of documents per sub folder.	DocumentCount
-dcp	Default code page to use for translating resource names, NOPs and indexes of the AFP spool to ASCII	CodePage
-DOC / ALL /	Output files to produce	FileCreationMode
-DOC_COLD /	Activate the AFP2web Scripting Facility. Index files must be written by the Scripting Facility routines.	FileCreationMode=DOC_COLD
-DOC_COLD: XML CSV	AFP2web writes index file in CSV or XML format.	IndexFormat
-DOC_INDEX /	Create output documents with index files.	FileCreationMode
-DOC_INDEX: CSV XML	Create output documents with index files and specify the format of the index file to create-	IndexFormat
-DOC_MERGE /	Output files to produce	FileCreationMode
-ed	Doc ID, stop converting after this document in spool file.	EndingDocument
-ell	Print error messages in a selected level of detail.	ExceptionLoggingLevel
-fd	AFP FORMDEF	(none)
-fdp	Path to FORMDEF resources	FormDefPath
-flush	Caching of output to either memory or file (on/off) .	Flush
-fnpt	Filename pattern (without extension)	FilenamePattern
-fp	Font path to additional Type1 and TrueType fonts.	ExtFontPath
-h or - ?	Help info about AFP2web (list of command line options)	(none)
-iASCII	Display optimal values for the options–mc and –mr	(none)
-if	LOG output: font infos	LoggingFont

-imcs	Maximum memory cache size for image, specified in Megabytes.	ImageMaxCacheSize
-ip	Path to afp2web.ini	(none)
-ipv	Specifies default values for missing input image properties like resolution.	ImageProcessValues
-irf	Filter to use for rotating images	ImageRotationFilter
-isf	Filters to be used for image stretching and shrinking	ImageResizeFilter
-jq:n	Jpeg Quality where n can be 1 to 100	JPEGQuality
-l	LOG output	Logging
-lf	LOG output: font infos	LoggingFont
-ll	Restrict LOG output to information categories.	LoggingLevel
-lp	Folder for LOG output	LogPath
-mc	Maximum columns per row	MaxCols
-mos	Write output document to a buffer in memory	MemoryOutputStream
-mr	Maximum rows per page	MaxRows
-mvo	Maximum number of vector objects	MaxVectorObjects
-ncm	Redirect messages to file	(none)
-nco	Retain text encoding or not	CodedOutput
-nocmyktorgb	Do not convert IOCA FS45 color images output to PDF	CMYKTORGB
-ocp	Path to object container resources	ObjectContainerPath
-op	Path to write output files.	OutputFilePath
-os	Size of the resulting output document specified as width and height in pixels.	OutputSize
-ovp	Path to overlay resources.	OverlayPath
-p	Write „protocol“ with Doc IDs	Protocol
-PAGE	One file per page in an AFP document.	FileCreationMode
-pfh	PDF Font Height Factor. Must be greater than or equal to 1.0.	PDFFontHeightFactor
-plw	Minimal line width for printing	PrintableLineWidth
-po	Create one file per page of the output document (only for TIFF as output format)	PageOutput
-pp:[fp][-tp]	Page range to convert (dstarting page) (ending page)	StartingPage / EndingPage
-pr	Resolution	Resolution
-ps	PDF Security	PDFSecurity

-psp	Folder for AFP Page Segment resources	PageSegmentPath
-pui	Viewer settings	PDFUIOptions
-pv	Display converted document in viewer	LaunchPreview
-pwn	Document display settings	PDFWinOptions
-q	Suppress processing information on the console	Quietmode
-r	Page rotation	PageRotation
-rf	AFP resource file	(none)
-rgp	Controls the generation of resource groups	ResourceGroup
-rp	Standard folder for AFP resources	ResPath
-s	Write a statistic report (stat*.txt)	Statistic
-sa	Arguments to pass to the script	ScriptArgument
-sd	Doc ID, start converting at this position in spool file.	StartingDocument
-sft	Option to specify the name of spool type section that has to used while processing given LPD/MMD input spool	(custom named section)
-si	Read from stdin	(none)
-so	Write to stdout	(none)
-sod	Save Object Container Data on disk	SaveOCDataOnDisk
-sp	Name of the script file.	ScriptProcedure
-std	Both options: -si and -so	(none)
-strict		Strict
-s2s	ScaleToSize, specifies expected output page size to which the input page must be scaled when condition match.	ScaleToSize
-TIFF: compressionformat or just: -compressionformat	TIFF format	FormatType
-tp	Temporary path is used to cache AFP2web temporary files. Default is the temporary path defined in the system. Temporary Path should exist.	TempPath
-u	Generate unique file names	GenerateUniqueFile
-usid	Generation of a system key	(none)
-v	Display version of AFP2web	(none)
-vall	Display versions of all AFP2web components	(noen)
-wm	Watermark	Watermark

-xf	The input file with the AFP index to be processed for the AFP document	(none)
-xp	Folder for writing index files.	IndexPath

6.4 Parameters of the mapping.def

6.4.1 Function and Structure

The file mapping.def is a text file located in the code pages folder afpcp/ created during installation. This file is used to control font mapping.

Before we describe the details, here is the overview of the sections and their general purpose:

Specifies the mode for processing fonts:

[CHARSET RENDERING]	Specifies the font processing mode for each AFP Character Set (Font Rasterizing, Substitution/Referencing, or Embedding).
---------------------	---

The sections, which define defaults for AFP fonts. Entries are only required if AFP2web cannot derive the information from the AFP input:

[XFONT]	AFP Character Set, AFP Code Page
[CHARSET]	AFP Character Set, Font Global ID
[FGID]	Font Global ID, AFP font name

In the following sections, each AFP font name is mapped to the respective substitution font. You can specify a list of fonts. AFP2web uses these as alternatives if it cannot find a particular font on the system:

[INPUT FONT ALIASES]	Font-Aliases to be used for the conversion of PDF input
[PDFFONT]	Font to be used for conversion to PDF.
[WINFONT]	Fonts to be used for conversion to raster output (TIFF/JPEG/...). These fonts must be installed in the Windows system
[UNIXFONT]	Fonts to be used for conversion to raster output (TIFF/JPEG/...). These fonts must be installed in the Unix system
[FONTSUFFIX]	Naming convention for font files.

Specifies the font to use for output to AFP:

[AFPFONT]	Maps PC fonts to AFP character sets and codepages
-----------	---

The following section maps codes pages and character sets:

[CODEPG]	Maps AFP code pages to code page files
----------	--

The following section maps GCSGID and FGID to an AFP character set:

[GCSGID]	Maps GCSGID and FGID to an AFP character set
----------	--

6.4.2 Usage Notes

You customize this file by adding your entries to the appropriate sections under ";user-defined". Doing this overrides the entries under ";standard". If you define rules, the first rule which applies will be executed, all succeeding are ignored.

Please note the following when you change the mapping.def:

- A comment line begins with a semicolon (";") as its first character.
- Each section has entries under "user defined" and "standard". An entry under "user defined" will override an existing entry under "standard". **Important: Do not change any entries under „standard“!**
- If you make the AFP font resources available to AFP2web, entries in the sections [XFONT], [CHARSET], [FGID] are no longer necessary.
- Any entry in the mapping.def must not exceed 1024 (1 Kb) characters.

6.4.3 [CHARSET RENDERING] Section

6.4.3.1 Syntax, Example

Syntax:

charsetname=RenderingFlag,option

Example:

COH000A0=2

6.4.3.2 Function

Used to specify processing flags for AFP character sets.

6.4.3.3 Parameters

Parameter	Description
charsetname	AFP Character Set. If required, the AFP Character Set is defined in the section [CHARSET] of the mapping.def. Note: Wild card characters are allowed while specifying character set names in [CHARSET RENDERING] section of mapping.def (Please see the usage notes below).
RenderingFlag	One of the following values:

	0	Font Rasterizing (Default) Font processing is based on the original fonts. This is the default font processing mode.
	1	Font Referencing/Substituting (Use Substitution Font) <ul style="list-style-type: none"> • PDF: Reference the 14 standard fonts for PDF or the supported external font • TIFF: Force writing text using the substitution font
	2	Font Embedding (Use Substitution Font) <ul style="list-style-type: none"> • PDF: Embed the substitution font into the PDF file • TIFF: Not applicable
option	Additional option for the conversion from PDF to AFP. Only valid for RenderingFlag=0. Note: Options annotated as "for future use" are planned for a later version of AFP2web. A value out of:	
	0	Use original font (default). AFP2web converts PDF fonts as they are provided from the PDF input. Type 3 fonts are converted to AFP raster fonts for future use: Type 1 and TrueType fonts are converted to AFP outline fonts
	1	Inline Image: AFP2web converts and rasterizes PDF Type 1 and TrueType fonts and creates IOCA images.
	2	for future use: AFP raster font: AFP2web converts Type 1 and TrueType fonts to AFP raster fonts.
	3	for future use: AFP outline font: AFP2web converts Type 1 and TrueType fonts to AFP outline fonts.

6.4.3.4 Usage

When using the rendering flag 0 for Font Rasterizing (this is the default and uses the original font), the result when converting to PDF will be an embedded font subset.

We recommend using a more recent version of the Adobe Acrobat Reader (at least Version 7) to view PDF files.

Using Wildcards for Character Sets

The following wild card characters are allowed while specifying character set names in the [CHARSET RENDERING] section of mapping.def:

Wild card character	Meaning
*	Match 1 or more sequence of characters

?	Match exactly one character
---	-----------------------------

Note: Please use wildcards carefully. AFP2web processes the entries in the section [CHARSET RENDERING] sequentially, top-down. The first match will take effect.

Example:

```
[CHARSET RENDERING]
;Syntax: <CharsetName>=<RenderingFlag>
; RenderingFlag => Values
; 0 => Rasterize
; 1 => Reference / Substitute
; 2 => Embed
;
C?H400B0=1
C?H400*=2
```

Result:

- **Case 1:** The character set “C0H400B0” matches both patterns “C?H400B0=1”, i.e. “C<Any character>H400B0” and “C?H400*=2”, i.e. “C<Any character>H400<Any string>”. AFP2web will take the first pattern with the rendering flag 1.
- **Case 2:** The character set “C4H40090” matches the second pattern “C?H400*=2”, i.e. “C<Any character>H400<Any string>”. AFP2web will take the second pattern with the rendering flag 2.

6.4.4 [XFONT] Section

6.4.4.1 Syntax, Example

Syntax:

xfont = character set, codepage

Example:

X0GB02=C0D0GB10,T1V10273

6.4.4.2 Function

Specifies the AFP character set and the AFP code page for a coded font.

6.4.4.3 Parameters

Parameter	Description
xfont	Coded font (See the notes below.)
character set	AFP Character Set (See the notes below.) This AFP Character Set must be defined in the section [CHARSET] of the mapping.def.
codepage	AFP code page This AFP code page must be defined in the section [CODEPG] of the mapping.def.

Notes on xfont, character set

The second character in the name of a coded font or character set indicates either raster font (either 0 or 1) or outline font (only Z). For example, a raster font is indicated if the name of a coded font begins with X0... or X1..., resp. the name of a character set begins with C0... or C1...

The following rules apply:

- If the second character is ?, AFP2web looks only for a raster font - not for an outline font.
- If the second character is either 0 or 1 (raster font), AFP2web will look for the defined value. If not found AFP2web will also look for the alternative value 0 or 1.
- The second character must be Z, if you want AFP2web to look for an outline font.

6.4.4.4 Usage

You need to add entries to this section if you use your own coded fonts or if you modify existing coded fonts.

Please put your entries under "user-defined". Do not change any entries under "standard".

The result in PDF will be an embedded font subset.

We recommend using an up-to-date version of the Adobe Acrobat Reader (Version 7) to view PDF files.

6.4.5 [CHARSET] Section

6.4.5.1 Syntax, Example

Syntax:

charset = fgid, height, width, strikeover, underline

Example:

COD0GB10=39,120,144,0,0

6.4.5.2 Function

Assigns AFP character set to a global font identifier. Sets additional font attributes (height, width).

6.4.5.3 Parameters

Parameter	Description
charset	AFP Character Set
fgid	Font Global ID Must be defined in the section [FGID]

height	Average height in tenths of a point A value within the range 1 to 999 Example: Enter 90 to specify 9 points.
width	Average width in tenths of a point. A value within the range 1 to 999 Example: Enter 90 for a width of 9 points. (This parameter is reserved for future use.)
strikeover	Draws a horizontal line through the middle of the character 0 no 1 strikeover
underline	Underlines the character. 0 no 1 underline

6.4.5.4 Usage

You need to add entries to this section if you use your own AFP character sets or if you modify existing AFP character sets.

Please put your entries under "user-defined". Do not change any entries under "standard".

6.4.6 [FGID] Section

6.4.6.1 Syntax, Example

Syntax:

fgid = logicalname, style, weight, italic

Example:

39 = GOTHIC ,MODERN,BOLD,0

6.4.6.2 Function

Assigns a font global identifier to a logical font name. Specifies additional font attributes (weight, italic).

6.4.6.3 Parameters

Parameter	Description
fgid	Font Global ID
logicalname	AFP logical font name. Must be defined in the section [PDFFONT], resp. [WINFONT] or [UNIXFONT].

style	Style:	
	SWISS	Proportional, without Serifs
	ROMAN	Proportional, with Serifs
	SCRIPT	mono-spaced, decorative 'handwriting'
	MODERN	mono-spaced, with/without Serifs
	DISPLAY	decorative Only used for TIFF under Windows
weight	Weight:	
	BOLD	bold
	MED	normal
	LIGHT	thin
	Note: For the current release of AFP2web, only BOLD or MED are valid.	
italic	0	straight, no italic
	1	italic (oblique)

6.4.6.4 Usage

Please put your entries under "user-defined". Do not change any entries under "standard".

6.4.7 [FONTSUFFIX] Section

6.4.7.1 Syntax, Example

Syntax:

```
<Style>=<suffix1>[,<suffix2>..,<suffixN>]
```

Example:

BoldItalicSuffix=bi,b,-BoldItalic,-BoldOblique

6.4.7.2 Function

Used to specify suffixes added to the name of font files.

Because file naming conventions vary for fonts, AFP2web will assume a default file naming convention. Suffixes are normally added to the file names of substitution fonts and are normally taken from additional font attributes, such as bold, italic. You can use this section to define the suffixes which are typically used on your system.

Note: Suffixes are case-sensitive under Unix.

6.4.7.3 Parameters

Parameter	Description
Style	One of the following values: BoldSuffix ItalicSuffix BoldItalicSuffix

6.4.7.4 Usage

For example, for the font name: Verdana with the font attributes: BoldItalic, we specify BoldItalic-Suffix=b,-BoldItalic. Based on this definition, AFP2web first looks for Verdanab.* and then for Verdana-BoldItalic.* in the specified font directories (pdffont\ as default). If there are more than one file (such as Verdana-BoldItalic.pfm, Verdana-BoldItalic.ttf), the first file found is used.

6.4.8 [INPUT FONT ALIASES] Section

6.4.8.1 Syntax, Example

Syntax:

<FontFamilyName>=<Alias_1>[,...,<Alias_N>]

Example:

ArialBold=helveticab

6.4.8.2 Function

Used to map PDF input fonts to external font files.

6.4.8.3 Parameters

Parameter	Description
FontFamilyName	Font family name
Alias_1, Alias_2,...,	Alias_n Name of the external font file

6.4.8.4 Usage

This section is used for the conversion of PDF to AFP.

Wildcards can be used for FontFamilyName:

Wildcard	Meaning
*	Zero, one or many characters
?	Exactly one character

Wildcard examples:

```
Arial*Bold*=helveticab  
Arial*Fett*=helveticab  
Arial*Italic*=helveticai  
Arial*Oblique*=helveticai
```

Note: The use of wildcards is not allowed for alias names.

Please put your entries under "user-defined". Do not change any entries under "standard".

6.4.9 [PDFFONT] Section

6.4.9.1 Syntax, Example

Syntax:

logicalname = Font, Alias_1, Alias_2,..., Alias_n

Example:

GOTHIC=Gothic,GothicText,CourierNew

6.4.9.2 Function

This section controls the font substitution for the conversion to PDF. The section [WINFONT] or [UNIXFONT] controls the font substitution for the conversion to TIFF.

Assigns fonts and alternative fonts to an AFP logical font name.

AFP2web searches through this list from left to right and takes the first font it can find in the system.

6.4.9.3 Parameters

Parameter	Description
logicalname	AFP logical font name References a value in the section [FGID]
Font	Font name (PostScript Type 1 font).
Alias_1, Alias_2,..., Alias_n	Additional fonts to look for.

6.4.9.4 Usage

Please put your entries under "user-defined". Do not change any entries under "standard".

Specific entries are required to reference the 14 standard fonts for PDF. Other Postscript Type 1 fonts can be either embedded or referenced. The fonts must be provided and they must have specific file names.

6.4.10 [WINFONT] Section

6.4.10.1 Syntax, Example

Syntax:

logicalname = Font, Alias_1, Alias_2,..., Alias_n

Example:

GOTHIC=Gothic,Gothic Text,Courier New

6.4.10.2 Function

This section controls the font substitution for the conversion to TIFF under Windows.

Assigns fonts and alternative fonts to an AFP logical font name.

AFP2web searches through this list from left to right and takes the first font it can find in the Windows system.

6.4.10.3 Parameters

Parameter	Description
logicalname	AFP logical font name References a value in the section [FGID]
Font	Name of a font family. Take the name as displayed in the Windows control panel "fonts".

Alias_1, Alias_2,...,Alias_n	Additional fonts to be used as alternatives.
------------------------------	--

6.4.10.4 Usage

Please put your entries under "user-defined". Do not change any entries under "standard".

6.4.11 [UNIXFONT] Section

6.4.11.1 Syntax, Example

Syntax:

logicalname = Font, Alias_1, Alias_2,..., Alias_n

Example:

GOTHIC=gothic,courier

6.4.11.2 Function

This section controls the font substitution for the conversion to TIFF under Unix.

Assigns fonts and alternative fonts to an AFP logical font name.

AFP2web searches through this list from left to right and takes the first font it can find in the Unix system.

6.4.11.3 Parameters

Parameter	Description
logicalname	AFP logical font name References a value in the section [FGID]
Font	Name of a font family.
Alias_1, Alias_2,...,Alias_n	Additional fonts to be used as alternatives.

6.4.11.4 Usage

Please put your entries under "user-defined". Do not change any entries under "standard".

6.4.12 [AFPFONT] Section

6.4.12.1 Syntax, Example

Syntax:

Mapping to AFP Raster Font:

<Typeface name>[#<Height>][#<Weight>[#<Slant>]]=<AFP Raster Coded Font Name>|<AFP Raster Character Set Name>,<AFP Code Page Name>

Mapping to AFP Outline Font:

<Typeface name>[#<Weight>[#<Slant>]]=<AFP Outline Coded Font Name>|<AFP Outline Character Set Name>,<AFP Code Page Name>

Example:

```
; Mapping to AFP raster fonts
Arial#100=C0H20000,T1001141
Arial#100##Italic=C0H30000,T1001141
Arial#100#Bold=C0H40000,T1001141
Arial#100#Bold#Italic=C0H50000,T1001141
```

```
; Mapping to AFP outline fonts
Arial=CZH200,T1001141
Arial##Italic=CZH300,T1001141
Arial#Bold=CZH400,T1001141
Arial#Bold#Italic=CZH500,T1001141
```

6.4.12.2 Function

This section is used for converting to AFP.

Specifies for each typeface the substitution font to use for AFP.

6.4.12.3 Parameters

Parameter	Description
Typeface name	Typeface name of font
Height	Value of font height (in tenths of points)
Weight	Font weight like Bold or Light
Slant	Font italic or not, possible values are Italic or Oblique

6.4.12.4 Usage

Please put your entries under "user-defined". Do not change any entries under "standard".

6.4.13 [CODEPG] Section

6.4.13.1 Syntax, Example

Syntax:

codepage = cpgid, wincp

Example:

T1V10273=273,ANSI

6.4.13.2 Function

Assigns an AFP code page to a Windows ANSI code page.

6.4.13.3 Parameters

Parameter	Description
codepage	AFP code page
cpgid	Name of the code page (specify the file name without the file extension ".cp") For ASCII-encoded AFP data enter 819 to specify the standard map file 819.cp
wincp	One of the Windows character sets: ANSI Standard character set SYMBOL NULL Built-in encoding

6.4.13.4 Usage

If you create your own AFP code pages or if you modify existing AFP code pages you must:

- Create the respective code page (<codepage ID>.cp) and place this file into the code page directory (afpcp/) used by AFP2web. Please consult Maas Holding GmbH for more information.
- Add a new entry in the section [CODEPG] of the MAPPING.DEF.

Please put your entries under "user-defined". Do not change any entries under "standard".

6.4.14 [GCSGID] Section

6.4.14.1 Syntax, Example

Syntax:

gcsgid[,fgid] = character set

Example:

2023,40=C1DOGT10
2041,2304=C1H20000
2041,2305=C1H40000
DEFAULT=C1H20000

6.4.14.2 Function

This section is used for mapping AFP MOD:CA IS1/IS2 Spool that map font resources using GCSGID and FGID instead of AFP coded font name or AFP character set/AFP code page names.

A separate entry is made under this section for each GCSGID, FGID pair used in AFP MOD:CA IS1/IS2 spool

6.4.14.3 Parameters

Parameter	Description
GCSGID	Global Character Set Global ID
FGID	Font Global ID
Character Set	Name of AFP character set font

6.4.14.4 Usage

When AFP spool map font using GCSGID and FGID on "Map Coded Font" section of page, then user must enter corresponding entries under this section. Based on the entry under this section, AFP2web will detect the equivalent AFP character set resource to be used for that particular GCSGID and FGID entry. Upon identifying the AFP character set name, AFP2web will further continue to map other details from other sections as given below

- [CHARSET] -- Character set name fetched from GCSGID section is used as key here and details from [CHARSET] section is loaded
- [FGID] -- Font global id fetched from CHARSET section is used as key here and details from [FGID] section is loaded
- [PDFFONT] -- Typeface name fetched from FGID section is used as key here and details from [PDFFONT] section is loaded
- [CODEPG] -- Code page name is fetched based on CP file name on right hand side (i.e. Code page name on left hand side is fetched)

Please put your entries under "user-defined". Do not change any entries under "standard".

6.5 Code Page Files (CP Files)

A code page file (CP file) maps each Graphic Character Global Identifier (GCGID) to an EBCDIC, ASCII and Unicode code point. These mappings are used for the following cases:

- Unicode is used when converting original fonts from AFP to PDF. Encoding makes PDF files available for a full-text search.
- ASCII is the encoding used by the Scripting Facility when parsing AFP resource names, AFP index elements, and is used as the base encoding for the Scripting Facility.
- When substituting fonts, codes are mapped properly to preserve the textual data. For example, AFP fonts have EBCDIC codes, PC fonts normally have ASCII codes. The CP files map EBCDIC to ASCII code points.
- Your AFP font resources might not use the standard Graphic Character Global Identifiers (GCGIDs). Thus, the CP files can also be used to define the logical AFP names for the characters in your AFP Character Set and their character codes.

You add a new custom CP file only if you need to define custom code pages or if your AFP data does not use the standard IBM names to identify characters in your AFP character sets (using the Graphic Character Global Identifiers GCGID).

The CP files have an additional column, which you can use to map a GCGID to the name of a particular glyph in your font. The glyph name is the one defined in the font file.

To clarify the character codes of your fonts, please consult your font vendor.

The format of CP file (which uses the file extension *.cp):

```
<GCGID> <EBCDICValue> <UNICODEValue> <ASCIIValue>[<GlyphName>]
```

Where

- GCGID is the name used to uniquely identify the character.
- The EBCDIC value and the ASCII value are given as two digit hexadecimal values.
- The Unicode value is given as 4 digit hexadecimal value.
- (Optional) The glyph name is the one defined in the font file.

Example:

```
LA020000 C1 0041 41
LB020000 C2 0042 42
LC020000 C3 0043 43
; ohungarumlaut
LO250000 CF 0151 F5 ohungarumlaut
```

If Unicode values are missing in a code page file (CP file), AFP2web takes the default from `gcid2unicode.def`. This file is located in the code pages folder `afpcp`, which is created during installation. `gcid2unicode.def` maps IBM standard GCGIDs to Unicode. For a definition of GCGIDs, please refer to the Technical Reference for IBM Expanded Core Fonts (S544-5228-01).

6.6 Output Logging

AFP2web allows to log output information using the INI-parameter **LoggingLevel=ALL** or the command line option -**ll:all** to produce a detail of the current spool and output files.

6.6.1 Page Logging

6.6.1.1 Format of the Page Logging Message

Information about the page is logged as shown in the below:

```
==>BPG Name=<Page Name>, Size=(W:<Value>[<Value>mm], H:<Value>[<Value>mm])  
==>EPG Name=<Page Name>
```

6.6.1.2 Page Logging Example

```
==>BPG Name=00000001, Size=(W:12240[215.90mm], H:15840[279.40mm])  
==>EPG Name=00000001
```

6.6.1.3 Codes used in the page logging messages

The codes used in the page logging messages and their meanings:

Code	Description
BPG	"BPG" means Begin Page
EPG	"EPG" means End Page
W	Size of the Page Width and corresponding mm value.
H	Size of the Page Height and corresponding mm value.

6.6.2 AFP Resource/Data Object Logging

This session explains about the logging of AFP Resources (like pagesegment, overlay) and Data Objects (like GOCA, Barcode)

6.6.2.1 Page Segment Logging

Format of the AFP Page Segment Logging Message

Information about the AFP page segment resource is logged based on below syntax:

```
==>BPS Name=<Pagesegment Name>, Pos=(X:<value>[<value>mm], Y=<value>[<value>mm]), Type=<Types>
==>EPS Name=<Pagesegment Name>
```

AFP Page Segment logging example

```
==>BPS Name=S1ISLOGO, Pos=(X:7329.00[129.28mm], Y=648.00[11.43mm]), Type=I
==>EPS Name=S1ISLOGO
```

Codes used in the AFP page segment logging messages

The codes used in the AFP page segment logging messages and their meanings:

Code	Description
BPS	Name of the Begin PageSegment
EPS	Name of the End PageSegment
X	X Position of the PageSegment value and corresponding mm value.
Y	Y Position of the PageSegment value and corresponding mm value.
Type	Where <Types> are <ul style="list-style-type: none"> •I - Page/Overlay included resource •SF - Scripting Facility included resource •M - Medium map included resource

6.6.2.2 Overlay Logging

Format of the AFP Overlay Logging Message

Information about the AFP overlay resource is logged based on below given syntax:

```
==>BMO Name=<Overlay name>, Size=(W:<Value>[<Value>mm], H:<Value>[<Value>mm]),
Pos=(W:<Value>[<Value>mm], H:<Value>[<Value>mm]), Type=I
==>EMO Name=<Overlay name>
```

AFP Overlay Logging Example

```
==>BMO Name=01MEDF01, Size=(W:12189[215.00mm], H=15840[279.40mm]), Pos=(X:0.00[0.00mm],
Y=0.00[0.00mm]), Type=I
==>EMO Name=01MEDF01
```

Codes used in the AFP overlay logging messages

The codes used in the AFP overlay resource logging messages and their meanings:

Code	Description
BMO	Name of the Begin Overlay
EMO	Name of the End Overlay
W	Size of the overlay width and corresponding mm value
H	Size of the overlay height and corresponding mm value
X	X Position of the Overlay value and corresponding mm value.
Y	Y Position of the Overlay value and corresponding mm value.
Type	<p>Where <Types> are</p> <ul style="list-style-type: none"> • I - Page/Overlay included resource • SF - Scripting Facility included resource • M - Medium map included resource

6.6.2.3 AFP Graphics Object Logging

This session explains about the AFP GOCA object logging information for line, arc, box, etc.,

Format of the GOCA Logging Message

GOCA Logging

Information about the AFP GOCA is logged based on following syntax:

```
==>BGR Pos=(X:<value>, Y:<value>)
==>EGR
```

Input Line Logging

Information about the input format of GOCA Line logging starts with the character sequence "==" as shown below:

```
==>GLine Orn=<HLine|VLine>, Width=<Value>[<Value>mm], Length=<Value>[<Value>mm],
StartPos=(X:<Value>[<Value>mm], Y:<Value>[<Value>mm]),
EndPos=(X:<Value>[<Value>mm], Y:<Value>[<Value>mm]), LT=<Line Type>
```

Output Line Logging

Information about the output format of GOCA Line logging starts with the character sequence "-->" as shown below:

```
-->GLine Orn=<HLine|VLine>, Width=<Value>[<Value>mm], Length=<Value>[<Value>mm],
StartPos=(X:<Value>[<Value>mm], Y:<Value>[<Value>mm]),
EndPos=(X:<Value>[<Value>mm], Y:<Value>[<Value>mm]), LT=<Line Type>
```

Input Arc Logging

Information about the input format of GOCA Arc logging starts with the character sequence "==>", as shown below:

```
==>Arc T=%s, CP=(%0.2f[%0.2fmm],%0.2f[%0.2fmm]), Hr=%d[%0.2fmm], Vr=%d[%0.2fmm], Pattern=%s, Start
Angle = %d, End Angle = %d
```

Output Arc Logging

Information about the output format of GOCA Arc logging starts with the character sequence "-->", as shown below:

```
-->Arc T=%s, CP=(%0.2f[%0.2fmm],%0.2f[%0.2fmm]), Hr=%d[%0.2fmm], Vr=%d[%0.2fmm], Pattern=%s, Start
Angle = %d, End Angle = %d
```

Input Box Logging

Information about the input format of GOCA Box logging starts with the character sequence "==>", as shown below:

```
==>Box T=%c, FP=(%0.2f[%0.2fmm], %0.2f[%0.2fmm]), DP=(%0.2f[%0.2fmm],%0.2f[%0.2fmm]), LT=%s[,
XR=(%0.2f[%0.2fmm]), YR=(%0.2f[%0.2fmm])]
```

Output Box Logging

Information about the output format of GOCA Box logging starts with the character sequence "-->", as shown below:

```
-->Box T=%c, FP=(%0.2f[%0.2fmm], %0.2f[%0.2fmm]), DP=(%0.2f[%0.2fmm],%0.2f[%0.2fmm]), LT=%s[,
XR=(%0.2f[%0.2fmm]), YR=(%0.2f[%0.2fmm])]
```

Input Fillet Logging

Information about the input format of GOCA Fillet logging starts with the character sequence "==>", as shown below:

```
==>Fillet, SP=(%0.2f[%0.2fmm], %0.2f[%0.2fmm]), EP=(%0.2f[%0.2fmm],%0.2f[%0.2fmm]), LT=%s
```

Output Fillet Logging

Information about the output format of GOCA Fillet logging starts with the character sequence "-->", as shown below:

```
-->Fillet, SP=(%0.2f[%0.2fmm], %0.2f[%0.2fmm]), EP=(%0.2f[%0.2fmm],%0.2f[%0.2fmm]), LT=%s
```

Input Marker Logging

Information about the input format of GOCA Marker logging starts with the character sequence "==>", as shown below:

```
==>Marker, Name=%s, Pos=(%0.2f[%0.2fmm], %0.2f[%0.2fmm]), CW=%0.2f[%0.2fmm], CH=%0.2f[%0.2fmm]
```

Output Marker Logging

Information about the output format of GOCA Marker logging starts with the character sequence "-->", as shown below:

```
-->Marker, Name=%s, Pos=(%0.2f[%0.2fmm], %0.2f[%0.2fmm]), CW=%0.2f[%0.2fmm], CH=%0.2f[%0.2fmm]
```

Codes Used in the GOCA Logging Messages

The codes used in the GOCA logging messages and their meanings:

Code	Description		
BGR	"BGR" means Begin Graphics Object		
EGR	"EGR" means End Graphics Object		
Arc	Identify the logging for Arc		
Box	Identify the logging for Box		
X	X Starting/Ending Position and corresponding mm value.		
Y	Y Starting/Ending Position and corresponding mm value.		
Orn	Line Orientation whether Vertical/Horizontal line (VLine/HLine)		
Width	Line width and corresponding mm value		
Length	Line Length and corresponding mm value		
LT	Line dash pattern and possible values are <ul style="list-style-type: none">•Default•Dotted•Short Dashed•Dash Dot•Double Dot•Long Dash•Dash Double Dot•Solid•Invisible		
T	Type based on object rendered		
	Object	Type	Description
	Arc	F	Full Arc
		P	Partial Arc

	Box	R	Rounded Corner Box
		S	Square Corner Box
CP	Arc center point		
Hr	Horizontal Radius of Arc and corresponding mm values		
Vr	Vertical Radius of Arc and corresponding mm values		
FP	First corner point of Box		
DP	Diagonal corner point of Box		
SP	Line Starting point		
EP	Line Ending point		
CW	Marker Cell width		
CH	Marker Cell height		
Angle	Arc Start/End angle		
X Len	Box Rounded Corner X Radius and corresponding mm value		
Y Len	Box Rounded Corner Y Radius and corresponding mm value		

GOCA Logging Example

For GOCA Logging Example

Information about the GOCA logging example:

```
==>BGR Pos=(X:5380.00, Y=1584.00)
==>EGR
```

For Line Logging Example

Information about the line logging example:

```
==>GLine Orn=VLine, Width=72.00[1.27mm], Length=8838.00[155.89mm], StartPos=(X:5652.00[99.69mm],
Y:1134.00[20.00mm] ), EndPos=(X:5652.00[99.69mm], Y:9972.00[175.89mm] ), LT=Default
-->GLine Orn=VLine, Width=72.00[1.27mm], Length=8838.00[155.89mm], StartPos=(X:5652.00[99.69mm],
Y:1134.00[20.00mm] ), EndPos=(X:5652.00[99.69mm], Y:9972.00[175.89mm] ), LT=Default
```

For Full Arc Example

Information about the GOCA full arc logging example:

```
==>Arc T=F, CP=(2880.00[50.80mm],2880.00[50.80mm]), Hr=1003[17.69mm], Vr=-1003[-17.69mm],  
Patterns=Solid  
-->Arc T=F, CP=(2880.00[50.80mm],2880.00[50.80mm]), Hr=1003[17.69mm], Vr=-1003[-17.69mm],  
Patterns=Solid
```

For Partial Arc Example

Information about the GOCA partial arc logging example:

```
==>Arc T=P, CP=(7200.00[127.00mm],2880.00[127.00mm]), Hr=720[12.70mm], Vr=-720[-12.70mm],  
Pattern=Long Dash, Start Angle = 315, End Angle = 135  
-->Arc T=P, CP=(7200.00[127.00mm],2880.00[50.80mm]), Hr=720[12.70mm], Vr=-720[-12.70mm],  
Pattern=Long Dash, Start Angle = 315, End Angle = 135
```

For Square Corner Box Logging Example

Information about the GOCA Square corner box logging example:

```
==>Box T=S, FP=(7200.00[127.00mm], 1440.00[25.40mm]), DP=(10016.00[176.67mm],2880.00[50.80mm]),  
LT=Short Dashed  
-->Box T=S, FP=(7200.00[127.00mm], 1440.00[25.40mm]), DP=(10016.00[176.67mm],2880.00[50.80mm]),  
LT=Short Dashed
```

For Rounded Corner Box Logging Example

Information about the GOCA Round corner box logging example:

```
==>Box T=R, FP=(2880.00[50.80mm], 6480.00[114.30mm]), DP=(5760.00[101.60mm],7920.00[139.70mm]),  
LT=Double Dot, XR=(283.00[4.99mm]), YR=(283.00[4.99mm])  
-->Box T=R, FP=(2880.00[50.80mm], 6480.00[114.30mm]), DP=(5760.00[101.60mm],7920.00[139.70mm]),  
LT=Double Dot, XR=(283.00[4.99mm]), YR=(283.00[4.99mm])
```

For Fillet Logging Example

Information about the GOCA Fillet logging example:

```
==>Fillet, SP=(2880.00[50.80mm], 6120.00[107.95mm]), EP=(12960.00[228.60mm],5400.00[95.25mm]),  
LT=Default  
-->Fillet, SP=(2880.00[50.80mm], 6120.00[107.95mm]), EP=(12960.00[228.60mm],5400.00[95.25mm]),  
LT=Default
```

For Marker Logging Example

Information about the GOCA Marker logging example:

```
==>Marker, Name=Cross, Pos=(3600.00[63.50mm], 6120.00[107.95mm]), CW=144.00[2.54mm],  
CH=144.00[2.54mm]  
-->Marker, Name=Cross, Pos=(3600.00[63.50mm], 6120.00[107.95mm]), CW=144.00[2.54mm],  
CH=144.00[2.54mm]
```

6.6.2.4 Bar Code Object Logging

Format of the Bar Code Logging Message

Information about the format of Bar Code logging starts with the character sequence "==>", as shown below:

```
==>BBC Name=<Name> Size=(W:<value>[<value>mm], H:<value>[<value>mm]), Pos=(X:<value>[<value>mm],  
Y:<value>[<value>mm]), Type=<Type of medium>  
==>EBC Name=<Name>
```

Bar Code Logging Example

Information about the Bar Code logging example as shown below:

```
==>BBC Name=BCODE Size=(W:5620[99.13mm], H=843[14.87mm]), Pos=(X:1124.00[19.83mm],  
Y=5781.00[101.97mm]), Type=I  
==>EBC Name=BCODE
```

Codes Used in the Bar Code Logging Messages

The codes used in the Bar Code logging messages and their meanings:

Code	Description
BBC	Begin Barcode
EBC	End Barcode
W	Bar Code Width and corresponding mm value.
H	Bar Code Height and corresponding mm value.
X	X Position of Bar Code value.
Y	Y Position of Bar Code value.
Type	Type how this barcode object is referenced on parent container I - Page/Overlay included resource

6.6.3 ColorSpace Logging

6.6.3.1 Format of the ColorSpace Logging Message

Input Color Logging

Information about the input color logging starts with the character sequence "==>", as shown below:

```
==>ColorSpace=<Colorspace>, Color=<Value>[0x<value>]
```

Output Color Logging

Information about the output color logging starts with the character sequence "-->", as shown below:

```
-->ColorSpace=<Colorspace>, Color=<Value>[0xFF00]
```

6.6.3.2 ColorSpace Logging Example

Information about the ColorSpace logging example as shown in the below:

```
==>ColorSpace=SOCA, Color=65284[0xFF04]
```

```
-->ColorSpace=RGB, Color=65280[0xFF00]
```

6.6.3.3 Codes used in the ColorSpace Logging Messages

The codes used in the ColorSpace logging messages and their meanings:

Code	Description
ColorSpace	Color space for given color value
Color	Color value

6.6.4 Font Logging

You use the INI-parameter **LoggingLevel=ALL**, **LoggingLevel=FONT** or **LoggingFont=on** or the command line option **-ll:all** or **-lf** to produce a detailed list of all fonts used for the current spool file.

6.6.4.1 Format of the Font Logging Message

Input Fonts

Information about the input font starts with the character sequence "==">", as shown in the following example:

```
==>[CF=<Coded Font Name>],[ CS=<Char Set Name>],[ FGID=<FGID>],[ TF=<Type Face Name>, W=M|L|B[I],  
S=<Size>, SF|OT|TT|T1|RF|OF[,  
CP=<CodePageName> | CPGID=<Code Page Global ID>][,ENC=<Encoding>]
```

Output Fonts

Information about the output font starts with the character sequence "-->", as shown in the following example:

```
-->[CF=<Coded Font Name>],[ CS=<Char Set Name>],[ FGID=<FGID>],[ TF=<Type Face Name>, W=M|L|B[I],  
S=<Size>, SF|OT|TT|T1|RF|OF|RI[,  
CP=<CodePageName> | CPGID=<Code Page Global ID>], ACP=<CP Source>:<CP File>[, ENC=<Encoding>]
```

Text to Which the Font Applies

The text to which the font applies is enclosed in angled brackets: > text <, as shown in the following example:

>Insured <
>Geoffrey R Stephens<

6.6.4.2 Codes Used in the Font Logging Messages

The codes used in the font logging messages and their meanings:

Code	Description														
ACP	ASCII Code Page. "ACP" says the CP file that is used for AFP code page and how the CP file name is obtained. It also reports the encoding use														
CF	AFP Coded Font name. Valid only for AFP as either input or output font. For AFP as input font: Written only if input font type is "Coded Font".														
CP	AFP Code Page. Valid only for AFP. For AFP as input: Written only if input font type is "Coded Font" or "Character Set", not in case of "FGID". For AFP as output: Written only if output font type is "Coded Font" or "Character Set".														
CP File	File Name of ASCII code page (*.cp) file														
CP Source	<div>Source CP Source says where from CP file is obtained. The following codes are possible:</div> <table><tr><th>Value</th><th>Description</th></tr><tr><td>MD</td><td>CP File obtained from code page name specific Mapping.def entry</td></tr><tr><td>CP-DSC</td><td>CP File obtained from "Code Page Global ID" specified either in "Code Page Descriptor" SFI or in "Map Coded Font" SFI</td></tr><tr><td>NC</td><td>CPFile obtained from Naming Convention (last four letters of code page name, left most zeros will be truncated)</td></tr><tr><td>MD-DEF</td><td>CPFile obtained from "DEFAULT" Mapping.def entry</td></tr><tr><td>INI</td><td>CPFile obtained from INI File default "CodePage" Parameter</td></tr><tr><td>PG-DEF</td><td>Program Default, AFP2web predefined EBCDIC-TO-ASCII translation array is used for mapping</td></tr></table>	Value	Description	MD	CP File obtained from code page name specific Mapping.def entry	CP-DSC	CP File obtained from "Code Page Global ID" specified either in "Code Page Descriptor" SFI or in "Map Coded Font" SFI	NC	CPFile obtained from Naming Convention (last four letters of code page name, left most zeros will be truncated)	MD-DEF	CPFile obtained from "DEFAULT" Mapping.def entry	INI	CPFile obtained from INI File default "CodePage" Parameter	PG-DEF	Program Default, AFP2web predefined EBCDIC-TO-ASCII translation array is used for mapping
Value	Description														
MD	CP File obtained from code page name specific Mapping.def entry														
CP-DSC	CP File obtained from "Code Page Global ID" specified either in "Code Page Descriptor" SFI or in "Map Coded Font" SFI														
NC	CPFile obtained from Naming Convention (last four letters of code page name, left most zeros will be truncated)														
MD-DEF	CPFile obtained from "DEFAULT" Mapping.def entry														
INI	CPFile obtained from INI File default "CodePage" Parameter														
PG-DEF	Program Default, AFP2web predefined EBCDIC-TO-ASCII translation array is used for mapping														
CPGID	AFP Code Page Global ID. Written only if input font type is "FGID". Valid only for AFP input														
CS	Character Set. AFP Character Set name. Valid only for AFP input or output. For AFP as input font: Written only if input font type is "Coded Font" or "Character Set" not in case of "FGID".														

Emb	Font is embedded, used along with OT, TT and T1	
ENC	Encoding used	
	For PDF input:	
	Value	Description
	ANSI	Windows ANSI encoding
	ROMAN	Macintosh ROMAN encoding
	EXPERT	Macintosh EXPERT encoding
	STANDARD	Adobe STANDARD encoding
	SYMBOL	Symbol encoding
	NULL	Font's built-in encoding
	CUSTOM	PDF creator defined encoding
	Identity-H	Standard Adobe CMAP table
	For PDF and TIFF output:	
	Value	Description
	ANSI	Windows ANSI encoding
	SYMBOL	Symbol encoding
	NULL	Font's default built-in encoding
	For AFP output:	
	Value	Description
	ASCII	ASCII encoding
	EBCDIC	EBCDIC encoding
FGID	AFP Font Global ID. Written when input font type is "Coded Font" or "Character Set" or "FGID". Valid only for AFP input	
OF	Font type is AFP Outline font	
OT	Font type is Open Type	
Ref	Font is referenced, used along with OT, TT and T1	

RF	Font type is AFP Raster font
RI	Text Rasterized as Inline image
S	Font size (height) in points
SF	Font type is Standard or System (for the input font). For PDF output: Standard Font For TIFF output: System Font
T1	Font type is Type 1
TF	Font typeface name.
TT	Font type is True Type
UsedFont	For PDF and TIFF output: Name of used font For AFP output: Coded font (and or) Character Set name
W	Font weight and slant, it can have following possible values: M (Medium), L (Light), B (Bold), I (Italic). Examples: W=B means font weight is bold W=MI means font weight is medium and slant is italic

Whenever an AFP font resource is missing, an asterisk (*) will be added as a suffix to the resource name.
Example: CS=C1H400B0(*), TF=...

6.6.4.3 Examples of Font Logging Messages

Log for Raster Fonts

PDF output example:

```
==>CS=C1H400B0, FGID=2305, TF=HELVETICA LATIN1, W=B, S=12.00, RF, CP=T1GI0361
-->UsedFont=C1H400B0, TF=F1, W=B, S=12.00, RF, ACP=MD:2065.cp, ENC=T1GI0361
  @(8496.00,862.00)>Insurance<
  @(8496.00,1120.00)>Specialists<
```

Note: “F1” is the font name obtained from the mpdf library, and font is referenced using this name within the PDF file.

TIFF output example:

```

==>CS=C1H400B0, FGID=2305, TF=HELVETICA LATIN1, W=B, S=12.00, RF, CP=T1GI0361
-->UsedFont=C1H400B0, TF=HELVETICA LATIN1, W=B, S=12.00, RF, ACP=MD:2065.cp, ENC=T1GI0361
  @(8496.00,862.00)>Insurance<
  @(8496.00,1120.00)>Specialists<

```

AFP output (Coded Font):

```

==>CF=X1ZP09, CS=C0R20090, FGID=65280, TF=Arial, W=M, S=9.00, RF, CP=T1U00500
-->CF=X1ZP09, CS=C0R20090, FGID=65280, TF=Arial, W=M, S=9.00, RF, CP=T1U00500, ACP=CP-DSC:500.cp
  @(8674,2091)>Vos paiements sont acceptés à la plupart des banques ou <
  @(7673,2302)>N° de compte <

```

AFP output (Character Set):

```

==>CS=C1N40000, FGID=2309, TF=TIMES NEW ROMAN LATIN1, W=B, S=10.00, RF, CP=T1000395
-->CS=C1N40000, FGID=2309, TF=TIMES NEW ROMAN LATIN1, W=B, S=10.00, RF, CP=T1000395, ACP=MD:395.cp
  @(4592,3044)>99998888<
  @(567,3327)>Jane<

```

Log for Outline Fonts

PDF/TIFF output:

```

==>CF=X0420080, CS=CZ4200, FGID=416, TF=COURIER LATIN1, W=M, S=8.00, OF, CP=T1001141
-->UsedFont=CZ4200, TF=COURIER-LATIN1, W=M, S=8.00, OF, ACP=MD:1141.cp, ENC=T1001141
  @(3645.00,2994.00)>Kd.-Nr.<
  @(4197.00,2994.00)>11.11111111<

```

AFP output (Coded Font):

```

==>CF=X0420080, CS=CZ4200, FGID=416, TF=COURIER LATIN1, W=M, S=8.00, OF, CP=T1001141
-->CF=X0420080, CS=CZ4200, FGID=416, TF=COURIER LATIN1, W=M, S=8.00, OF, CP=T1001141
  @(3645.00,2994.00)>Kd.-Nr.<
  @(4197.00,2994.00)>11.11111111<

```

AFP Output (Character Set):

```

==>CS=CZH200, FGID=2304, TF=HELVETICA LATIN1, W=M, S=8.00, OF, CP=T1001141
-->FGID=2304, TF=HELVETICA LATIN1, W=M, S=8.00, RF, ACP=PG-DEF:500.cp
  @(3645,2994)>Kd.-Nr.<
  @(4197,2994)>11.11111111<

```

Log for Font Referencing

CASE 1: Referencing one of the 14 standard fonts for PDF output, or system fonts for TIFF output

PDF/TIFF output:

```

==>CS=C1H400B0, FGID=2305, TF=HELVETICA LATIN1, W=B, S=12.00, RF, CP=T1GI0361
-->UsedFont=Helvetica-Bold, TF=Helvetica-Bold, W=B, S=12.00, SF, ACP=MD:2065.cp, ENC=ANSI
  @(8496.00,862.00)>Insurance<
  @(8496.00,1120.00)>Specialists<

```

CASE 2: Referencing external Type1/TrueType font files

PDF/TIFF output:

```

==>CS=C1H400B0, FGID=2305, TF=HELVETICA LATIN1, W=B, S=12.00, RF, CP=T1GI0361
-->UsedFont=./extfont/timesb.pfm, TF=FreeSerifBold, W=B, S=12.00, T1, ReF, ACP=MD:2065.cp, ENC=ANSI
  @(8496.00,862.00)>Insurance<
  @(8496.00,1120.00)>Specialists<

```

CASE 3: Referencing external AFP fonts

AFP output (Coded Font)

```

==>CF=X0420080, CS=CZ4200, FGID=416, TF=COURIER LATIN1, W=M, S=8.00, RF, CP=T1001141
-->CF=X0420080, CS=CZ4200, FGID=416, TF=COURIER LATIN1, W=M, S=8.00, Ref, CP=T1001141
  @(3645.00,2994.00)>Kd.-Nr.<
  @(4197.00,2994.00)>11.11111111<

```

AFP output (Character Set)

```

==>CS=C1H400B0, FGID=2305, TF=HELVETICA LATIN1, W=B, S=12.00, RF, CP=T1GI0361
-->CS=C1H400B0, FGID=2305, TF=HELVETICA LATIN1, W=B, S=12.00, RF, CP=T1GI0361, ACP=MD:2065.cp
  @(8496,862)>Insurance<
  @(8496,1120)>Specialists<

```

Log for Font Embedding

PDF output:

```

==>CS=C1H200D0 TF=Helvetica Font File Found="./extfont/helvetica.pfb". Font type=PFB(TYPE1 Data).
==>CS=C1H200D0 TF=Helvetica Font File Found="./extfont/helvetica.pfm". Font type=PFM(TYPE1 Metric).
  Parsing Type1 PFM File "./extfont/helvetica.pfm"
  PostScript Name "FreeSans"
==>CS=C1H200D0, FGID=2304, TF=HELVETICA LATIN1, W=M, S=14.00, RF, CP=T1GI0361
-->UsedFont=./extfont/helvetica.pfb, TF=FreeSans, W=M, S=14.00, T1, Emb, ACP=MD:2065.cp, ENC=ANSI
  @(792.00,2638.00)>The Preferred Professional<

```

For TIFF output, font embedding is not possible, TIFF output is therefore always treated as “Font Referencing”.

AFP output is not handled.

Log for FGID Font

PDF/TIFF output:

```

==>FGID=2304, TF=HELVETICA LATIN1, W=M, S=10.00, RF, CP=T1GIG037, CPGID=37
-->UsedFont=Helvetica, TF=Helvetica, W=M, S=10.00, SF, ACP=MD:37.cp, ENC=ANSI
  @(720.00,1260.00)> 5200539 DR. DIETRICH BRUCE E. STREET & BEM CODE/NAAM 99999999<
  @(720.00,1440.00)> MRKTER CODE/NAME<

```

AFP output is not handled.

Log for Text added through the Scripting Facility

PDF/TIFF output:

```

==>TF=Helvetica, W=M, S=10.00, SF
-->UsedFont=Helvetica, TF=Helvetica, W=M, S=10.00(D), SF, ENC=ANSI
  @(283.46,472.44)>Maas Holding GmbH<

```

6.6.5 Image Logging

6.6.5.1 Format of the Image Logging Message

Input Image Logging

Information about the image logging starts with the character sequence “==>”, as shown below:

```
==>Image=<Image Name>, ImgSize=(W:<value>[<value>mm], H:<value>[<value>mm]), Type=<Image Type>,  
Compression=<Image Compression>, Pos=(X:<value>[<value>mm], Y:<value>[<value>mm])
```

Output Image Logging

Information about the image logging starts with the character sequence "-->", as shown below:

```
-->Image=<Image Name>, ImgSize=(W:<value>[<value>mm], H:<value>[<value>mm]), BPP=<Bits/Pixel>,  
Compression=<Image Compression>, Pos=(X:<value>[<value>mm], Y:<value>[<value>mm]),  
PresentationSize=(X:<value>[<value>mm], Y:<value>[<value>mm])
```

6.6.5.2 Image Logging Example

Information about the Image logging example as shown below:

```
==>Image=IMAG_BLK, ImgSize=(W:160.00[16.93mm], H:192.00[20.32mm]), Type=IM Raster,  
Compression=Uncompressed, Pos=(X:7329.00[129.28mm], Y:14040.00[247.65mm])  
-->Image=IMAG_BLK, ImgSize=(W:160.00[16.93mm], H:192.00[20.32mm]), BPP=1, Compression=DEFLATE,  
Pos=(X:7329.00[129.28mm], Y:14040.00[247.65mm]),  
PresentationSize=(W:160.00[16.93mm], H:192.00[20.32mm])
```

6.6.5.3 Codes Used in the Image Logging Messages

The codes used in the Image logging messages and their meanings:

Code	Description
Image	Image Name
W	Image/Presentation size of width and corresponding mm value.
H	Image/Presentation size of Height and corresponding mm value.
Type	Where <Image Types> are <ul style="list-style-type: none">•IM Raster•IOCA•Raster
Compression	Specifies Input/Output compression type
X	X Position of the Image and corresponding mm value.
Y	Y Position of the Image and corresponding mm value.
BPP	Image Bits Per Pixels.

6.6.6 Line Logging

6.6.6.1 Format of the Line Logging Message

Input Line Logging

Information about the line logging starts with the character sequence "==>", as shown below:

```
==>Line Orn=<HLine|VLine>, Width=<Value>[<Value>mm], Length=<Value>[<Value>mm],  
StartPos=(X:<Value>[<Value>mm], Y:<Value>[<Value>mm]), EndPos=(X:<Value>[<Value>mm],  
Y:<Value>[<Value>mm])
```

Output Line Logging

Information about the line logging starts with the character sequence "-->", as shown below:

```
-->Line Orn=<HLine|VLine>, Width=<Value>[<Value>mm], Length=<Value>[<Value>mm],  
StartPos=(X:<Value>[<Value>mm], Y:<Value>[<Value>mm]), EndPos=(X:<Value>[<Value>mm],  
Y:<Value>[<Value>mm])
```

6.6.6.2 Line Logging Example

Information about the line logging example as shown below:

```
==>Line Orn=HLine, Width=11.00[0.19mm], Length=12019[212.00mm], StartPos=(X:0.00[0.00mm],  
Y:13310.00[234.77mm]), EndPos=(X:12019.00[212.00mm], Y:13321.00[234.97mm])  
-->Line Orn=HLine, Width=11.00[0.19mm], Length=12019[212.00mm], StartPos=(X:0.00[0.00mm],  
Y:13304.50[234.68mm]), EndPos=(X:12019.00[212.00mm], Y:13315.50[234.87mm])
```

6.6.6.3 Codes Used in the Line Logging Messages

The codes used in the Line logging messages and their meanings:

Code	Description
Line	Line Orientation whether Vertical/ Horizontal (VLine/HLine)
Width	Line width and corresponding mm value.
Length	Line length and corresponding mm value.
X	X Starting/Ending Position of the Line and corresponding mm value.
Y	Y Starting/Ending Position of the Line and corresponding mm value.

6.6.7 AFP Modca-IS2 Compliance Logging

For AFP output with Modca-IS2 compliance, PT1 subset must be used. When IS2 compliance flag "AFPForcePTValueControlFlag" is off, AFP2web allows PTOCA control sequences to have values that are out of range defined by PT1 subset and writes warning messages to the log file.

```
==>TF=Arial, W=M, S=8.00, Type1, ENC=WinAnsi
-->CS=CZH200, TF=Arial, W=M, S=8.00, OF, CP=T1001141(*), ACP=MD:1141.cp
@(45,1749)>Geoffrey R Stephens<Warning! Variable Space Increment(PTOCA SVI) value "884" is not
within valid range(0-853) for Modca-IS2 compliance.
```

6.7 Identifying Fonts Using the IBM Naming Convention

6.7.1 Overview

If you do not provide the AFP font resources Character Set and Coded Font in-line, AFP2web will take the settings of the mapping.def to identify the substitution fonts to be used for the conversion.

If no information is found in the mapping.def, AFP2web will derive font characteristics from the naming conventions for these resources. Thus, AFP2web will still be able to map AFP fonts to substitution fonts.

In this section, we describe the rules for this automatic font mapping. Please note that we follow conventions, which are specific to IBM.

6.7.2 The AFP Font Naming Convention

6.7.2.1 The General Structure

An AFP Character Set or Coded Font has a name with up to 8 characters. Each character has the following information encoded:

A	F	R	S	T	C	P	X
---	---	---	---	---	---	---	---

Meaning		Used by AFP2web
A	Component	Yes
F	Format or orientation	Ignored
R	Type family	Yes
S	Style	Yes
T	Weight	Yes

C	Complement.	Ignored
P	Point size	Yes
X	Used to determine the code page.	Ignored

For example, the name of AFP resource COT07560 has the following information encoded:

A=C	The resource is a Character Set.
F=0	(Ignored)
R=T	The type family is Sonoran Serif.
S=0	Style is Roman.
T=7	Weight is Bold.
C=5	(Ignored)
P=6	Point size 6.
X=0	(Ignored)

The encoding rules are summarized in the following tables.

6.7.2.2 A: Component

A	Component
C	Character Set
X	Coded Font
A	Component

6.7.2.3 R: Type family

For regular fonts R is one the following:

R	Type Family	AFP2web Default Type Family for PDF	AFP2web Default Type Family for UNIX TIFF	AFP2web Default Type Family for Windows TIFF
4	Courier	Courier	courier	Courier

5	Letter Gothic	Courier	courier	Courier
6	Gothic Text	Courier	courier	Courier
7	Prestige	Courier	courier	Courier
8	Boldface	Times-Roman	times	Times New Roman
9	OCR	Courier	courier	Courier
B	BookMaster	Courier	courier	Courier
H	Helvetica	Helvetica	helvetica	Helvetica
N	Times New Roman	Times-Roman	times	Times New Roman

For licensed fonts, R is one the following:

R	Type Family	AFP2web Default Type Family for PDF	AFP2web Default Type Family for UNIX TIFF	AFP2web Default Type Family for Windows TIFF
2	Proprinter Emulation	Helvetica	helvetica	Helvetica
A	Sonoran Sans Serif	Helvetica	helvetica	Helvetica
B	Bar Code	Helvetica	helvetica	Helvetica
C	Century Schoolbook	NewCenturySchlbk	new century schoolbook	Century Schoolbook
G	Monotype Garamond	Courier	courier	Courier
J	Sonoran Display	Times-Roman	times	Times New Roman
M	Mathematic and Science	Symbol	symbol	Symbol
O	Optical Character Recognition	Courier	courier	Courier
P	Pi Sans Serif	Helvetica	helvetica	Helvetica
Q	Pi Serif	Times-Roman	times	Times New Roman
S	ITC Souvenir	Helvetica	helvetica	Helvetica
T	Sonoran Serif	Times-Roman	times	Times New Roman
V	ITC Avant Garde Gothic	Helvetica	helvetica	Helvetica

Z	Sonoran Petite	Times-Roman	times	Times New Roman
---	----------------	-------------	-------	-----------------

6.7.2.4 S: Style

S	Style	AFP2web Default Style
0	Roman	Medium
1	Italic	Italic
2	Roman Medium	Medium
3	Italic Medium	Italic
4	Roman Bold	Bold
5	Italic Bold	Italic Bold
6	Roman Medium Reverse	Medium

6.7.2.5 T: Weight

For licensed fonts T is one the following (for regular fonts T is ignored):

T	Weight	AFP2web Default Weight: TIFF output under Windows	AFP2web Default Weight: PDF and TIFF output under Unix
1	Ultralight	Light	Medium
2	Extralight	Light	Medium
3	Light	Light	Medium
4	Semilight	Light	Medium
5	Medium	Medium	Medium
6	Semibold	Bold	Bold
7	Bold	Bold	Bold
8	Extrabold	Bold	Bold
9	Ultrabold	Bold	Bold

6.7.2.6 P: Point size

P	Point Size (points)	P	Point Size (points)
4	4	K	21
5	5	L	22
6	6	M	23
7	7	N	24
8	8	O	25
9	9	P	26
0	10	Q	27
A	11	R	28
B	12	S	29
C	13	T	30
D	14	U	31
E	15	V	32
F	16	W	33
G	17	X	34
H	18	Y	35
I	19	Z	36
J	20		

6.8 Scripting Facility Quick Reference

6.8.1 Scripting Facility Interface and Processing Levels

The Scripting Facility offers an interface used to intercept AFP2web processing at particular processing events. A custom script must have the following sections defined:

- sub afp2web()
- sub initialize()
- sub initializeDoc()
- sub initializePage()
- sub finalizePage()
- sub finalizeDoc()
- sub finalize()

In the following tables, a quick-reference shows which API method is available in which scripting subroutine.

6.8.2 Script Routine and Methods

6.8.2.1 Script Routine: initialize

The following packages and methods are available:

Package	Method
a2w::Bookmark	addChild, getColor, getFirstChild, getLevelNumber, getNextChild, getTargetName, getTargetType, getTitle, isBoldFont, isItalicFont, new, remove, setBoldFont, setColor, setItalicFont, setLevelNumber, setTargetName, setTargetType, setTitle
a2w::Comment	new, setValue, getValue
a2w::Config	getAttribute, setAttribute
a2w::Font	getCharacterSetName, getCodedFontName, getCodePageName, getEncoding, getHeight, getName, getWidth, isBold, isItalic, isLight, isStrikeover, isUnderline, new, new(FontType, Charset, CodePage), new(FontType, CodedFont), new(FontType, FontName), setBold, setCharacterSetName, setCodedFontName, setCodePageName, setEncoding, setFamilyName, setHeight, setItalic, setLight, setName, setStrikeover, setUnderline, setWidth
a2w::Index	getIndexedObjectName, getIndexedObjectType, getLevelNumber, getName, getSequenceNumber, getValue, new, remove, setLevelNumber, setName, setSequenceNumber, setValue

a2w::Kernel	getExitStatus, getIndexFilename, getResourceFilename, getSpoolFilename, getVersionAll, getLastErrorMessage
a2w::Line	getColor, getLength, getWidth, getXPos, getYPos, isHorizontal, isNegative, isVertical, new, remove, setColor, setHorizontal, setLength, setNegative, setVertical, setWidth, setXPos, setYPos
a2w::NOP	getLength, getValue, new, remove, setValue
a2w::Page	addAnnotation, addBookmark, addImage, addIndex, addLine, addNOP, addOverlay, addPSEG, addText, getFirstBookmark, getFirstIndex, getFirstLine, getFirstNOP, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getNextBookmark, getNextIndex, getNextLine, getNextNOP, getNextOverlay, getNextPageSegment, getNextText, getResolution, getRotation, getWidth, isOverlayFound, new, setBackgroundColor, setHeight, setResolution, setRotation, setWidth, getFirstImage, getNextImage, getFirstVector, getNextVector
a2w::Text	getAngle, getColor, getFont, getText, getTextLen, getXPos, getYPos, new, remove, setAngle, setColor, setFont, setText, setTextLen, setXPos, setYPos
a2w::UserData	getData, getDataLength, getId, new, setData, setDataLength, setId

6.8.2.2 Script Routine: initializeDoc

The following packages and methods are available:

Package	Method
a2w::Bookmark	addChild, getColor, getFirstChild, getLevelNumber, getNextChild, getTargetName, getTargetType, getTitle, isBoldFont, isItalicFont, new, remove, setBoldFont, setColor, setItalicFont, setLevelNumber, setTargetName, setTargetType, setTitle
a2w::Comment	new, setValue, getValue, remove
a2w::Config	getAttribute
a2w::Document	addBookmark, addIndex, addNOP, addPage, addUserData, getFirstBookmark, getFirstIndex, getFirstUserData, getId, getName, getNextBookmark, getNextIndex, getNextUserData, getOutputFilename, getPID, getSimpleFilename, getSimpleResourceGroupName, getUserData, setName, setOutputFilename, setSimpleResourceGroupName, addComment, getFirstComment, getNextComment, getProperty, getMetaDataStream
a2w::Font	getCharacterSetName, getCodedFontName, getCodePageName, getEncoding, getHeight, getName, getWidth, isBold, isItalic, isLight, isStrikeover, isUnderline, new, new(FontType, Charset, CodePage), new(FontType, CodedFont), new(FontType, FontName), setBold, setCharacterSetName, setCodedFontName, setCodePageName, setEncoding, setFamilyName, setHeight, setItalic, setLight, setName, setStrikeover, setUnderline, setWidth
a2w::Index	getIndexedObjectName, getIndexedObjectType, getLevelNumber, getName, getSequenceNumber, getValue, new, remove, setLevelNumber, setName, setSequenceNumber, setValue
a2w::Kernel	getExitStatus, getIndexFilename, getResourceFilename, getSpoolFilename, getVersionAll, getLastErrorMessage

a2w::Line	getColor, getLength, getWidth, getXPos, getYPos, isHorizontal, isNegative, isVertical, new, remove, setColor, setHorizontal, setLength, setNegative, setVertical, setWidth, setXPos, setYPos
a2w::NOP	getLength, getValue, new, remove, setValue
a2w::Page	addAnnotation, addBookmark, addImage, addIndex, addLine, addNOP, addOverlay, addPSEG, addText, getFirstBookmark, getFirstIndex, getFirstLine, getFirstNOP, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getNextBookmark, getNextIndex, getNextLine, getNextNOP, getNextOverlay, getNextPageSegment, getNextText, getResolution, getRotation, getWidth, isOverlayFound, new, setBackgroundColor, setHeight, setResolution, setRotation, setWidth, getFirstImage, getNextImage, getFirstVector, getNextVector
a2w::Text	getAngle, getColor, getFont, getText, getTextLen, getXPos, getYPos, new, remove, setAngle, setColor, setFont, setText, setTextLen, setXPos, setYPos
a2w::UserData	getData, getDataLength, getId, new, setData, setDataLength, setId

6.8.2.3 Script Routine: initializePage

The following packages and methods are available:

Package	Method
a2w::Bookmark	addChild, getColor, getFirstChild, getLevelNumber, getNextChild, getTargetName, getTargetType, getTitle, isBoldFont, isItalicFont, new, remove, setBoldFont, setColor, setItalicFont, setLevelNumber, setTargetName, setTargetType, setTitle
a2w::Comment	new, setValue, getValue, remove
a2w::Config	getAttribute
a2w::Document	addBookmark, addIndex, addNOP, addPage, addUserData, getFirstBookmark, getFirstIndex, getFirstUserData, getId, getName, getNextBookmark, getNextIndex, getNextUserData, getOutputFilename, getPID, getSimpleFilename, getSimpleResourceGroupName, getUserData, setName, setOutputFilename, setSimpleResourceGroupName, addComment, getFirstComment, getNextComment, getProperty, getMetaDataStream
a2w::Font	getCharacterSetName, getCodedFontName, getCodePageName, getEncoding, getHeight, getName, getWidth, isBold, isItalic, isLight, isStrikeover, isUnderline, new, new(FontType, Charset, CodePage), new(FontType, CodedFont), new(FontType, FontName), setBold, setCharacterSetName, setCodedFontName, setCodePageName, setEncoding, setFamilyName, setHeight, setItalic, setLight, setName, setStrikeover, setUnderline, setWidth
a2w::Index	getIndexedObjectName, getIndexedObjectType, getLevelNumber, getName, getSequenceNumber, getValue, new, remove, setLevelNumber, setName, setSequenceNumber, setValue
a2w::Kernel	getExitStatus, getIndexFilename, getResourceFilename, getSpoolFilename, getVersionAll, getLastErrorMessages
a2w::Line	getColor, getLength, getWidth, getXPos, getYPos, isHorizontal, isNegative, isVertical, new, remove, setColor, setHorizontal, setLength, setNegative, setVertical, setWidth, setXPos, setYPos

a2w::NOP	getEBCDICValue, getLength, getValue, new, remove, setValue
a2w::Page	addAnnotation, addBookmark, addImage, addIndex, addLine, addNOP, addOverlay, addPSEG, addText, getFirstBookmark, getFirstIndex, getFirstLine, getFirstNOP, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getName, getNextBookmark, getNextIndex, getNextLine, getNextNOP, getNextOverlay, getNextPageSegment, getNextText, getOutputFilename, getPageGroupName, getParsedId, getResolution, getRotation, getSimpleFilename, getWidth, isOverlayFound, new, setBackgroundColor, setHeight, setOutputFilename, setResolution, setRotation, setWidth, getFirstImage, getNextImage, getFirstVector, getNextVector
a2w::Text	getAngle, getColor, getFont, getMappedFontLocalId, getText, getTextLen, getXPos, getYPos, new, remove, setAngle, setColor, setFont, setText, setEBCDICText, setTextLen, setXPos, setYPos
a2w::UserData	getData, getDataLength, getId, new, setData, setDataLength, setId

6.8.2.4 Script Routine: **afp2web**

The following packages and methods are available:

Package	Method
a2w::Bookmark	addChild, getColor, getFirstChild, getLevelNumber, getNextChild, getTargetName, getTargetType, getTitle, isBoldFont, isItalicFont, new, remove, setBoldFont, setColor, setItalicFont, setLevelNumber, setTargetName, setTargetType, setTitle
a2w::Comment	new, setValue, getValue, remove
a2w::Config	getAttribute
a2w::Document	addBookmark, addIndex, addNOP, addPage, addUserData, getFirstBookmark, getFirstIndex, getFirstUserData, getId, getName, getNextBookmark, getNextIndex, getNextUserData, getOutputFilename, getPID, getSimpleFilename, getSimpleResourceGroupName, getUserData, setName, setOutputFilename, setSimpleResourceGroupName, addComment, getFirstComment, getNextComment, getProperty, getMetaDataStream
a2w::Font	getCharacterSetName, getCodedFontName, getCodePageName, getEncoding, getHeight, getName, getTypefaceName, getWidth, isBold, isItalic, isLight, isStrikeover, isUnderline, new, new(FontType, Charset, CodePage), new(FontType, CodedFont), new(FontType, FontName), setBold, setCharacterSetName, setCodedFontName, setCodePageName, setEncoding, setFamilyName, setHeight, setItalic, setLight, setName, setStrikeover, setUnderline, setWidth
a2w::Image	getXPos, getYPos, getAngle, getName, getData, getDataLength, getCompression, getBitsPerPixel, getSamplesPerPixel, getWidth, getHeight, getPresentationWidth, getPresentationHeight, getUniqueColorsCount, remove
a2w::Index	getEBCDICName, getEBCDICValue, getIndexedObjectName, getIndexedObjectType, getLevelNumber, getName, getNameLength, getSequenceNumber, getValue, getValueLength, new, remove, setLevelNumber, setName, setSequenceNumber, setValue
a2w::Kernel	getExitStatus, getIndexFilename, getResourceFilename, getSpoolFilename, getVersionAll, getLastErrorMessages

a2w::Line	getColor, getLength, getWidth, getXPos, getYPos, isHorizontal, isNegative, isVertical, new, remove, setColor, setHorizontal, setLength, setNegative, setVertical, setWidth, setXPos, setYPos
a2w::MediumMap	getDuplexControl, getFormdefName, getHeight, getName, getNupControl, getResolution, getWidth
a2w::NOP	getEBCDICValue, getLength, getValue, new, remove, setValue
a2w::Overlay	getFirstLine, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getIncludedXPosition, getIncludedYPosition, getName, getNextLine, getNextOverlay, getNextPageSegment, getNextText, getResolution, getWidth
a2w::Page	addAnnotation, addBookmark, addImage, addIndex, addLine, addNOP, addOverlay, addPSEG, addText, getAppliedMediumMap, getFirstBookmark, getFirstIndex, getFirstLine, getFirstMediumOverlay, getFirstNOP, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getName, getNextBookmark, getNextIndex, getNextLine, getNextMediumOverlay, getNextNOP, getNextOverlay, getNextPageSegment, getNextText, getOutputFilename, getPageGroupName, getParseId, getResolution, getRotation, getSimpleFilename, getWidth, isConstantBack, isOverlayFound, new, setBackgroundColor, setHeight, setOutputFilename, setResolution, setRotation, setWidth, getFirstImage, getNextImage, getFirstVector, getNextVector
a2w::PSEG	getIncludedXPosition, getIncludedYPosition, getName
a2w::Text	getAngle, getColor, getEBCDICText, getFont, getMappedFontLocalId, getText, getTextLen, getXPos, getYPos, new, remove, setAngle, setColor, setFont, setText, setEBCDICText, setTextLen, setXPos, setYPos
a2w::UserData	getData, getDataLength, getId, new, setData, setDataLength, setId
a2w::Vector	getXPos, getYPos, getAngle, getWidth, getHeight, remove

6.8.2.5 Script Routine: finalizePage

The following packages and methods are available:

Package	Method
a2w::Bookmark	addChild, getColor, getFirstChild, getLevelNumber, getNextChild, getTargetName, getTargetType, getTitle, isBoldFont, isItalicFont, new, remove, setBoldFont, setColor, setItalicFont, setLevelNumber, setTargetName, setTargetType, setTitle
a2w::Comment	new, setValue, getValue, remove
a2w::Config	getAttribute
a2w::Document	addBookmark, addIndex, addNOP, addPage, addUserData, getFirstBookmark, getFirstIndex, getFirstUserData, getId, getName, getNextBookmark, getNextIndex, getNextUserData, getOutputFilename, getPID, getSimpleFilename, getSimpleResourceGroupName, getUserData, setName, setOutputFilename, setSimpleResourceGroupName, addComment, getFirstComment, getNextComment, getProperty, getMetaDataStream

a2w::Font	getCharacterSetName, getCodedFontName, getCodePageName, getEncoding, getHeight, getName, getTypefaceName, getWidth, isBold, isItalic, isLight, isStrikeover, isUnderline, new, new(FontType, CharSet, CodePage), new(FontType, CodedFont), new(FontType, FontName), setBold, setCharacterSetName, setCodedFontName, setCodePageName, setEncoding, setFamilyName, setHeight, setItalic, setLight, setName, setStrikeover, setUnderline, setWidth
a2w::Image	getXPos, getYPos, getAngle, getName, getData, getDataLength, getCompression, getBitsPerPixel, getSamplesPerPixel, getWidth, getHeight, getPresentationWidth, getPresentationHeight, getUniqueColorsCount, remove
a2w::Index	getEBCDICName, getEBCDICValue, getIndexedObjectName, getIndexedObjectType, getLevelNumber, getName, getNameLength, getSequenceNumber, getValue, getValueLength, new, remove, setLevelNumber, setName, setSequenceNumber, setValue
a2w::Kernel	getExitStatus, getIndexFilename, getResourceFilename, getSpoolFilename, getVersionAll, getLastErrorMessage
a2w::Line	getColor, getLength, getWidth, getXPos, getYPos, isHorizontal, isNegative, isVertical, new, remove, setColor, setHorizontal, setLength, setNegative, setVertical, setWidth, setXPos, setYPos
a2w::MediumMap	getDuplexControl, getFormdefName, getHeight, getName, getNupControl, getResolution, getWidth
a2w::NOP	getEBCDICValue, getLength, getValue, new, remove, setValue
a2w::Overlay	getFirstLine, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getIncludedXPosition, getIncludedYPosition, getName, getNextLine, getNextOverlay, getNextPageSegment, getNextText, getResolution, getWidth
a2w::Page	addAnnotation, addBookmark, addImage, addIndex, addLine, addNOP, addOverlay, addPSEG, addText, getAppliedMediumMap, getFirstBookmark, getFirstIndex, getFirstLine, getFirstMediumOverlay, getFirstNOP, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getName, getNextBookmark, getNextIndex, getNextLine, getNextMediumOverlay, getNextNOP, getNextOverlay, getNextPageSegment, getNextText, getOutputFilename, getPageGroupName, getParsed, getResolution, getRotation, getSimpleFilename, getWidth, isConstantBack, isOverlayFound, new, setBackgroundColor, setHeight, setOutputFilename, setResolution, setRotation, setWidth, getFirstImage, getNextImage, getFirstVector, getNextVector
a2w::PSEG	getIncludedXPosition, getIncludedYPosition, getName
a2w::Text	getAngle, getColor, getEBCDICText, getFont, getMappedFontLocalId, getText, getTextLen, getXPos, getYPos, new, remove, setAngle, setColor, setFont, setText, setEBCDICText, setTextLen, setXPos, setYPos
a2w::UserData	getData, getDataLength, getId, new, setData, setDataLength, setId
a2w::Vector	getXPos, getYPos, getAngle, getWidth, getHeight, remove

6.8.2.6 Script Routine: finalizeDoc

The following packages and methods are available:

Package	Method
a2w::Bookmark	getColor, getFirstChild, getLevelNumber, getNextChild, getTargetName, getTargetType, getTitle, isBoldFont, isItalicFont
a2w::Comment	getValue
a2w::Config	getAttribute
a2w::Document	getFirstBookmark, getFirstIndex, getFirstPage, getFirstUserData, getId, getName, getNextBookmark, getNextIndex, getNextPage, getNextUserData, getOutputBuffer, getOutputBufferLength, getOutputFilename, getPageCount, getPID, getSimpleFilename, getSimpleResourceGroupName, getSize, getUserData, getFirstComment, getNextComment, getProperty, getMetaDataStream
a2w::Font	getCharacterSetName, getCodedFontName, getCodePageName, getEncoding, getHeight, getName, getTypefaceName, getWidth, isBold, isItalic, isLight, isStrikeover, isUnderline
a2w::Image	getXPos, getYPos, getAngle, getName, getData, getDataLength, getCompression, getBitsPerPixel, getSamplesPerPixel, getWidth, getHeight, getPresentationWidth, getPresentationHeight, getUniqueColorsCount
a2w::Index	getEBCDICName, getEBCDICValue, getIndexedObjectName, getIndexedObjectType, getLevelNumber, getName, getNameLength, getSequenceNumber, getValue, getValueLength
a2w::Kernel	getExitStatus, getIndexFilename, getResourceFilename, getSpoolFilename, getVersionAll, getLastErrorMessages
a2w::Line	getColor, getLength, getWidth, getXPos, getYPos, isHorizontal, isNegative, isVertical
a2w::MediumMap	getDuplexControl, getFormdefName, getHeight, getName, getNupControl, getResolution, getWidth
a2w::NOP	getEBCDICValue, getLength, getValue
a2w::Overlay	getFirstLine, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getIncludedXPosition, getIncludedYPosition, getName, getNextLine, getNextOverlay, getNextPageSegment, getNextText, getResolution, getWidth
a2w::Page	getAppliedMediumMap, getFirstBookmark, getFirstIndex, getFirstLine, getFirstMediumOverlay, getFirstNOP, getFirstOverlay, getFirstPageSegment, getFirstText, getHeight, getId, getName, getNextBookmark, getNextIndex, getNextLine, getNextMediumOverlay, getNextNOP, getNextOverlay, getNextPageSegment, getNextText, getOutputBuffer, getOutputBufferLength, getOutputFilename, getPageGroupName, getParseId, getResolution, getRotation, getSimpleFilename, getSize, getWidth, isConstantBack, isOverlayFound, getFirstImage, getNextImage, getFirstVector, getNextVector
a2w::PSEG	getIncludedXPosition, getIncludedYPosition, getName
a2w::Text	getAngle, getColor, getEBCDICText, getFont, getMappedFontLocalId, getText, getTextLen, getXPos, getYPos

a2w::UserData	getData, getDataLength, getId
a2w::Vector	getXPos, getYPos, getAngle, getWidth, getHeight

6.8.2.7 Script Routine: finalize

The following packages and methods are available:

Package	Method
a2w::Config	getAttribute
a2w::Kernel	getExitStatus, getIndexFilename, getResourceFilename, getSpoolFilename, getVersionAll, getLastErrorMessage

6.8.3 Methods and Where Used in a Script

6.8.3.1 a2w::Bookmark Methods

Methods	Available in Script Routines
addChild	initialize, initializeDoc, initializePage, afp2web, finalizePage
getColor	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstChild	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getLevelNumber	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextChild	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getTargetName	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getTargetType	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getTitle	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isBoldFont	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isItalicFont	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage

remove	initialize, initializeDoc, initializePage, afp2web, finalizePage
setBoldFont	initialize, initializeDoc, initializePage, afp2web, finalizePage
setColor	initialize, initializeDoc, initializePage, afp2web, finalizePage
setItalicFont	initialize, initializeDoc, initializePage, afp2web, finalizePage
setLevelNumber	initialize, initializeDoc, initializePage, afp2web, finalizePage
setTargetName	initialize, initializeDoc, initializePage, afp2web, finalizePage
setTargetType	initialize, initializeDoc, initializePage, afp2web, finalizePage
setTitle	initialize, initializeDoc, initializePage, afp2web, finalizePage

6.8.3.2 a2w::Comment Methods

Methods	Available in Script Routines
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
setValue	initialize, initializeDoc, initializePage, afp2web, finalizePage
getValue	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
remove	initializeDoc, initializePage, afp2web, finalizePage

6.8.3.3 a2w::Config Methods

Methods	Available in Script Routines
getAttribute	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc, finalize
setAttribute	initialize

6.8.3.4 a2w::Document Methods

Methods	Available in Script Routines
addBookmark	initializeDoc, initializePage, afp2web, finalizePage
addIndex	initializeDoc, initializePage, afp2web, finalizePage
addNOP	initializeDoc, initializePage, afp2web, finalizePage
addPage	initializeDoc, initializePage, afp2web, finalizePage

addUserData	initializeDoc, initializePage, afp2web, finalizePage
getFirstBookmark	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstIndex	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstPage	finalizeDoc
getFirstUserData	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getId	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getName	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextBookmark	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextIndex	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextPage	finalizeDoc
getNextUserData	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getOutputBuffer	finalizeDoc
getOutputBufferLength	finalizeDoc
getOutputFilename	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getPageCount	finalizeDoc
getPID	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getSimpleFilename	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getSimpleResourceGroupName	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getSize	finalizeDoc
getUserData	initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
setName	initializeDoc, initializePage, afp2web, finalizePage
setOutputFilename	initializeDoc, initializePage, afp2web, finalizePage
setSimpleResourceGroupName	initializeDoc, initializePage, afp2web, finalizePage
addComment	initializeDoc, initializePage, afp2web, finalizePage
getFirstComment	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc

getNextComment	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getProperty	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getMetaDataStream	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc

6.8.3.5 a2w::Font Methods

Methods	Available in Script Routines
getCharacterSetName	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getCodedFontName	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getCodePageName	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getEncoding	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getHeight	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getName	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getTypefaceName	afp2web, finalizePage, finalizeDoc
getWidth	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isBold	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isItalic	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isLight	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isStrikeover	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isUnderline	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
new(FontType, Charset, CodePage)	initialize, initializeDoc, initializePage, afp2web, finalizePage
new(FontType, CodedFont)	initialize, initializeDoc, initializePage, afp2web, finalizePage
new(FontType, FontName)	initialize, initializeDoc, initializePage, afp2web, finalizePage
setBold	initialize, initializeDoc, initializePage, afp2web, finalizePage
setCharacterSetName	initialize, initializeDoc, initializePage, afp2web, finalizePage

setCodedFontName	initialize, initializeDoc, initializePage, afp2web, finalizePage
setCodePageName	initialize, initializeDoc, initializePage, afp2web, finalizePage
setEncoding	initialize, initializeDoc, initializePage, afp2web, finalizePage
setFamilyName	initialize, initializeDoc, initializePage, afp2web, finalizePage
setHeight	initialize, initializeDoc, initializePage, afp2web, finalizePage
setItalic	initialize, initializeDoc, initializePage, afp2web, finalizePage
setLight	initialize, initializeDoc, initializePage, afp2web, finalizePage
setName	initialize, initializeDoc, initializePage, afp2web, finalizePage
setStrikeover	initialize, initializeDoc, initializePage, afp2web, finalizePage
setUnderline	initialize, initializeDoc, initializePage, afp2web, finalizePage
setWidth	initialize, initializeDoc, initializePage, afp2web, finalizePage

6.8.3.6 a2w::Image Methods

Methods	Available in Script Routines
getXPos	afp2web, finalizePage, finalizeDoc
getYPos	afp2web, finalizePage, finalizeDoc
getAngle	afp2web, finalizePage, finalizeDoc
getName	afp2web, finalizePage, finalizeDoc
getData	afp2web, finalizePage, finalizeDoc
getDataLength	afp2web, finalizePage, finalizeDoc
getCompression	afp2web, finalizePage, finalizeDoc
getBitsPerPixel	afp2web, finalizePage, finalizeDoc
getSamplesPerPixel	afp2web, finalizePage, finalizeDoc
getWidth	afp2web, finalizePage, finalizeDoc
getHeight	afp2web, finalizePage, finalizeDoc

getPresentationWidth	afp2web, finalizePage, finalizeDoc
getPresentationHeight	afp2web, finalizePage, finalizeDoc
getUniqueColorsCount	afp2web, finalizePage, finalizeDoc
remove	afp2web, finalizePage

6.8.3.7 a2w::Index Methods

Methods	Available in Script Routines
getEBCDICName	afp2web, finalizePage, finalizeDoc
getIndexedObjectName	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getIndexedObjectType	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getLevelNumber	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getName	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNameLength	afp2web, finalizePage, finalizeDoc
getSequenceNumber	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getValue	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getValueLength	afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
remove	initialize, initializeDoc, initializePage, afp2web, finalizePage
setLevelNumber	initialize, initializeDoc, initializePage, afp2web, finalizePage
setName	initialize, initializeDoc, initializePage, afp2web, finalizePage
setSequenceNumber	initialize, initializeDoc, initializePage, afp2web, finalizePage
setValue	initialize, initializeDoc, initializePage, afp2web, finalizePage

6.8.3.8 a2w::Kernel Methods

Methods	Available in Script Routines
getExitStatus	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc, finalize

getIndexFilename	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc, finalize
getResourceFilename	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc, finalize
getSpoolFilename	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc, finalize
getVersionAll	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc, finalize
getLastErrorMessages	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc, finalize

6.8.3.9 a2w::Line Methods

Methods	Available in Script Routines
getColor	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getLength	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getWidth	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getXPos	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getYPos	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isHorizontal	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isNegative	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isVertical	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
remove	initialize, initializeDoc, initializePage, afp2web, finalizePage
setColor	initialize, initializeDoc, initializePage, afp2web, finalizePage
setHorizontal	initialize, initializeDoc, initializePage, afp2web, finalizePage
setLength	initialize, initializeDoc, initializePage, afp2web, finalizePage
setNegative	initialize, initializeDoc, initializePage, afp2web, finalizePage
setVertical	initialize, initializeDoc, initializePage, afp2web, finalizePage
setWidth	initialize, initializeDoc, initializePage, afp2web, finalizePage
setXPos	initialize, initializeDoc, initializePage, afp2web, finalizePage

setYPos	initialize, initializeDoc, initializePage, afp2web, finalizePage
---------	--

6.8.3.10 a2w::MediumMap Methods

Methods	Available in Script Routines
getDuplexControl	afp2web, finalizePage, finalizeDoc
getFormdefName	afp2web, finalizePage, finalizeDoc
getHeight	afp2web, finalizePage, finalizeDoc
getName	afp2web, finalizePage, finalizeDoc
getNupControl	afp2web, finalizePage, finalizeDoc
getResolution	afp2web, finalizePage, finalizeDoc
getWidth	afp2web, finalizePage, finalizeDoc

6.8.3.11 a2w::NOP Methods

Methods	Available in Script Routines
getEBCDICValue	initializePage, afp2web, finalizePage, finalizeDoc
getLength	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getValue	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
remove	initialize, initializeDoc, initializePage, afp2web, finalizePage
setValue	initialize, initializeDoc, initializePage, afp2web, finalizePage

6.8.3.12 a2w::Overlay Methods

Methods	Available in Script Routines
getFirstLine	afp2web, finalizePage, finalizeDoc
getFirstOverlay	afp2web, finalizePage, finalizeDoc
getFirstPageSegment	afp2web, finalizePage, finalizeDoc
getFirstText	afp2web, finalizePage, finalizeDoc

getHeight	afp2web, finalizePage, finalizeDoc
getIncludedXPosition	afp2web, finalizePage, finalizeDoc
getIncludedYPosition	afp2web, finalizePage, finalizeDoc
getName	afp2web, finalizePage, finalizeDoc
getNextLine	afp2web, finalizePage, finalizeDoc
getNextOverlay	afp2web, finalizePage, finalizeDoc
getNextPageSegment	afp2web, finalizePage, finalizeDoc
getNextText	afp2web, finalizePage, finalizeDoc
getResolution	afp2web, finalizePage, finalizeDoc
getWidth	afp2web, finalizePage, finalizeDoc

6.8.3.13 a2w::Page Methods

Methods	Available in Script Routines
addAnnotation	initialize, initializeDoc, initializePage, afp2web, finalizePage
addBookmark	initialize, initializeDoc, initializePage, afp2web, finalizePage
addImage	initialize, initializeDoc, initializePage, afp2web, finalizePage
addIndex	initialize, initializeDoc, initializePage, afp2web, finalizePage
addLine	initialize, initializeDoc, initializePage, afp2web, finalizePage
addNOP	initialize, initializeDoc, initializePage, afp2web, finalizePage
addOverlay	initialize, initializeDoc, initializePage, afp2web, finalizePage
addPSEG	initialize, initializeDoc, initializePage, afp2web, finalizePage
addText	initialize, initializeDoc, initializePage, afp2web, finalizePage
getAppliedMediumMap	afp2web, finalizePage, finalizeDoc
getFirstBookmark	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstIndex	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc

getFirstLine	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstMediumOverlay	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstNOP	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstOverlay	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstPageSegment	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getFirstText	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getHeight	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getId	finalizeDoc
getName	initializePage, afp2web, finalizePage, finalizeDoc
getNextBookmark	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextIndex	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextLine	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextMediumOverlay	afp2web, finalizePage, finalizeDoc
getNextNOP	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextOverlay	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextPageSegment	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getNextText	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getOutputBuffer	finalizeDoc
getOutputBufferLength	finalizeDoc
getOutputFilename	initializePage, afp2web, finalizePage, finalizeDoc
getPageGroupName	initializePage, afp2web, finalizePage, finalizeDoc
getParseId	initializePage, afp2web, finalizePage, finalizeDoc
getResolution	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getRotation	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getSimpleFilename	initializePage, afp2web, finalizePage, finalizeDoc

getSize	finalizeDoc
getWidth	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
isConstantBack	afp2web, finalizePage, finalizeDoc
isOverlayFound	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
setBackgroundColor	initialize, initializeDoc, initializePage, afp2web, finalizePage
setHeight	initialize, initializeDoc, initializePage, afp2web, finalizePage
setOutputFilename	initializePage, afp2web, finalizePage
setResolution	initialize, initializeDoc, initializePage, afp2web, finalizePage
setRotation	initialize, initializeDoc, initializePage, afp2web, finalizePage
setWidth	initialize, initializeDoc, initializePage, afp2web, finalizePage
getFirstImage	afp2web, finalizePage, finalizeDoc
getNextImage	afp2web, finalizePage, finalizeDoc
getFirstVector	afp2web, finalizePage, finalizeDoc
getNextVector	afp2web, finalizePage, finalizeDoc

6.8.3.14 a2w::PSEG Methods

Methods	Available in Script Routines
getIncludedXPosition	afp2web, finalizePage, finalizeDoc
getIncludedYPosition	afp2web, finalizePage, finalizeDoc
getName	afp2web, finalizePage, finalizeDoc

6.8.3.15 a2w::Text Methods

Methods	Available in Script Routines
getAngle	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getColor	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc

getEBCDICText	afp2web, finalizePage, finalizeDoc
getFont	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getMappedFontLocalId	initializePage, afp2web, finalizePage, finalizeDoc
getText	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getTextLen	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getXPos	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getYPos	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
remove	initialize, initializeDoc, initializePage, afp2web, finalizePage
setAngle	initialize, initializeDoc, initializePage, afp2web, finalizePage
setColor	initialize, initializeDoc, initializePage, afp2web, finalizePage
setFont	initialize, initializeDoc, initializePage, afp2web, finalizePage
setText	initialize, initializeDoc, initializePage, afp2web, finalizePage
setEBCDICText	initializePage, afp2web, finalizePage
setTextLen	initialize, initializeDoc, initializePage, afp2web, finalizePage
setXPos	initialize, initializeDoc, initializePage, afp2web, finalizePage
setYPos	initialize, initializeDoc, initializePage, afp2web, finalizePage

6.8.3.16 a2w::UserData Methods

Methods	Available in Script Routines
getData	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getDataLength	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
getId	initialize, initializeDoc, initializePage, afp2web, finalizePage, finalizeDoc
new	initialize, initializeDoc, initializePage, afp2web, finalizePage
setData	initialize, initializeDoc, initializePage, afp2web, finalizePage

setDataLength	initialize, initializeDoc, initializePage, afp2web, finalizePage
setId	initialize, initializeDoc, initializePage, afp2web, finalizePage

6.8.3.17 a2w::Vector Methods

Methods	Available in Script Routines
getXPos	afp2web, finalizePage, finalizeDoc
getYPos	afp2web, finalizePage, finalizeDoc
getAngle	afp2web, finalizePage, finalizeDoc
getWidth	afp2web, finalizePage, finalizeDoc
getHeight	afp2web, finalizePage, finalizeDoc
getUniqueColorsCount	afp2web, finalizePage, finalizeDoc
remove	afp2web, finalizePage

6.9 Scripting Facility API Reference

6.9.1 Overview

Conventions used in this document

- "[NEW]" indicates a new module or interface.
- "[DEPRECATED]" indicates that a module or interface, which is no longer recommended. It will be removed in the future. The same functionality can be achieved using another new generic interface.
- "[MODIFIED]" indicates a change to an existing module or interface.
- "[EXTENDED]" indicates additional functionality.

AFP2web provides "a2w:*Constants" Scripting Facility modules where Constants are defined to maintain a standard on parameter values passed to the various Scripting Facility module provided APIs and AFP2web recommend to use module defined constants during Scripting Facility customizations.

6.9.2 afp2web.pm

The module afp2web.pm is the gateway script of the AFP2web Scripting Facility and this module will be loaded and executed by the AFP2web Kernel (using an embedded PERL interpreter). The description below lists the interfaces exposed by the module afp2web.pm and which can be used to communicate with the AFP2web Kernel.

6.9.2.1 afp2web

Parameter:

None

Return value data type:

Integer

Remarks:

Called while processing (after binding resources) each output page.

At this stage, it is possible to analyze page content and, by setting the appropriate return code, to instruct the AFP2web Kernel to change its flow.

One of the following return values are possible:

Return value	Description
<0	A negative value indicates an error to AFP2web. When returning an error code to AFP2web, it is also possible to pass an error message for a more descriptive error report. NOTE: AFP2web will abort process when negative value is returned from this API and returned error code/message will be given as exception message
0	Append page to current document
1	Skip page
2	First page / New document
4	Last page (complete current document with this page)

Example:

```
sub afp2web(){
    $APPEND = 0;# append page to Current Document
    $SKIP = 1;# skip page
    $NEWDOC = 2;# new document
    $LASTPAGE = 4; last page of current document
    $iRetTmp = $APPEND; # default: append page
    #---- Do something
    ...
    #---- Return value
}
```

```

        if ( $iProcessOk ){
            return $iRetTmp; #---- Success
        } else {
            return (-1, "afp2web failed" );#---- Error
        }
    }
}

```

6.9.2.2 finalize

Parameter:

None

Return value data type:

Integer

Remarks:

Called after processing each spool. This routine is used for spool finalizing.

The return value must be zero or greater than zero. A negative value indicates an error to AFP2web. When returning an error code to AFP2web, it is also possible to pass an error message for a more descriptive error report.

NOTE: AFP2web will abort process when negative value is returned from this API and returned error code/message will be given as exception message

Example:

```

sub finalize (){
    #---- Do something
    ...
    #---- Return value
    if ( $iProcessOk ){
        return 0; #---- Success
    } else {
        return (-1, "finalize failed" ); #---- Error
    }
}

```

6.9.2.3 finalizeDoc

Parameter:

None

Return value data type:

Integer

Remarks:

Called after processing each output document. This routine is used for document finalizing.

The return value must be zero or greater than zero. A negative value indicates an error to AFP2web. When returning an error code to AFP2web, it is also possible to pass an error message for a more descriptive error report.

NOTE: AFP2web will abort process when negative value is returned from this API and returned error code/message will be given as exception message

Example:

```
sub finalizeDoc(){
    #---- Do something
    ...
    #---- Return value
    if ( $iProcessOk ){
        return 0; #---- Success
    } else {
        return (-1, "finalizeDoc failed" ); #---- Error
    }
}
```

6.9.2.4 finalizePage**Parameter:**

None

Return value data type:

Integer

Remarks:

Called after processing each output page. This routine is used for page finalizing.

The return value must be zero or greater than zero. A negative value indicates an error to AFP2web. When returning an error code to AFP2web, it is also possible to pass an error message for a more descriptive error report.

NOTE: AFP2web will abort process when negative value is returned from this API and returned error code/message will be given as exception message

Example:

```
sub finalizePage(){
    #---- Do something
    ...
    #---- Return value
    if ( $iProcessOk ){
        return 0; #---- Success
    } else {
        return (-1, "finalizePage failed" ); #---- Error
    }
}
```

6.9.2.5 initialize

Parameter:

Configuration instance of type a2w::Config

Kernel instance of type a2w::Kernel

Return value data type:

Integer

Remarks:

Called at the beginning of each input spool. Using a configuration instance, it is possible to read and modify configuration parameters (which were entered in INI file or as command line options). Using an instance of the kernel object, it is possible to read information about the input spool.

The return value must be zero or greater than zero. A negative value indicates an error to AFP2web. When returning an error code to AFP2web, it is also possible to pass an error message for a more descriptive error report.

NOTE: AFP2web will abort process when negative value is returned from this API and returned error code/message will be given as exception message

Example:

```
sub initialize(){
    #---- Get parameters of initialize ( a2w::Config, a2w::Kernel )
    ( $a2wConfigPar, $a2wKernelPar ) = @_ ;
    #---- Do something
    ...
    #---- Return value
    if ( $iProcessOk ){
        return 0;#---- Success
    } else {
        return (-1, "Initialization failed" );#---- Error
    }
}
```

6.9.2.6 initializeDoc

Parameter:

Document instance of type a2w::Document

Return value data type:

Integer

Remarks:

Called at the beginning of each output document process, the user can initialize output document dependent processes.

The return value must be zero or greater than zero. A negative value indicates an error to AFP2web. When returning an error code to

AFP2web, it is also possible to pass an error message for a more descriptive error report.

NOTE: AFP2web will abort process when negative value is returned from this API and returned error code/message will be given as exception message

Example:

```
sub initializeDoc(){
    #---- Get parameters of initializeDoc ( a2w::Document )
    ( $a2wDocumentPar ) = @_;
    #---- Do something
    ...
    #---- Return value
    if ( $iProcessOk ){
        return 0;#---- Success
    } else {
        return (-1, "initializeDoc failed" );#---- Error
    }
}
```

6.9.2.7 initializePage

Parameter:

Page instance of type a2w::Page

Return value data type:

Integer

Remarks:

Called at the beginning of each output page process, user can initialize output page dependent processes.

The return value must be zero or greater than zero. A negative value indicates an error to AFP2web. When returning an error code to AFP2web, it is also possible to pass an error message for a more descriptive error report.

NOTE: AFP2web will abort process when negative value is returned from this API and returned error code/message will be given as exception message

Example:

```
sub initializePage(){
    #---- Get parameters of initializePage ( a2w::Page )
    ( $a2wPagePar ) = @_;
    #---- Do something
    ...
    #---- Return value
}
```

```

        if ( $iProcessOk ){
            return 0;#---- Success
        } else {
            return (-1, "initializePage failed" );#---- Error
        }
    }
}

```

6.9.3 a2w::Bookmark

"a2w::Bookmark" module provides input spool bookmark handle and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.3.1 addChild

Parameter:

Bookmark instance of type a2w::Bookmark

Return value data type:

Void

Remarks:

Adds a child to the bookmark.

Example:

```

#--- Get a Parent Bookmark
$a2wBookmark = $a2wPage->getFirstBookmark();
#--- Create a new Bookmark
$a2wChildBookmark = new a2w::Bookmark();
#--- Set Title text of Bookmark
$a2wChildBookmark->setTitle( "Insured" );
#--- Set Page as Target type for Bookmark
$a2wChildBookmark->setTargetType( 0 );
#--- Set Page 1 as Target for Bookmark
$a2wChildBookmark->setTargetName( "1" );
....
#--- Add child Bookmark to the Parent Bookmark
$a2wBookmark->addChild( $a2wChildBookmark );

```

6.9.3.2 getColor

Parameter:

None

Return value data type:

Integer

Remarks:

Gets Color of bookmark. Value should be interpreted in RGB color space as given below:



Example:

```
#--- Get the color of bookmark
$iColorTmp = $a2wBookmark->getColor();
```

6.9.3.3 getFirstChild

Parameter:

None

Return value data type:

Bookmark instance of type a2w::Bookmark

Remarks:

Gets first child of bookmark.

Example:

```
$a2wChildTmp = $a2wBookmark->getFirstChild();
while ($a2wChildTmp != 0 ){
    #---- Access Child info
    ...
    #---- Get next Child
    $a2wChildTmp = $a2wBookmark->getNextChild();
}
```

6.9.3.4 **getLevelNumber**

Parameter:

None

Return value data type:

Integer

Remarks:

Gets level number of bookmark.

Example:

```
#--- Fetch the bookmark level number
$iLevelNoTmp = $a2wBookmark->getLevelNumber();
```

6.9.3.5 **getNextChild**

Parameter:

None

Return value data type:

Bookmark instance of type a2w::Bookmark

Remarks:

Gets next child of bookmark. This method is normally used in an iteration loop after using `getFirstChild`.

Example:

```
$a2wChildTmp = $a2wBookmark->getFirstChild();
while ($a2wChildTmp != 0 ){
    #---- Access Child info
    ...
    #---- Get next Child
    $a2wChildTmp = $a2wBookmark->getNextChild();
}
```

6.9.3.6 **getTargetName**

Parameter:

None

Return value data type:

String

Remarks:

Gets target name of bookmark.

Example:

```
#--- Fetch the target name
$sTargetNameTmp = $a2wBookmark->getTargetName();
```

6.9.3.7 **getTargetType**

Parameter:

None

Return value data type:

Buffer of type byte

Remarks:

Gets target type of bookmark.

Example:

```
#--- Fetch the target type
$byTargetTypeTmp = $a2wBookmark->getTargetType();
```

6.9.3.8 **getTitle**

Parameter:

None

Return value data type:

String

Remarks:

Gets title of the bookmark.

Example:

```
#--- Fetch the bookmark title
$sTitleTmp = $a2wBookmark->getTitle();
```

6.9.3.9 isBoldFont**Parameter:**

None

Return value data type:

Boolean

Remarks:

Returns true if bookmark font is bold.

Example:

```
#--- Check the Font is Bold or not
$bBoldFontTmp = $a2wBookmark->isBoldFont();
```

6.9.3.10 isItalicFont**Parameter:**

None

Return value data type:

Boolean

Remarks:

Returns true if bookmark font is italic.

Example:

```
#--- Check the Font is Italic or not
$bItalicFontTmp = $a2wBookmark->isItalicFont();
```

6.9.3.11 new

Parameter:

None

Return value data type:

Bookmark instance of type a2w::Bookmark

Remarks:

Allocate new bookmark instance.

Example:

```
#--- Create a new Bookmark
$a2wBookmark = new a2w::Bookmark();
```

6.9.3.12 remove

Parameter:

None

Return value data type:

Void

Remarks:

Removes the bookmark from the presentation data.

Example:

```
#--- Get First Bookmark of the page
$a2wBookmark = $a2wPage->getFirstBookmark();
#--- Assert Bookmark
if( $a2wBookmark ){
    #--- Remove Bookmark
    $a2wBookmark->remove();
}
```

6.9.3.13 setBoldFont

Parameter:

Bold font of type boolean

Return value data type:

Void

Remarks:

Set bold font for the bookmark.

Example:

```
#--- Set the bold font is true
$a2wBookmark->setBoldFont( 1 );
```

6.9.3.14 setColor

Parameter:

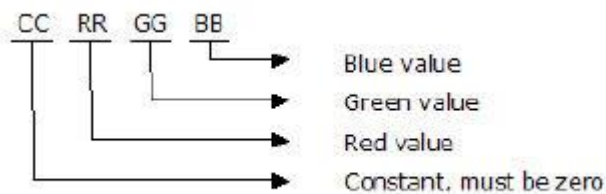
Color in RGB Format of type integer

Return value data type:

Void

Remarks:

Set bold font for the bookmark.Sets the color of Bookmark in RGB Colorspace Color value is in <0xCCRRGGBB> where:

**Example:**

```
#--- Set the color of Bookmark as Blue
$a2wBookmark->setColor( 0x000000FF );
```

6.9.3.15 setItalicFont

Parameter:

Italic font of type boolean

Return value data type:

Void

Remarks:

Set italic font for the bookmark.

Example:

```
#--- Set the italic font is true
$a2wBookmark->setItalicFont( 1 );
```

6.9.3.16 **setLevelNumber**

Parameter:

Level number of type Integer

Return value data type:

Void

Remarks:

Sets level number of bookmark.

Example:

```
#--- Create a new Bookmark
$a2wBookmark = new a2w::Bookmark();
#--- Set Level number of Bookmark
$a2wBookmark->setLevelNumber( 1 );
```

6.9.3.17 **setTargetName**

Parameter:

Target name of type string

Return value data type:

Void

Remarks:

Set target name of bookmark.

Example:

```
#--- Set Page as Target name for Bookmark  
$a2wBookmark->setTargetName( "Section_1" );
```

6.9.3.18 setTargetType**Parameter:**

TargetType of type byte:

0 => Page

1 => Page Group

Return value data type:

Void

Remarks:

Sets target type of bookmark..

Example:

```
#--- Set target type of Bookmark  
$a2wBookmark = new a2w::Bookmark();  
#--- Set Page as Target type for Bookmark  
$a2wBookmark->setTargetType( 0 );
```

6.9.3.19 setTitle**Parameter:**

Title of type String

Return value data type:

Void

Remarks:

Sets title of bookmark.

Example:

```
#--- Create a new Bookmark  
$a2wBookmark = new a2w::Bookmark();  
#--- Set Title text of Bookmark  
$a2wBookmark->setTitle( "Insured" );
```

6.9.4 a2w::Comment [NEW]

The a2w::Comment module exposes the following interfaces to the AFP2web Scripting Facility. User can use this module to add comments on output. Comment value will be added on output following appropriate output format syntax. i.e, for PDF output given value will be added on PDF stream as given below

%<comment value>

PDF comment starts with “%” character, so user given comment value is prefixed with that character and written on PDF output

6.9.4.1 getValue [NEW]

Parameter:

None

Return value data type:

String

Remarks:

Get the Comment's value

Example:

```
use a2w::Comment;
use a2w::Document;
#...
sub finalizeDoc{
    #...
    #---- Get document comment
    my $a2wCommentTmp = $a2wDocumentPar->getFirstComment(),

    #---- Get comment value
    my $sCommentValueTmp = $a2wCommentTmp->getValue();
    #...
}
```

6.9.4.2 new [NEW]

Parameter:

None

Return value data type:

Comment instance of type a2w::Comment

Remarks:

Returns newly allocated instance of a2w::Comment

Example:

```
use a2w::Comment;
use a2w::Document;
#...
sub initializeDoc{
    #...
    #--- Create a new comment
    $a2wComment = new a2w::Comment();
}
```

6.9.4.3 remove [NEW]

Parameter:

None

Return value data type:

Void

Remarks:

Removes Comment from it's container (Page or Document)

Example:

```
use a2w::Comment;
use a2w::Document;
#...
sub afp2web{
    #...
    #---- Remove comment from document
    my $a2wCommentTmp = $a2wDocumentPar->getFirstComment();
    $a2wCommentTmp->remove();
}
```

6.9.4.4 setValue [NEW]

Parameter:

Comment value of type String

Return value data type:

Void

Remarks:

Set Comment value

Example:

```
use a2w::Comment;
use a2w::Document;
#...
sub afp2web{
    #...
    #---- Create comment
    my $a2wCommentTmp = new a2w::Comment();

    #---- Set comment value
    $a2wCommentTmp->setValue( "Test Comment added by AFP2web" );
    #...
}
```

6.9.5 a2w::Config

The a2w::Config module allows to set/alter/access the ini configuration attributes and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.5.1 getAttribute

Parameter:

Attribute name of type string

Return value data type:

String

Remarks:

Gets the current value of the INI attribute.

Example:

```
$sIndexFilePathTmp = $a2wConfigPar->getAttribute( $a2w::ConfigConstants::INDEXPATH );
```

6.9.5.2 setAttribute

Parameter:

Attribute name of type string

Attribute value of type string

Return value data type:

Void

Remarks:

Sets the INI attribute (the processing parameter, which is originally set in the configuration file called the INI file or as command line option).

Example:

```
$a2wConfigPar->setAttribute( $a2w::ConfigConstants::QUIETMODE, "on" );
```

6.9.6 a2w::ConfigConstants [NEW]

"a2w::ConfigConstants" module defines constants for AFP2web's afp2web.ini parameters. All parameters names are the same as within afp2web.ini but in UPPERCASE. For further details, please refer to [INI Parameter Reference](#).

This module defined constants are advised to use while using "a2w::Config" module provided APIs.

6.9.6.1 [Settings] Section Constants

[Settings] section INI parameter name constants

Constants in **BOLD** text **MUST NOT BE USED TO MODIFY** the INI parameter value, **MUST BE USED ONLY TO GET** the INI parameter value only.

Constant	Value	Type	Description
\$INIPATH	IniFilePath	String	INI file path
\$INIFILENAME	IniFilename	String	INI file name

\$LICENSEE	Licensee	String	Licensee
\$SERIALNR	SerialNr	String	Serial Number
\$TITLE	Title	String	Output document's Title property
\$SUBJECT	Subject	String	Output document's Subject property
\$KEYWORDS	Keywords	String	Output document's Keywords property
\$LOGGING	Logging	String	To turn on/off logging
\$LOGGINGFONT	LoggingFont	String	To turn on/off font logging
\$LOGGINGLEVEL	LoggingLevel	String	To set logging levels
\$EXCEPTIONLOGGINGLEVEL	ExceptionLoggingLevel	String	To set exception logging level
\$LOGPATH	LogPath	String	Log file path
\$RESPATH	ResPath	String	Resource file path
\$CPPATH	CpPath	String	ASCII code page path
\$CODEPAGE	CodePage	String	Code page
\$OUTPUTFILEPATH	OutputFilePath	String	Output file path
\$INDEXPATH	IndexPath	String	Index path
\$INDEXFORMAT	IndexFormat	String	Index file format (XML or CSV)
\$INDEXRECORD	IndexRecord	String	Index record format for CSV format
\$PAGEPARSE	PageParse	String	Parse page up to given number of pages
\$OUTPUTFORMAT	OutputFormat	String	Output format
\$INPUTFORMAT	InputFormat	String	Input format
\$RESOLUTION	Resolution	String	Output page resolution
\$FORMATTYPE	FormatType	String	Format type of output
\$IOMODE	IOMode	String	IO mode to read in from STDIN and to write out on STDOUT
\$COLOR	Color	String	To turn on color output

\$PAGEOUTPUT	PageOutput	String	To generate single page output documents
\$STATISTIC	Statistic	String	To turn on conversion statistics
\$PROTOCOL	Protocol	String	To turn on conversion protocol
\$PDFCOMPRESSION	PDFCompression	String	To generate compressed PDF or not
\$FILECREATIONMODE	FileCreationMode	String	File creation mode
\$PAGELENGTH	PageLength	String	Maximum page length
\$QUIETMODE	QuietMode	String	To turn on/off console messages
\$CREATOR	Creator	String	Output document's Creator property
\$EXTFONTPATH	ExtFontPath	String	External font path
\$AFPFontPATH	AFPFontPath	String	AFP font path
\$GENERATEUNIQUEFILE	GenerateUniqueFile	String	To generate unique output filename (to avoid overwriting during conversion)
\$PDFDOCLIMITS	PDFDocLimits	String	PDF document limits
\$PDFUIOPTIONS	PDFUIOptions	String	PDF user interface options
\$PDFWINOPTIONS	PDFWinOptions	String	PDF windows opening options
\$FORMDEFEXT	FormdefExt	String	AFP Form Definition resource extension
\$OVERLAYEXT	OverlayExt	String	AFP Overlay resource extension
\$PAGESEGEXT	PageSegExt	String	AFP Page Segment resource extension
\$CHARSETEXT	CharSetExt	String	AFP Character Set resource extension
\$CODEPAGEEXT	CodePageExt	String	AFP Code Page resource extension
\$CODEDFONTEXT	CodedFontExt	String	AFP Coded Font resource extension
\$CODEPAGEDEFAULTCHAR	CodePageDefaultChar	String	Code page default character
\$LAUNCHPREVIEW	LaunchPreview	String	Launch preview of output document after conversion (using system default application associated with output format)
\$PDFBOOKMARK	PDFBookMark	String	To turn on PDF bookmarks on output PDF

\$MAXCOLS	MaxCols	String	Maximum columns for ASCII output
\$MAXROWS	MaxRows	String	Maximum rows for ASCII output
\$OVERLAYPATH	OverlayPath	String	AFP Overlay resource path
\$PAGESEGMENTPATH	PageSegmentPath	String	AFP Page Segment resource path
\$FORMDEFPATH	FormDefPath	String	AFP Form Definition resource path
\$STRICT	Strict	String	To turn on strict option, AFP2web will stop conversion when conversion required resource is missing/not found
\$DOCUMENTCOUNT	DocumentCount	String	Output document count, output document will be grouped on sub directories in output file path based on this count
\$STARTINGDOCUMENT	StartingDocument	String	Starting document id, from which the conversion must start
\$ENDINGDOCUMENT	EndingDocument	String	Ending document id, at which the conversion must end
\$STARTINGPAGE	StartingPage	String	Starting page, from which the conversion must start
\$ENDINGPAGE	EndingPage	String	Ending page, at which the conversion must end
\$JPEGQUALITY	JPEGQuality	String	Quality for JPEG output
\$WATERMARK	Watermark	String	Watermark text that has to written on output pages
\$SAVEOCDATAONDISK	SaveOCDataOnDisk	String	Optimize memory usage by storing big AFP Container Object data on disk
\$TEMPPATH	TempPath	String	Temporary file path used by AFP2web during conversion process
\$MARKER_HEIGHT	MarkerHeight	String	AFP GOCA marker height
\$MARKER_WIDTH	MarkerWidth	String	AFP GOCA marker width
\$SCRIPTARGUMENT	ScriptArgument	String	Script arguments passed to Scripting Facility module
\$SCRIPTPROCEDURE	ScriptProcedure	String	Script procedure used for Scripting Facility

\$SCRIPTUNITBASE	ScriptUnitBase	String	Unit base used to set/get measurement parameters from/to Scripting Facility APIs
\$FILENAMEPATTERN	FilenamePattern	String	Output file name pattern
\$SKIPPAGE	SkipPage	String	Skip output page based on skip object size
\$SKIPOBJECTSIZE	SkipObjectSize	String	Skip object size
\$PDFSECURITY	PDFSecurity	String	Parameters to secure output PDF
\$CMYKTORGB	CMYKTORGB	String	Convert CMYK images to RGB images
\$AUTOSPLIT	Autosplit	String	To turn on auto split
\$MEMORYOUTPUTSTREAM	MemoryOutputStream	String	To turn on memory output stream (instead of generating output on file system)
\$RESOURCEGROUP	ResourceGroup	String	AFP output resource group generation controller
\$IMAGE_RESIZE_FILTER	ImageResizeFilter	String	Filter to be used for image resizing process
\$IMAGE_ROT_FILTER	ImageRotationFilter	String	Filter to be used for image rotation process
\$COLORSPACE	Colorspace	String	Color space
\$CODEDOUTPUT	CodedOutput	String	Coded output or non coded output
\$FLUSH	Flush	String	Flush pages instead of caching them on memory
\$PDFALEVEL	PDFALevel	String	Level of PDF/A output
\$SPOOLFILETYPE	SpoolFileType	String	Spool file type for LPD and MMD inputs
\$RESOURCEGROUPFILENAME	ResourceGroupFilename	String	AFP resource group file name to be used for given AFP spool
\$DOCUMENTINDEXFILENAME	DocumentIndexFilename	String	AFP document index file name to be used for given AFP spool
\$FORMDEFFILENAME	FormdefFilename	String	AFP form definition file name to be used for given AFP/MMD spool
\$PAGEROTATION	PageRotation	String	Output page rotation
\$IMAGEMAXCACHESIZE	ImageMaxCacheSize	String	Maximum image cache size
\$FONTRESOLUTION	FontResolution	String	Font resolution

\$AFPCOMPLIANCELEVEL	AFPComplianceLevel	String	AFP compliance level to generate AFP outputs
\$MAXVECTOROBJECTS	MaxVectorObjects	String	Maximum vector objects on page
\$FONTALIASING	FontAliasing	String	Font aliasing
\$OUTPUTSIZE	OutputSize	String	Output page size
\$NOCONSOLEMESSAGE	NoConsoleMessage	String	Do not pass on any messages to console
\$IMAGEPROCESSVALUES	ImageProcessValues	String	Default values for missing input image properties like resolution.

6.9.7 a2w::Document [EXTENDED]

a2w::Document module provides document handle and exposes the following interfaces to the Scripting Facility.

6.9.7.1 addBookmark

Parameter:

Bookmark name of type string

Bookmark value of type string

Return value data type:

Void

Remarks:

Adds a bookmark to the document. Valid only for PDF output

Example:

```
$a2wDocumentPar->addBookmark( "Software", "AFP2web" );
```

6.9.7.2 addComment [NEW]

Parameter:

Comment instance of type a2w::Comment.

Return value data type:

Void

Remarks:

Adds a Comment element to the document. The Comment element is inserted based on output format.
For PDF output, comments are added before "trailer" dictionary

Example:

```
use a2w::Comment;
use a2w::Document;
#...
sub initializeDoc{
    #...
    #---- Create new Comment
    my $a2wCommentTmp = new a2w::Comment();
    #---- Set value
    $a2wCommentTmp->setValue( "Test comment added by AFP2web" );
    #---- Add comment to document
    $a2wDocumentPar->addComment( $a2wCommentTmp );
}
```

6.9.7.3 addIndex

Parameter:

Index instance of type a2w::Index.

Return value data type:

Void

Remarks:

Adds an index element to the document.

Example:

```
#---- Create new index
$a2wIndexTmp = new a2w::Index();
#---- Set index info
$a2wIndexTmp->setName( "Producer" );
$a2wIndexTmp->setValue( "Maas Holding GmbH" );
#---- Add index to document
$a2wDocumentPar->addIndex( $a2wIndexTmp );
```

6.9.7.4 addNOP

Parameter:

NOP instance of type a2w::NOP

Return value data type:

Void

Remarks:

Adds an NOP element to the document. The NOP element is inserted before first BPG of the document and valid only for AFP output

Example:

```
#---- Create new NOP
$a2wNOPTmp = new a2w::NOP();
#---- Set NOP value
$a2wNOPTmp->setValue( "Created by AFP2web" );
#---- Add NOP to document
$a2wDocumentPar->addNOP( $a2wNOPTmp );
```

6.9.7.5 addPage

Parameter:

Page instance of type a2w::Page

Page identifier of the output document of type unsigned integer

Return value data type:

Void

Remarks:

Adds a page to the document.

Example:

```
#---- Create new Page
$a2wPageTmp = new a2w::Page();
#---- Add some content
$a2wPageTmp->addText( $a2wTextTmp );
#---- Add page to document as second page
$a2wDocumentPar->addPage( $a2wPageTmp, 2 );
```

6.9.7.6 addUserData

Parameter:

UserData instance of type a2w::UserData.

Return value data type:

Void

Remarks:

Adds an user data object to the document. Useful for Java SDK APIs, where Scripting Facility added user data can be accessed through "A2WDocumentHandler::handle" call back implementation.

Example:

```
#---- Create new user data
$a2wUserDataTmp = new a2w::UserData();
#---- Set user data id
$a2wUserDataTmp->setId( "DOCINDEX" );
#---- Create document index
$sIndexDataTmp = "Insured=Geoffrey R Stephens";
#---- Set document index as data
$a2wUserDataTmp->setData( $sIndexDataTmp, length( $sIndexDataTmp ) );
#---- Add user data to document
$a2wDocumentPar->addUserData( $a2wUserDataTmp );
```

6.9.7.7 getFirstBookmark

Parameter:

None

Return value data type:

Bookmark instance of type a2w::Bookmark

Remarks:

Gets the first bookmark at the document level.

Example:

```
$a2wBookmarkTmp = $a2wDocumentPar->getFirstBookmark();
while ( $a2wBookmarkTmp != 0 ){
    #---- Access bookmark info
    ...
    #---- Get next bookmark
    $a2wBookmarkTmp = $a2wDocumentPar->getNextBookmark();
}
```

6.9.7.8 **getFirstComment [NEW]**

Parameter:

None

Return value data type:

Comment instance of type a2w::Comment

Remarks:

Gets the first comment at the document level.

Example:

```
use a2w::Comment;
use a2w::Document;
#...
sub finalizeDoc{
    #...
    #---- Loop through document level comments and process them ----#
    #---- Fetch first document level comment
    $a2wCommentTmp = $a2wDocumentPar->getFirstComment();

    #---- Loop all comments
    while ( $a2wCommentTmp != 0 ){
        #---- Access comment info
        #...
        #---- Fetch next document level comment
        $a2wCommentTmp = $a2wDocumentPar->getNextComment();
    }
}
```

6.9.7.9 **getFirstIndex**

Parameter:

None

Return value data type:

Index instance of type a2w::Index

Remarks:

Gets the first index at the document level.

Example:

```
$a2wIndexTmp = $a2wDocumentPar->getFirstIndex();
while ($a2wIndexTmp != 0 ){
    #---- Access index info
    ...
    #---- Get next index
    $a2wIndexTmp = $a2wDocumentPar->getNextIndex();
}
```

6.9.7.10 getFirstPage**Parameter:**

None

Return value data type:

Page instance of type a2w::Page

Remarks:

Gets the first page of the document.

Example:

```
$a2wPageTmp = $a2wDocumentPar->getFirstPage();
while ($a2wPageTmp != 0 ){
    #---- Access page info
    ...
    #---- Get next page
    $a2wPageTmp = $a2wDocumentPar->getNextPage();
}
```

6.9.7.11 getFirstUserData**Parameter:**

None

Return value data type:

UserData instance of type a2w::UserData

Remarks:

Gets the first user data object at the document level. Useful for Java SDK APIs, where Scripting Facility added user data can be accessed through "A2WDocumentHandler:: handle" call back implementation.

Example:

```
$a2wUserDataTmp = $a2wDocumentPar->getFirstUserData();  
while ($a2wUserDataTmp != 0 ){  
    #---- Access user data info  
    ...  
    #---- Get next user data  
    $a2wUserDataTmp = $a2wDocumentPar->getNextUserData();  
}
```

6.9.7.12 getId**Parameter:**

None

Return value data type:

Integer

Remarks:

Gets the document ID.

Example:

```
$iDocIdTmp = $a2wDocumentPar->getId();
```

6.9.7.13 getMetaDataStream [NEW]**Parameter:**

None

Return value data type:

String

Remarks:

PDF input only, returns the XMP metadata stream of input PDF

Example:

```
use a2w::Document;  
use a2w::DocumentConstants;  
#...  
sub initializeDoc(){
```

```
#...
my $sMetaDataStreamTmp = $a2wDocumentPar->getMetaDataStream();
# PROCESS META DATA STREAM CONTENT
}
```

6.9.7.14 **getName**

Parameter:

None

Return value data type:

String

Remarks:

Gets the document name.

Example:

```
$sDocNameTmp = $a2wDocumentPar->getName();
```

6.9.7.15 **getNextBookmark**

Parameter:

None

Return value data type:

Bookmark instance of type a2w::Bookmark

Remarks:

Gets the next bookmark at the document level. This method is normally used in an iteration loop after using `getFirstBookmark`.

Example:

```
$a2wBookmarkTmp = $a2wDocumentPar->getFirstBookmark();
while ($a2wBookmarkTmp != 0 ){
    #---- Access bookmark info
    ...
    #---- Get next bookmark
    $a2wBookmarkTmp = $a2wDocumentPar->getNextBookmark();
}
```

6.9.7.16 getNextComment [NEW]

Parameter:

None

Return value data type:

Comment instance of type a2w::Comment

Remarks:

Gets the next comment at the document level. This method is normally used in an iteration loop after first using `getFirstComment`.

Example:

```
use a2w::Comment;
use a2w::Document;
#...
sub finalizeDoc{
    #...
    #---- Loop through document level comments and process them ----#
    #---- Fetch first document level comment
    $a2wCommentTmp = $a2wDocumentPar->getFirstComment();

    #---- Loop all comments
    while ( $a2wCommentTmp != 0 ){
        #---- Access comment info
        #...
        #---- Fetch next document level comment
        $a2wCommentTmp = $a2wDocumentPar->getNextComment();
    }
}
```

6.9.7.17 getNextIndex

Parameter:

None

Return value data type:

Index instance of type a2w::Index

Remarks:

Gets the next index at the document level. This method is normally used in an iteration loop after first using `getFirstIndex`.

Example:

```
$a2wIndexTmp = $a2wDocumentPar->getFirstIndex();
while ($a2wIndexTmp != 0 ){
    #---- Access index info
    ...
    #---- Get next index
    $a2wIndexTmp = $a2wDocumentPar->getNextIndex();
}
```

6.9.7.18 getNextPage**Parameter:**

None

Return value data type:

Page instance of type a2w::Page

Remarks:

Gets the next page of the document. This method is normally used in an iteration loop after first using `getFirstPage`.

Example:

```
$a2wPageTmp = $a2wDocumentPar->getFirstPage();
while ($a2wPageTmp != 0 ){
    #---- Access page info
    ...
    #---- Get next page
    $a2wPageTmp = $a2wDocumentPar->getNextPage();
}
```

6.9.7.19 getNextUserData**Parameter:**

None

Return value data type:

UserData instance of type a2w::UserData

Remarks:

Gets the next user data object at the document level. This method is normally used in an iteration loop after using `getFirstUserData`.

Useful for Java SDK APIs, where Scripting Facility added user data can be accessed through "A2WDocumentHandler::handle" call back implementation.

Example:

```
$a2wUserDataTmp = $a2wDocumentPar->getFirstUserData();
while ($a2wUserDataTmp != 0 ){
    #---- Access user data info
    ...
    #---- Get next user data
    $a2wUserDataTmp = $a2wDocumentPar->getNextUserData();
}
```

6.9.7.20 **getOutputBuffer**

Parameter:

None

Return value data type:

Buffer of type byte

Remarks:

Gets the document's output buffer. Valid only when "MemoryOutputStream" INI attribute is turned on.

Example:

```
$byDocBufferTmp = $a2wDocumentPar->getOutputBuffer();
```

6.9.7.21 **getOutputBufferLength**

Parameter:

None

Return value data type:

Integer

Remarks:

Gets the length of the document's output buffer. Valid only when "MemoryOutputStream" INI attribute is turned on.

Example:

```
$iDocBufferLenTmp = $a2wDocumentPar->getOutputBufferLength();
```

6.9.7.22 **getOutputFilename**

Parameter:

None

Return value data type:

String

Remarks:

Gets the name of the output file.

Example:

```
$sOutputFilenameTmp = $a2wDocumentPar->getOutputFilename();
```

6.9.7.23 **getPageCount**

Parameter:

None

Return value data type:

Integer

Remarks:

Gets the document's page count.

Example:

```
$iPageCountTmp = $a2wDocumentPar->getPageCount();
```

6.9.7.24 **getPID**

Parameter:

None

Return value data type:

String

Remarks:

Gets the process ID.

Example:

```
$sProcessIdTmp = $a2wDocumentPar->getPID();
```

6.9.7.25 **getProperty [NEW]**

Parameter:

1. Document property name (Constants defined in "[a2w::DocumentConstants](#)" module)

Return value data type:

String

Remarks:

PDF input only, returns the queried input PDF document property value

Example:

```
use a2w::Document;
use a2w::DocumentConstants;
#...
sub initializeDoc(){
    #....
    #---- Fetch document property "Title"
    my $sTitleTmp = $a2wDocumentPar->getProperty( $a2w::DocumentConstants::TITLE );
}
```

6.9.7.26 **getSimpleFilename**

Parameter:

None

Return value data type:

String

Remarks:

Gets the simple file name (filename without path and extension) of the output file.

Example:

```
$sSimpleFilenameTmp = $a2wDocumentPar->getSimpleFilename();
```

6.9.7.27 getSimpleResourceGroupName**Parameter:**

None

Return value data type:

String

Remarks:

Gets the simple name of the output AFP resource group file. This method applies only for AFP output.

Example:

```
$sSimpleRGnameTmp = $a2wDocumentPar->getSimpleResourceGroupName();
```

6.9.7.28 getSize**Parameter:**

None

Return value data type:

Long

Remarks:

Gets the file size of the output document.

Example:

```
$lDocSizeTmp = $a2wDocumentPar->getSize();
```

6.9.7.29 **getUserData**

Parameter:

User data id of type string

Return value data type:

UserData instance of type a2w::UserData

Remarks:

Gets the user data object for the given user data id. Useful for Java SDK APIs, where Scripting Facility added user data can be accessed through "A2WDocumentHandler:: handle" call back implementation.

Example:

```
$a2wUserDataTmp = $a2wDocumentPar->getUserData( "DataId_1" );
```

6.9.7.30 **setName**

Parameter:

Document name of type string

Return value data type:

Void

Remarks:

Sets the name of the document. When set, Document name, along with "FilenamePattern" ini parameter, is used to generate Output filename. But a2w::Document::setOutputFileName() has precedence over this method.

Example:

```
$a2wDocumentPar->setName( "AFP2webSampleDoc" );
```

6.9.7.31 **setOutputFilename**

Parameter:

Output filename of type string

Return value data type:

Void

Remarks:

Sets the name of the output file with extension for this document.

Example:

```
$a2wDocumentPar->setOutputFilename ( "AFP2webSample.pdf" );
```

6.9.7.32 setSimpleResourceGroupName

Parameter:

Simple resource group name of type string

Return value data type:

Void

Remarks:

Sets the simple name for the output AFP resource group file of this document. This method applies only for AFP output.

Example:

```
$a2wDocumentPar->setSimpleResourceGroupName( "AFP2webResGroup" );
```

6.9.8 a2w::DocumentConstants [NEW]

"a2w::DocumentConstants" module defines various constants that can be used while using "a2w::Document" module provided APIs.

6.9.8.1 Document Property Constants

Constants meant for fetching document properties using "a2w::Document::getProperty" API

Constant	Value	Type	Description
\$TITLE	Title	String	Document Title property key
\$AUTHOR	Author	String	Document Author property key
\$SUBJECT	Subject	String	Document Subject property key
\$KEYWORDS	Keywords	String	Document Keywords property key

\$CREATOR	Creator	String	Document Creator property key
\$PRODUCER	Producer	String	Document Producer property key
\$CREATIONDATE	CreationDate	String	Document Creation date property key
\$MODDATE	ModDate	String	Document Modification date property key
\$TRAPPED	Trapped	String	Document Trapped property key

6.9.9 a2w::Font

a2w::Font module provides the font handle that has been applied to text and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.9.1 getCharacterSetName

Parameter:

None

Return value data type:

String

Remarks:

Gets the name of the character set. Use only for AFP input.

Example:

```
$sCharSetNameTmp = $a2wFont->getCharacterSetName();
```

6.9.9.2 getCodedFontName

Parameter:

None

Return value data type:

String

Remarks:

Gets the coded font name. Use only for AFP input.

Example:

```
$sCodedFontNameTmp = $a2wFont->getCodedFontName();
```

6.9.9.3 getCodepageName

Parameter:

None

Return value data type:

String

Remarks:

Gets the name of the mapped code page. Use only for AFP input.

Example:

```
$sCodePageNameTmp = $a2wFont->getCodePageName();
```

6.9.9.4 getEncoding

Parameter:

None

Return value data type:

String

Remarks:

Returns the font encoding.

Example:

```
$sFontEncodingTmp = $a2wFont->getEncoding();
```

6.9.9.5 **getHeight**

Parameter:

None

Return value data type:

Float

Remarks:

Returns the font's height.

Example:

```
$fFontHeightTmp = $a2wFont->getHeight();
```

6.9.9.6 **getName**

Parameter:

None

Return value data type:

String

Remarks:

Returns the font name.

Example:

```
$sNameTmp = $a2wFont->getName();
```

6.9.9.7 **getTypefaceName**

Parameter:

None

Return value data type:

String

Remarks:

Returns the AFP font type face name.

Example:

```
$sTypefaceNameTmp = $a2wFont->getTypefaceName();
```

6.9.9.8 getWidth**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the font's width.

Example:

```
$iFontWidthTmp = $a2wFont->getWidth();
```

6.9.9.9 isBold**Parameter:**

None

Return value data type:

Boolean

Remarks:

Returns whether the font's weight is bold or not.

Example:

```
$bWeightTmp = $a2wFont->isBold();
```

6.9.9.10 isItalic

Parameter:

None

Return value data type:

Boolean

Remarks:

Returns whether the font's slant is italic or not.

Example:

```
$bSlantTmp = $a2wFont->isItalic();
```

6.9.9.11 isLight

Parameter:

None

Return value data type:

Boolean

Remarks:

Returns whether the font's weight is light or not.

Example:

```
$bWeightTmp = $a2wFont->isLight();
```

6.9.9.12 isStrikeover

Parameter:

None

Return value data type:

Boolean

Remarks:

Returns whether font is strike over or not.

Example:

```
$bStrikeOverTmp = $a2wFont->isStrikeover();
```

6.9.9.13 isUnderline

Parameter:

None

Return value data type:

Boolean

Remarks:

Returns whether font is underlined or not.

Example:

```
$bUnderlineTmp = $a2wFont->isUnderline();
```

6.9.9.14 new

Parameter:

Font type (Constants defined in "**a2w::FontConstants**" module)

Depending the the first parameter, the following is delivered as additional parameter(s):

Either:

- PC (TYPE1, TRUETYPE, OPENTYPE) Font Typeface name of type string
- AFP Coded Font name of type string
- AFP Character set name of type string,
AFP code page name of type string (Optional)

NOTE: For font type 1 (AFP), parameter 2 will be considered as AFP Coded Font name when parameter 3 is missing else parameter 2 will be considered as AFP Character Set name

Return value data type:

Font instance of type a2w::Font

Remarks:

Allocates and returns a new font instance.

NOTE: Creating AFP fonts allowed only for AFP input

Example:

```
# Allocate a PC font
$a2wFont = new a2w::Font();
$a2wFont->setName( "Helvetica" );

# Allocate a PC font
$a2wFont = new a2w::Font( $a2w::FontConstants::TYPE_TYPE1, "Helvetica" );

# Allocates an AFP coded font
$a2wFont = new a2w::Font( $a2w::FontConstants::TYPE_AFP, "X0N220B1" );

# Allocates an AFP character set and code page
$a2wFont = new a2w::Font( $a2w::FontConstants::TYPE_AFP, "c1h20000", "t1d0base" );
```

6.9.9.15 setBold**Parameter:**

Font weight of type boolean

Return value data type:

Void

Remarks:

Sets the font's weight to bold.

Example:

```
$a2wFont->setBold( 1 );
```

6.9.9.16 setCharacterSetName**Parameter:**

Character set name of type string

Return value data type:

Void

Remarks:

Sets the name for the character set of this font.

Use only for AFP input.

Example:

```
$a2wFont->setCharacterSetName( "C0H20080" );
```

6.9.9.17 setCodedFontName**Parameter:**

Coded font name of type string

Return value data type:

Void

Remarks:

Sets the name for the coded font.

Use only for AFP input.

Example:

```
$a2wFont->setCodedFontName( "X0H20080" );
```

6.9.9.18 setCodePageName**Parameter:**

Code page name of type string

Return value data type:

Void

Remarks:

Sets the name for the code page of this font.

Use only for AFP input.

Example:

```
$a2wFont->setCodePageName( "T1V10500" );
```

6.9.9.19 **setEncoding**

Parameter:

Encoding of type string. Possible values are either ANSI, SYMBOL or NULL.

Return value data type:

Void

Remarks:

Sets the encoding of the font.

Example:

```
$a2wFont->setEncoding( "ANSI" );
```

6.9.9.20 **setFamilyName**

Parameter:

Family name of type string

Return value data type:

Void

Remarks:

Sets the family name of the font.

Example:

```
$a2wFont->setFamilyName( "Latin1" );
```

6.9.9.21 **setHeight**

Parameter:

Font height of type float

Return value data type:

Void

Remarks:

Sets the height of the font. The value must be in points.

NOTE: On floating height value, only one digit is considered in the precision part. For example, the value "10.768" will be treated as "10.7".

Example:

```
$a2wFont->setHeight( 10.0 );
```

6.9.9.22 **setItalic**

Parameter:

Font slant of type boolean

Return value data type:

Void

Remarks:

Sets the font's slant to italic.

Example:

```
$a2wFont->setItalic( 1 );
```

6.9.9.23 **setLight**

Parameter:

Font weight of type boolean

Return value data type:

Void

Remarks:

Sets the font's weight to light.

Example:

```
$a2wFont->setLight( 1 );
```

6.9.9.24 setName

Parameter:

Font name of type string

Return value data type:

Void

Remarks for PDF Output:

Sets the name of a font.

Font name should be one of the standard Type1 fonts as given below.

- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
Helvetica-BoldOblique
- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Symbol
- ZapfDingbats

The default font name is "Helvetica". All names are case sensitive, using same names with different case leads AFP2web to search for external font files.

Note: Adobe has its own definition for above set of fonts, so they can be used with PDF.

Example for PDF Output:

```
$a2wFont->setName( "Time-Roman" );
```

Remarks For TIFF Output:

Font name is the typeface name of font. To find out the typeface name of a font follow given steps below appropriate to OS. Default font name is "Helvetica" for both Unix & Windows.

Example for TIFF Output:

```
$a2wFont->setName( "Verdana" );
```

How to Determine Font Names on Windows:

1. Open "Control Panel->Fonts"
2. Double click on the required font
3. The font window will be displayed showing font information
4. For TrueType fonts, use the name displayed for "Typeface name" as parameter to the Scripting Facility method "Font::setName()"
5. For Type1 fonts, use the name given on top of the font window as parameter to Scripting Facility method "Font::setName()".

How to Determine Font Names on a Unix Operating System:

Type "xlsfonts" at the command prompt

```
prompt$>xlsfonts
```

If issuing this command on the terminal window, then make sure that the "DISPLAY" environment variable is set properly with display ID.

The "xlsfonts" command will display the names of all available in XLFD (X Logical Font Descriptor) format (as given below):

```
prompt$>xlsfonts
-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso10646-1
-adobe-courier-medium-o-normal--10-100-75-75-m-60-iso10646-1
-adobe-helvetica-bold-o-normal--10-100-75-75-p-60-iso10646-1
-adobe-helvetica-medium-o-normal--10-100-75-75-p-57-
iso10646
```

In XLFD format, font attributes are separated by '-' character.

Use the second attribute that passes the typeface name of the font as parameter to setName().

Example of a Font Name on Unix:

```
-adobe-courier-bold-o-normal-10-100-75-75-m-60-iso10646-1
```

Note: The text "courier" is the font typeface name.

```
$a2wFont->setName( "courier" );
```

6.9.9.25 setStrikeover

Parameter:

Font strikeover of type boolean

Return value data type:

Void

Remarks

Sets the font to strike over.

Example

```
$a2wFont->setStrikeover( 1 );
```

6.9.9.26 **setUnderline**

Parameter:

Font underline of type boolean

Return value data type:

Void

Remarks:

Sets the font to underlined.

Example:

```
$a2wFont->setUnderline( 1 );
```

6.9.9.27 **setWidth**

Parameter:

Font width of type integer

Return value data type:

Void

Remarks:

Sets the width of the font.

Example:

```
$a2wFont->setWidth( 40 );
```

6.9.10 **a2w::FontConstants [NEW]**

"a2w::FontConstants" module defines various constants that can be used while creating "a2w::Font" module provided "new" API.

6.9.10.1 Font Constants

Constants meant for creating font using "a2w::Font::new" API

Constant	Value	Type	Description
\$TYPE_TYPE1	0	Integer	PC Type1 font
\$TYPE_TRUETYPE	0	Integer	PC True Type font
\$TYPE_OPENTYPE	0	Integer	PC Open Type font
\$TYPE_AFP	1	Integer	AFP Font (Coded Font or Char set & Code Page)

6.9.11 a2w::Image [NEW]

a2w::Image module provides image object handle of page content and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.11.1 getAngle

Parameter:

None

Return value data type:

Integer

Remarks:

Get angle (rotation) in clockwise (0,90,180,270)

Example:

```
$iAngleTmp = $a2wImage->getAngle();
```

6.9.11.2 getBitsPerPixel

Parameter:

None

Return value data type:

Integer

Remarks:

Get image bits per pixel

Example:

```
$iBitsPerPixelTmp = $a2wImage->getBitsPerPixel();
```

6.9.11.3 getCompression**Parameter:**

None

Return value data type:

Integer

Remarks:

Get image compression identifier

Compression Identifier	Compression Name
0	Uncompressed Image Data
1	CCITT G3
2	CCITT G3 2D
3	CCITT G4
4	AFP IOCA IBM MMR
5	AFP IOCA RL4
6	AFP IOCA ABIC
7	AFP IOCA ABIC CONCAT
8	AFP IOCA OS2 COLOR
9	JPEG
10	OLD JPEG
11	LZW
12	PACK BITS
13	PNG

14	RLE8
15	GIF
16	PCX
17	BMP
18	RLE4
19	UNSUPPORTED
20	CCITT RLE
21	CCITT RLEW
22	IT8 CT PAD
23	IT8 LW
24	IT8 MP
25	IT8 BL
26	PIXAR FILM
27	PIXAR LOG
28	DEFLATE
29	ADOBE DEFLATE
30	THUNDERSCAN
31	NEXT
32	KODAK DCS
33	JBIG
34	SGILOG
35	SGILOG 24
36	JPEG 2000

Example:

```
$iCompressionTmp = $a2wImage->getCompression();
```

6.9.11.4 **getData**

Parameter:

None

Return value data type:

Buffer of type byte

Remarks:

Gets the image's data buffer.

Example:

```
$byImageDataTmp = $a2wImage->getData();
```

6.9.11.5 **getDataLength**

Parameter:

None

Return value data type:

Integer

Remarks:

Gets the image's data buffer length.

Example:

```
$iImageDataLenTmp = $a2wImage->getDataLength();
```

6.9.11.6 **getHeight**

Parameter:

None

Return value data type:

Integer

Remarks:

Get image height in pixels

Example:

```
$iHeightTmp = $a2wImage->getHeight();
```

6.9.11.7 **getName**

Parameter:

None

Return value data type:

String

Remarks:

Returns image name

Example:

```
$sNameTmp = $a2wImage->getName();
```

6.9.11.8 **getPresentationHeight**

Parameter:

None

Return value data type:

Integer

Remarks:

Get image presentation height

Example:

```
$iPresentationHeightTmp = $a2wImage->getPresentationHeight();
```

6.9.11.9 **getPresentationWidth**

Parameter:

None

Return value data type:

Integer

Remarks:

Get image presentation width

Example:

```
$iPresentationWidthTmp = $a2wImage->getPresentationWidth();
```

6.9.11.10 `getSamplesPerPixel`

Parameter:

None

Return value data type:

Integer

Remarks:

Get image samples per pixel

Example:

```
$iSamplesPerPixelTmp = $a2wImage->getSamplesPerPixel();
```

6.9.11.11 `getUniqueColorsCount`

Parameter:

iMaxColorsPar, maximum colors to be analyzed. Beyond this limit makes no sense to count. Default is -1, means count all unique colors.

Return value data type:

Integer

- In case of error, < 0
- In case of success, Maximum colors counted considering the parameter. The value could be
when iMaxColorsPar = -1, Full count of unique colors
when iMaxColorsPar >= 0, Actual count of unique colors if image has <= Max colors or Max colors[+ some count], if image has > Max colors

Remarks:

Get unique colors count of image object

Example:

```
$iUniqueColorCountTmp = $a2wImage->getUniqueColorsCount();
```

6.9.11.12 `getWidth`

Parameter:

None

Return value data type:

Integer

Remarks:

Get image width in pixels

Example:

```
$iWidthTmp = $a2wImage->getWidth();
```

6.9.11.13 getXPos**Parameter:**

None

Return value data type:

Integer

Remarks:

Get X position of image object

Example:

```
$iXPosTmp = $a2wImage->getXPos();
```

6.9.11.14 getYPos**Parameter:**

None

Return value data type:

Integer

Remarks:

Get Y position of image object

Example:

```
$iYPosTmp = $a2wImage->getYPos();
```

6.9.11.15 remove**Parameter:**

None

Return value data type:

Void

Remarks:

Removes image from presentation, but positioning image object on output coordinate system is done still

Example:

```
$a2wImage->remove();
```

6.9.12 a2w::ImageConstants [NEW]

"a2w::ImageConstants" module defines integer constants that are returned by "getCompression()" method of "a2w::Image" module.

6.9.12.1 Constants

Constant	Value	Description
\$COMPRESSION_UNCOMPRESSED	0	Uncompressed Image Data
\$COMPRESSION_CCITT_G3	1	CCITT G3
\$COMPRESSION_CCITT_G3_2D	2	CCITT G3 2D
\$COMPRESSION_CCITT_G4	3	CCITT G4
\$COMPRESSION_AFP_IOCA_IBM_MMR	4	AFP IOCA IBM MMR
\$COMPRESSION_AFP_IOCA_RL4	5	AFP IOCA RL4
\$COMPRESSION_AFP_IOCA_ABIC	6	AFP IOCA ABIC
\$COMPRESSION_AFP_IOCA_ABIC_CONCAT	7	AFP IOCA ABIC CONCAT
\$COMPRESSION_AFP_IOCA_OS2_COLOR	8	AFP IOCA OS2 COLOR
\$COMPRESSION_JPEG	9	JPEG
\$COMPRESSION_JPEG_OLD	10	OLD JPEG
\$COMPRESSION_LZW	11	LZW

\$COMPRESSION_PACK_BITS	12	PACK BITS
\$COMPRESSION_PNG	13	PNG
\$COMPRESSION_RLE8	14	RLE8
\$COMPRESSION_GIF	15	GIF
\$COMPRESSION_PCX	16	PCX
\$COMPRESSION_BMP	17	BMP
\$COMPRESSION_RLE4	18	RLE4
\$COMPRESSION_UNSUPPORTED	19	UNSUPPORTED
\$COMPRESSION_CCITT_RLE	20	CCITT RLE
\$COMPRESSION_CCITT_RLEW	21	CCITT RLEW
\$COMPRESSION_IT8_CT_PAD	22	IT8 CT PAD
\$COMPRESSION_IT8_LW	23	IT8 LW
\$COMPRESSION_IT8_MP	24	IT8 MP
\$COMPRESSION_IT8_BL	25	IT8 BL
\$COMPRESSION_PIXAR_FILM	26	PIXAR FILM
\$COMPRESSION_PIXAR_LOG	27	PIXAR LOG
\$COMPRESSION_DEFLATE	28	DEFLATE
\$COMPRESSION_ADOBE_DEFLATE	29	ADOBE DEFLATE
\$COMPRESSION_THUNDERSCAN	30	THUNDERSCAN
\$COMPRESSION_NEXT	31	NEXT
\$COMPRESSION_KODAK_DCS	32	KODAK DCS
\$COMPRESSION_JBIG	33	JBIG
\$COMPRESSION_SGILOG	34	SGILOG
\$COMPRESSION_SGILOG_24	35	SGILOG 24
\$COMPRESSION_JPEG_2000	36	JPEG 2000

6.9.13 a2w::Index

a2w::Index module provides document/page index handle and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.13.1 getEBCDICName

Parameter:

None

Return value data type:

String

Remarks:

Gets the the EBCDIC name of the index.

Example:

```
$sEBCDICNameTmp = $a2wIndexTmp->getEBCDICName();
```

6.9.13.2 getEBCDICValue

Parameter:

None

Return value data type:

String

Remarks:

Gets the index value in EBCDIC.

Example:

```
$sEBCDICValueTmp = $a2wIndexTmp->getEBCDICValue();
```

6.9.13.3 getIndexedObjectName

Parameter:

None

Return value data type:

String

Remarks:

Gets the name of the indexed object. The indexed object can be an indexed page or page group.

Example:

```
$sIndexedObjectNameTmp = $a2wIndexTmp->getIndexedObjectName();
```

6.9.13.4 getIndexedObjectType**Parameter:**

None

Return value data type:

Integer value:

0	Page
1	Page group

Remarks:

Gets the type of the indexed object (page or page group).

Example:

```
$iIndexedObjectTypeTmp = $a2wIndexTmp->getIndexedObjectName();
```

6.9.13.5 getLevelNumber**Parameter:**

None

Return value data type:

Integer

Remarks:

Gets the level number (used to position hierarchically).

Example:

```
$iLevelNrTmp = $a2wIndexTmp->getLevelNumber();
```

6.9.13.6 **getName**

Parameter:

None

Return value data type:

String

Remarks:

Gets the index name.

Note: This method was previously named "getAttributeName".

Example:

```
$sNameTmp = $a2wIndexTmp->getName();
```

6.9.13.7 **getNameLength**

Parameter:

None

Return value data type:

Integer

Remarks:

Gets the the length of the index name. This length is used to interpret the name which is retrieved as EBCDIC string.

Example:

```
$iNameLengthTmp = $a2wIndexTmp->getNameLength();
```

6.9.13.8 **getSequenceNumber**

Parameter:

None

Return value data type:

Integer

Remarks:

Gets the sequence number (used to identify identical indexes).

Example:

```
$iSeqNrTmp = $a2wIndexTmp->getSequenceNumber();
```

6.9.13.9 getValue**Parameter:**

None

Return value data type:

String

Remarks:

Gets the index value.

Note: This method was previously named "getAttributeValue".

Example:

```
$sValueTmp = $a2wIndexTmp->getValue();
```

6.9.13.10 getValueLength**Parameter:**

None

Return value data type:

Integer

Remarks:

Get the length of the index value in EBCDIC.

Example:

```
$iValueLengthTmp = $a2wIndexTmp->getValueLength();
```

6.9.13.11 new**Parameter:**

None

Return value data type:

Index instance of type a2w::Index

Remarks:

Allocates and returns the index instance.

Example:

```
$a2wIndexTmp = new a2w::Index();
```

6.9.13.12 remove**Parameter:**

None

Return value data type:

None

Remarks:

Removes the index from the presentation data.

Example:

```
$a2wIndexTmp->remove();
```

6.9.13.13 setLevelNumber**Parameter:**

Level number of type integer

Return value data type:

Void

Remarks:

Sets the level number (used to position hierarchically).

Example:

```
$a2wIndexTmp->setLevelNumber( 1 );
```

6.9.13.14 setName**Parameter:**

Index name of type string

Return value data type:

Void

Remarks:

Sets the index name.

Note: This method was previously called as "setAttributeName".

Example:

```
$a2wIndexTmp->setName( "Producer" );
```

6.9.13.15 setSequenceNumber**Parameter:**

Sequence number of type integer

Return value data type:

Void

Remarks:

Sets the sequence number of the index (used to identify identical indexes).

Example:

```
$a2wIndexTmp->setSequenceNumber( 10 );
```

6.9.13.16 setValue**Parameter:**

Index value of type string.

Return value data type:

Void

Remarks:

Sets the index value.

Note: This method was previously called as "setAttributeValue".

Example:

```
$a2wIndexTmp->setValue( "Maas Holding GmbH" );
```

6.9.14 a2w::Kernel

a2w::Kernel module provides the Kernel handle and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.14.1 **getExitStatus**

Parameter:

None

Return value data type:

Integer

Remarks:

Returns the AFP2web exit status.

Example:

```
$iExitStatusTmp = $a2wKernelPar->getExitStatus();
```

6.9.14.2 **getIndexFilename**

Parameter:

None

Return value data type:

String

Remarks:

Returns name of the index file used for the current spool. Valid only when AFP input with "-xf:<DocIndexFile>" command line option is specified.

Example:

```
$sIndexFilenameTmp = $a2wKernelPar->getIndexFilename();
```

6.9.14.3 **getLastErrorMessage [NEW]**

Parameter:

None

Return value data type:

String

Remarks:

Returns AFP2web last error message.

Example:

```
$sA2WLastErrTmp = $a2wKernelPar->getLastErrorMessage();
```

6.9.14.4 **getResourceFilename**

Parameter:

None

Return value data type:

String

Remarks:

Returns the name of the resource file used for the current spool.

Valid only when AFP input with "-rf:<ResourceFilename>" command line option is specified.

Example:

```
$sResFilenameTmp = $a2wKernelPar->getResourceFilename();
```

6.9.14.5 **getSpoolFilename**

Parameter:

None

Return value data type:

String

Remarks:

Returns the name of the spool file, which is currently being processed.

Example:

```
$sSpoolFilenameTmp = $a2wKernelPar->getSpoolFilename();
```

6.9.14.6 **getVersionAll**

Parameter:

None

Return value data type:

String

Remarks:

Returns the version of A2W and it's sub components.

Example:

```
$sVersionAllTmp = $a2wKernelPar->getVersionAll();
```

6.9.15 a2w::Line

"a2w::Line" module provides line object handle and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.15.1 getColor**Parameter:**

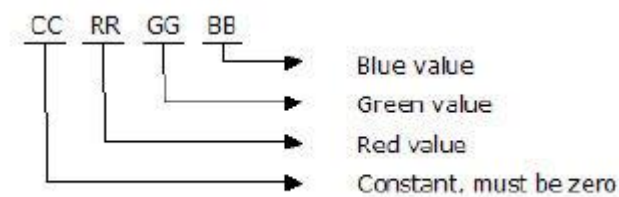
None

Return value data type:

Integer

Remarks:

Returns the color of the line. Value should be interpreted in RGB color space as given below

**Example:**

```
$iColorTmp = $a2wLine->getColor();
```

6.9.15.2 getLength**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the line length.

Example:

```
$iLengthTmp = $a2wLine->getLength();
```

6.9.15.3 getWidth**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the line width.

Example:

```
$iWidthTmp = $a2wLine->getWidth();
```

6.9.15.4 getXPos**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the X position of the line.

Example:

```
$iXPosTmp = $a2wLine->getXPos();
```

6.9.15.5 getYPos**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the Y position of the line.

Example:

```
$iYPosTmp = $a2wLine->getYPos();
```

6.9.15.6 isHorizontal**Parameter:**

None

Return value data type:

Boolean

Remarks:

Returns whether the line is horizontal or not.

Example:

```
$bHorizontalTmp = $a2wLine->isHorizontal();
```

6.9.15.7 isNegative**Parameter:**

None

Return value data type:

Boolean

Remarks:

Returns the line is set to negative or not.

Example:

```
$bNegativeTmp = $a2wLine->isNegative();
```

6.9.15.8 isVertical**Parameter:**

None

Return value data type:

Boolean

Remarks:

Returns whether the line is vertical or not.

Example:

```
$bVerticalTmp = $a2wLine->isVertical();
```

6.9.15.9 new**Parameter:**

None

Return value data type:

Instance of a line object of type a2w::Line

Remarks:

Returns a new allocated line instance.

Example:

```
$a2wLine = new a2w::Line();
```

6.9.15.10 remove**Parameter:**

None

Return value data type:

Void

Remarks:

Removes the line from its container, but positioning is still done in the coordinate system.

Example:

```
$a2wLine->remove();
```

6.9.15.11 setColor**Parameter:**

None

Return value data type:

Void

Remarks:

Sets the color of the line. Value is interpreted in RGB color space as given below

Figure 17 sf_api_ccrrgbb_image



Example:

```
$a2wLine->setColor( 0x000000FF );
```

6.9.15.12 setHorizontal

Parameter:

Horizontal flag of type boolean

Return value data type:

Void

Remarks:

Sets the horizontal attribute of the line.

Example:

```
$a2wLine->setHorizontal( 1 );
```

6.9.15.13 setLength

Parameter:

Line length of type integer

Return value data type:

Void

Remarks:

Sets the line length.

Example:

```
$a2wLine->setLength( 150 );
```

6.9.15.14 setNegative

Parameter:

Negative flag of type boolean

Return value data type:

Void

Remarks:

Sets the negative attribute of the line.

Example:

```
$a2wLine->setNegative( 1 );
```

6.9.15.15 setVertical**Parameter:**

Vertical flag of type boolean

Return value data type:

Void

Remarks:

Sets the vertical attribute of the line.

Example:

```
$a2wLine->setVertical( 1 );
```

6.9.15.16 setWidth**Parameter:**

Line width of type integer

Return value data type:

Void

Remarks:

Sets the line's width.

Example:

```
$a2wLine->setWidth( 2 );
```

6.9.15.17 setXPos**Parameter:**

X position of type integer

Return value data type:

Void

Remarks:

Set the X position of the line.

Example:

```
$a2wLine->setXPos( 100 );
```

6.9.15.18 setYPos**Parameter:**

Y position of type integer

Return value data type:

Void

Remarks:

Sets the Y position of the line.

Example:

```
$a2wLine->setYPos( 200 );
```

6.9.16 a2w::MediumMap

a2w::MediumMap module provides page applied Medium map handle and exposes the following interfaces to the AFP2web Scripting Facility. Valid only for AFP and MMD input.

6.9.16.1 getDuplexControl**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the duplex control setting, like simple, double, or triple.

1	Simple
2	Double
3	Triple

Example:

```
$iDupCtrlTmp = $a2wMediumMap->getDuplexControl();
```

6.9.16.2 getFormdefName**Parameter:**

None

Return value data type:

String

Remarks:

Returns the name of the Formdef.

Example:

```
$sFormdefNameTmp = $a2wMediumMap->getFormdefName();
```

6.9.16.3 getHeight**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the height of the medium.

Example:

```
$iHeightTmp = $a2wMediumMap->getHeight();
```

6.9.16.4 getName**Parameter:**

None

Return value data type:

String

Remarks:

Returns the name of medium map.

Example:

```
$sNameTmp = $a2wMediumMap->getName();
```

6.9.16.5 getNupControl**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the N-up control setting and the return value must be in between 1 to 4.

Example:

```
$iNupCtrlTmp = $a2wMediumMap->getNupControl();
```

6.9.16.6 getResolution**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the resolution of medium

Example:

```
$iResolutionTmp = $a2wMediumMap->getResolution();
```

6.9.16.7 getWidth**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the width of the medium.

Example:

```
$iWidthTmp = $a2wMediumMap->getWidth();
```

6.9.17 a2w::NOP

a2w::NOP module provides AFP No Operation SFI data access and exposes the following interfaces to the AFP2web Scripting Facility. Valid only for AFP input.

6.9.17.1 getEBCDICValue**Parameter:**

None

Return value data type:

String

Remarks:

Gets the NOP's EBCDIC value

Example:

```
$sEBCDICValueTmp = $a2wNOP->getEBCDICValue();
```

6.9.17.2 getLength**Parameter:**

None

Return value data type:

Integer

Remarks:

Gets the NOP's length

Example:

```
$iLengthTmp = $a2wNOP->getLength();
```

6.9.17.3 getValue**Parameter:**

None

Return value data type:

String

Remarks:

Gets the NOP's value.

Example:

```
$sValueTmp = $a2wNOP->getValue();
```

6.9.17.4 new**Parameter:**

None

Return value data type:

NOP instance of type a2w::NOP

Remarks:

Returns newly allocated instance of a2w::NOP

Example:

```
$a2wNOP = new a2w::NOP();
```

6.9.17.5 remove**Parameter:**

None

Return value data type:

Void

Remarks:

Removes NOP from it's container (page/document)

Example:

```
$a2wNOP->remove();
```

6.9.17.6 setValue**Parameter:**

NOP value of type string

Return value data type:

Void

Remarks:

Set NOP value

Example:

```
$a2wNOP->setValue( "Created by AFP2web" );
```

6.9.18 a2w::Overlay

a2w::Overlay module provides page applied form/overlay handle and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.18.1 getFirstLine**Parameter:**

None

Return value data type:

Line instance of type a2w::Line

Remarks:

Returns the first line object of overlay.

Example:

```
$a2wLineTmp = $a2wOverlay->getFirstLine();  
while ( $a2wLineTmp != 0 ){  
    #---- Access line info  
    ...  
    #---- Get next line  
    $a2wLineTmp = $a2wOverlay->getNextLine();  
}
```

6.9.18.2 getFirstOverlay**Parameter:**

None

Return value data type:

Overlay instance of type a2w::Overlay

Remarks:

Returns the first included overlay resource.

Example:

```
$a2wOlyTmp = $a2wOverlay->getFirstOverlay();
while $a2wOlyTmp != 0 ){
    #---- Access overlay info
    ...
    #---- Get next overlay
    $a2wOlyTmp = $a2wOverlay->getNextOverlay();
}
```

6.9.18.3 getFirstPageSegment**Parameter:**

None

Return value data type:

Page segment instance of type a2w::PSEG

Remarks:

Returns the first included page segment resource.

Example:

```
$a2wPSEGTmp = $a2wOverlay->getFirstPageSegment();
while $a2wPSEGTmp != 0 ){
    #---- Access page segment info
    ...
    #---- Get next page segment
    $a2wPSEGTmp = $a2wOverlay->getNextPageSegment();
}
```

6.9.18.4 getFirstText**Parameter:**

None

Return value data type:

Text instance of type a2w::Text

Remarks:

Get the first text object of overlay.

Note: This method was previously called "getFirstTextObject".

Example:

```
$a2wTextTmp = $a2wOverlay->getFirstText();  
while ($a2wTextTmp != 0){  
    #---- Access text info  
    ...  
    #---- Get next text  
    $a2wTextTmp = $a2wOverlay->getNextText();  
}
```

6.9.18.5 getHeight**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the height of the overlay.

Example:

```
$iHeightTmp = $a2wOverlay->getHeight();
```

6.9.18.6 getIncludedXPosition**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the X position of this overlay included in the page.

Example:

```
$iInclXPosTmp = $a2wOverlay->getIncludedXPosition();
```

6.9.18.7 getIncludedYPosition**Parameter:**

None

Return value data type:

Integer

Remarks:

Returns the Y position of this overlay included in the page.

Example:

```
$iInclYPosTmp = $a2wOverlay->getIncludedYPosition();
```

6.9.18.8 getName**Parameter:**

None

Return value data type:

String

Remarks:

Returns the name of the overlay.

Example:

```
$sNameTmp = $a2wOverlay->getName();
```

6.9.18.9 getNextLine**Parameter:**

None

Return value data type:

Line instance of type a2w::Line

Remarks:

Returns the next line object of the overlay. This method is normally used in an iteration loop after using `getFirstLine`.

Example:

```
$a2wLineTmp = $a2wOverlay->getFirstLine();
while $a2wLineTmp != 0 ){
    #---- Access line info
    ...
    #---- Get next line
    $a2wLineTmp = $a2wOverlay->getNextLine();
}
```

6.9.18.10 getNextOverlay**Parameter:**

None

Return value data type:

Overlay instance of type a2w::Overlay

Remarks:

Returns the next included overlay resource (used to traverse the list of included overlays). This method is normally used in an iteration loop after using `getFirstOverlay`.

Example:

```
$a2wOlyTmp = $a2wOverlay->getFirstOverlay();
while $a2wOlyTmp != 0 ){
    #---- Access overlay info
    ...
    #---- Get next overlay
    $a2wOlyTmp = $a2wOverlay->getNextOverlay();
}
```

6.9.18.11 getNextPageSegment**Parameter:**

None

Return value data type:

Page segment instance of type a2w::PSEG

Remarks:

Returns the next included page segment resource (used to traverse the list of included overlays). This method is normally used in an iteration loop after using `getFirstPageSegment`.

Example:

```
$a2wPSEGTmp = $a2wOverlay->getFirstPageSegment();
while $a2wPSEGTmp != 0 ){
    #---- Access page segment info
    ...
    #---- Get next page segment
    $a2wPSEGTmp = $a2wOverlay->getNextPageSegment();
}
```

6.9.18.12 getNextText

Parameter:

None

Return value data type:

Text instance of type a2w::Text

Remarks:

Get the next text object of overlay (used to iterate the list of overlay text objects after calling "getFirstText").

Note: This method was previously called "getNextTextObject".

Example:

```
$a2wTextTmp = $a2wOverlay->getFirstText();
while ($a2wTextTmp != 0 ){
    #---- Access text info
    ...
    #---- Get next text
    $a2wTextTmp = $a2wOverlay->getNextText();
}
```

6.9.18.13 **getResolution**

Parameter:

None

Return value data type:

Integer

Remarks:

Returns the resolution of the overlay.

Example:

```
$iResTmp = $a2wOverlay->getResolution();
```

6.9.18.14 **getWidth**

Parameter:

None

Return value data type:

Integer

Remarks:

Returns the width of the overlay

Example:

```
$iWidthTmp = $a2wOverlay->getWidth();
```

6.9.19 **a2w::Page**

a2w::Page module provides currently processed page handle and exposes following interfaces to the AFP2web Scripting Facility.

6.9.19.1 addAnnotation

Parameter:

X position of type integer

Y position of type integer

Width of type integer

Height of type integer

URL text of type string

Border width of type integer

Color of type integer

Flags of type integer

Return value data type:

Void

Remarks:

Add annotation to page, Valid only for PDF output.

Example:

```
$a2wPagePar->addAnnotation( 0, 0, 100, 10, "Maas Holding GmbH", 2,0xFF0000, 0 );
```

6.9.19.2 addBookmark

Parameter:

Bookmark name of type string

Bookmark value of type string

Return value data type:

Void

Remarks:

Add bookmark to page

Example:

```
$a2wPagePar->addBookmark( "Software", "AFP2web" );
```

6.9.19.3 addImage

Parameter:

Image name of type string

X include position of type integer

Y include position of type integer

Presentation Width of type integer

Presentation Height of type integer

Rotation of type integer

Return value data type:

Void

Remarks:

Add raster(tif, jpg, png) image to page with rotation in clockwise (0,90,180,270).

Example:

```
$a2wPagePar->addImage( "BGFORM.jpg", 0, 100, 1000, 1500, 0 );
```

6.9.19.4 addIndex

Parameter:

Index instance of type a2w::Index

Return value data type:

Void

Remarks:

Add index to page

Example:

```
#---- Create new index
$a2wIndexTmp = new a2w::Index();
#---- Set index info
$a2wIndexTmp->setName( "Producer" );
$a2wIndexTmp->setValue( "Maas Holding GmbH" );
#---- Add index to page
$a2wPagePar->addIndex( $a2wIndexTmp );
```

6.9.19.5 addLine

Parameter:

Line instance of type a2w::Line

Return value data type:

Void

Remarks:

Add line to page

Example:

```
#---- Create new line
$a2wLineTmp = new a2w::Line();
#---- Set line info
$a2wLineTmp->setLength( 200 );
$a2wLineTmp->setXPos( 100 );
$a2wLineTmp->setYPos( 100 );
...
#---- Add line to page
$a2wPagePar->addLine( $a2wLineTmp );
```

6.9.19.6 addNOP

Parameter:

No Operation instance of type a2w::NOP

Return value data type:

Void

Remarks:

Add No Operation to page. The No Operation element is inserted after "Begin Page" SFI and valid only for AFP output.

Example:

```
##---- Create new NOP
$a2wNOPTmp = new a2w::NOP();
#---- Set NOP value
$a2wNOPTmp->setValue( "Created by AFP2web" );
#---- Add NOP to page
$a2wPagePar->addNOP( $a2wNOPTmp );
```

6.9.19.7 addOverlay

Parameter:

Overlay name of type string

X include position of overlay of type integer

Y include position of overlay of type integer

Return value data type:

Void

Remarks:

Add overlay to page

Example:

```
$a2wPagePar->addOverlay( "01MEDF01", 0, 100 );
```

6.9.19.8 addPSEG

Parameter:

Page segment name of type string

X include position of type integer

Y include position of type integer

Presentation Width of type integer

Presentation Height of type integer

Rotation of type integer

Return value data type:

Void

Remarks:

Add page segment to page with rotation in clockwise (0,90,180,270).

Example:

```
$a2wPagePar->addPSEG( "S1A2W001", 0, 100, 1000, 1500, 0 );
```

6.9.19.9 addText

Parameter:

Text instance of type a2w::Text

Return value data type:

Void

Remarks:

Add text to page

Example:

```
#---- Create new text
$a2wTextTmp = new a2w::Text();
#---- Create and set font on text
$a2wFontTmp = new a2w::Font();
$a2wFontTmp->setName( "Helvetica" ); # Set font name
#---- Set font on text
$a2wTextTmp->setFont( $a2wFontTmp );
#---- Set text info
$a2wTextTmp->setText( "Watermark" );
$a2wTextTmp->setXPos( 100 );
$a2wTextTmp->setYPos( 100 );
...
#---- Add text to page
$a2wPagePar->addText( $a2wTextTmp );
```

6.9.19.10 getAppliedMediumMap

Parameter:

None

Return value data type:

a2w::MediumMap

Remarks:

Returns medium map applied on given page

Example:

```
$a2wMediumMapTmp = $a2wPagePar->getAppliedMediumMap();
```

6.9.19.11 getFirstBookmark

Parameter:

None

Return value data type:

Bookmark instance of type a2w::Bookmark

Remarks:

Gets the first bookmark at the page level.

Example:

```
$a2wBookmarkTmp = $a2wPagePar->getFirstBookmark();
while ( $a2wBookmarkTmp != 0 ){
    #---- Access bookmark info
    ...
    #---- Get next bookmark
    $a2wBookmarkTmp = $a2wPagePar->getNextBookmark();
}
```

6.9.19.12 getFirstImage

Parameter:

None

Return value data type:

Image instance of type a2w::Image

Remarks:

Returns first page included image.

Example:

```
$a2wImageTmp = $a2wPagePar->getFirstImage();
while ( $a2wImageTmp != 0 ){
    #---- Access Image info
    ...
    #---- Get next image
    $a2wImageTmp = $a2wPagePar->getNextImage();
}
```

6.9.19.13 **getFirstIndex**

Parameter:

None

Return value data type:

Index instance of type a2w::Index

Remarks:

Get first index of page

Example:

```
$a2wIndexTmp = $a2wPagePar->getFirstIndex();
while ( $a2wIndexTmp != 0 ){
    #---- Access index info
    ...
    #---- Get next index
    $a2wIndexTmp = $a2wPagePar->getNextIndex();
}
```

6.9.19.14 **getFirstLine**

Parameter:

None

Return value data type:

Line instance of type a2w::Line

Remarks:

Returns first line object of page.

Example:

```
$a2wLineTmp = $a2wPagePar->getFirstLine();
while ( $a2wLineTmp != 0 ){
    #---- Access line info
    ...
    #---- Get next line
    $a2wLineTmp = $a2wPagePar->getNextLine();
}
```

6.9.19.15 **getFirstMediumOverlay**

Parameter:

None

Return value data type:

Overlay instance of type a2w::Overlay

Remarks:

Returns first medium applied overlay.

Example:

```
$a2wOlyTmp = $a2wPagePar->getFirstMediumOverlay();
while ( $a2wOlyTmp != 0 ){
    #---- Access overlay info
    ...
    #---- Get next overlay
    $a2wOlyTmp = $a2wPagePar->getNextMediumOverlay();
}
```

6.9.19.16 **getFirstNOP**

Parameter:

None

Return value data type:

No Operation instance of type a2w::NOP

Remarks:

Get first No Operation of page

Example:

```
$a2wNOPTmp = $a2wPagePar->getFirstNOP();
while ( $a2wNOPTmp != 0 ){
    #---- Access NOP info
    ...
    #---- Get next NOP
    $a2wNOPTmp = $a2wPagePar->getNextNOP();
}
```

6.9.19.17 **getFirstOverlay**

Parameter:

None

Return value data type:

Overlay instance of type a2w::Overlay

Remarks:

Get first page included overlay

Example:

```
$a2wOlyTmp = $a2wPagePar->getFirstOverlay();
while ( $a2wOlyTmp != 0 ){
    #---- Access overlay info
    ...
    #---- Get next overlay
    $a2wOlyTmp = $a2wPagePar->getNextOverlay();
}
```

6.9.19.18 **getFirstPageSegment**

Parameter:

None

Return value data type:

Page segment instance of type a2w::PSEG

Remarks:

Returns first page included page segment.

Example:

```
$a2wPSEGTmp = $a2wPagePar->getFirstPageSegment();
while ( $a2wPSEGTmp != 0 ){
    #---- Access page segment info
    ...
    #---- Get next page segment
    $a2wPSEGTmp = $a2wPagePar->getNextPageSegment();
}
```

6.9.19.19 **getFirstText**

Parameter:

None

Return value data type:

Text instance of type a2w::Text

Remarks:

Get first text object of page, previously called as "getFirstTextObject"

Example:

```
$a2wTextTmp = $a2wPagePar->getFirstText();
while ( $a2wTextTmp != 0 ){
    #---- Access Text info
    ...
    #---- Get next text
    $a2wTextTmp = $a2wPagePar->getNextText();
}
```

6.9.19.20 **getFirstVector**

Parameter:

None

Return value data type:

Vector instance of type a2w::Vector

Remarks:

Returns first page included vector.

Example:

```
$a2wVectorTmp = $a2wPagePar->getFirstVector();
while ( $a2wVectorTmp != 0 ){
    #---- Access Vector info
    ...
    #---- Get next vector
    $a2wVectorTmp = $a2wPagePar->getNextVector();
}
```

6.9.19.21 **getHeight**

Parameter:

None

Return value data type:

Integer

Remarks:

Get page height

Example:

```
$iHeightTmp = $a2wPagePar->getHeight();
```

6.9.19.22 **getId**

Parameter:

None

Return value data type:

Integer

Remarks:

Get page id

Example:

```
$iIdTmp = $a2wPagePar->getId();
```

6.9.19.23 **getName**

Parameter:

None

Return value data type:

String

Remarks:

Returns page name

Example:

```
$sNameTmp = $a2wPagePar->getName();
```

6.9.19.24 getNextBookmark**Parameter:**

None

Return value data type:

Bookmark instance of type a2w::Bookmark

Remarks:

Gets the next bookmark at the page level. This method is normally used in an iteration loop after using `getFirstBookmark`.

Example:

```
$a2wBookmarkTmp = $a2wPagePar->getFirstBookmark();  
while ($a2wBookmarkTmp != 0 ){  
    #---- Access bookmark info  
    ...  
    #---- Get next bookmark  
    $a2wBookmarkTmp = $a2wPagePar->getNextBookmark();  
}
```

6.9.19.25 getNextImage**Parameter:**

None

Return value data type:

Image instance of type a2w::Image

Remarks:

Get next image object of page (used to iterate list of page image object after calling "getFirstImage").

Example:

```
$a2wImageTmp = $a2wPage->getFirstImage();
while ( $a2wImageTmp != 0 ){
    #---- Access Image info
    ...
    #---- Get next image
    $a2wImageTmp = $a2wPagePar->getNextImage();
}
```

6.9.19.26 getNextIndex**Parameter:**

None

Return value data type:

Index instance of type a2w::Index

Remarks:

Get next index of page (used to iterate list of page indexes after calling "getFirstIndex")

Example:

```
$a2wIndexTmp = $a2wPagePar->getFirstIndex();
while ( $a2wIndexTmp != 0 ){
    #---- Access index info
    ...
    #---- Get next index
    $a2wIndexTmp = $a2wPagePar->getNextIndex();
}
```

6.9.19.27 getNextLine**Parameter:**

None

Return value data type:

Line instance of type a2w::Line

Remarks:

Returns next line object of page (used to traverse the list of page lines) and must be called after "getFirstLine".

Example:

```
$a2wLineTmp = $a2wPagePar->getFirstLine();
while ( $a2wLineTmp != 0 ){
    #---- Access line info
    ...
    #---- Get next line
    $a2wLineTmp = $a2wPagePar->getNextLine();
}
```

6.9.19.28 getNextMediumOverlay**Parameter:**

None

Return value data type:

Overlay instance of type a2w::Overlay

Remarks:

Returns next medium applied overlay (used to traverse the list of medium applied overlays).

Example:

```
$a2wOlyTmp = $a2wPagePar->getFirstMediumOverlay();
while ( $a2wOlyTmp != 0 ){
    #---- Access overlay info
    ...
    #---- Get next overlay
    $a2wOlyTmp = $a2wPagePar->getNextMediumOverlay();
}
```

6.9.19.29 getNextNOP**Parameter:**

None

Return value data type:

No Operation instance of type a2w::NOP

Remarks:

Get next No Operation of page (used to iterate list of page nops after calling "getFirstNOP")

Example:

```
$a2wNOPTmp = $a2wPagePar->getFirstNOP();
while ( $a2wNOPTmp != 0 ){
    #---- Access NOP info
    ...
    #---- Get next NOP
    $a2wNOPTmp = $a2wPagePar->getNextNOP();
}
```

6.9.19.30 getNextOverlay**Parameter:**

None

Return value data type:

Overlay instance of type a2w::Overlay

Remarks:

Get next page included overlay (used to iterate list of page indexes after calling "getFirstOverlay")

Example:

```
$a2wOlyTmp = $a2wPagePar->getFirstOverlay();
while ( $a2wOlyTmp != 0 ){
    #---- Access overlay info
    ...
    #---- Get next overlay
    $a2wOlyTmp = $a2wPagePar->getNextOverlay();
}
```

6.9.19.31 getNextPageSegment**Parameter:**

None

Return value data type:

Page segment instance of type a2w::PSEG

Remarks:

Returns next page included page segment (used to traverse the list of page included page segments).

Example:

```
$a2wPSEGTmp = $a2wPage->getFirstPageSegment();
while ( $a2wPSEGTmp != 0 ){
    #---- Access page segment info
    ...
    #---- Get next page segment
    $a2wPSEGTmp = $a2wPagePar->getNextPageSegment();
}
```

6.9.19.32 getNextText**Parameter:**

None

Return value data type:

Text instance of type a2w::Text

Remarks:

Get next text object of page (used to iterate list of page text object after calling "getFirstText"), previously called as "getNextTextObject"

Example:

```
$a2wTextTmp = $a2wPage->getFirstText();
while ( $a2wTextTmp != 0 ){
    #---- Access Text info
    ...
    #---- Get next text
    $a2wTextTmp = $a2wPagePar->getNextText();
}
```

6.9.19.33 getNextVector**Parameter:**

None

Return value data type:

Vector instance of type a2w::Vector

Remarks:

Get next vector object of page (used to iterate list of page vector object after calling "getFirstVector").

Example:

```
$a2wVectorTmp = $a2wPagePar->getFirstVector();
while ( $a2wVectorTmp != 0 ){
    #---- Access Vector info
    ...
    #---- Get next vector
    $a2wVectorTmp = $a2wPagePar->getNextVector();
}
```

6.9.19.34 getOutputBuffer**Parameter:**

None

Return value data type:

Buffer of type byte*

Remarks:

Gets the output buffer for the page. Valid only when "MemoryOutputStream" and "PageOutput" INI attributes are turned on.

Example:

```
$byPageBufferTmp = $a2wPagePar->getOutputBuffer();
```

6.9.19.35 getOutputBufferLength**Parameter:**

None

Return value data type:

Integer

Remarks:

Gets the length of the page output buffer. Valid only when "MemoryOutputStream" and "PageOutput" INI attributes are turned on.

Example:

```
$iPageBufferLenTmp = $a2wPagePar->getOutputBufferLength();
```

6.9.19.36 **getOutputFilename**

Parameter:

None

Return value data type:

String

Remarks:

Gets the name of the output file for this page. Valid only when "PageOutput" INI attribute is turned on.

Example:

```
$sOutputFilenameTmp = $a2wPagePar->getOutputFilename();
```

6.9.19.37 **getPageGroupName**

Parameter:

None

Return value data type:

String

Remarks:

Returns name of page group, which contain this page

Example:

```
$sPageGroupNameTmp = $a2wPagePar->getPageGroupName();
```

6.9.19.38 **getParseId**

Parameter:

None

Return value data type:

Integer

Remarks:

Get page parse id (sequence id of page as in input spool)

Example:

```
$iParseIdTmp = $a2wPagePar->getParseId();
```

6.9.19.39 getResolution**Parameter:**

None

Return value data type:

Integer

Remarks:

Get page resolution

Example:

```
$iResTmp = $a2wPagePar->getResolution();
```

6.9.19.40 getRotation**Parameter:**

None

Return value data type:

Integer

Remarks:

Gets the page clockwise rotation (0,90,180,270).

Example:

```
$iRotationTmp = $a2wPagePar->getRotation();
```

6.9.19.41 **getSimpleFilename**

Parameter:

None

Return value data type:

String

Remarks:

Gets the file name of the output file of this page. Valid only when "PageOutput" INI attribute is turned on.

Example:

```
$sSimpleFilenameTmp = $a2wPagePar->getSimpleFilename();
```

6.9.19.42 **getSize**

Parameter:

None

Return value data type:

Long

Remarks:

Gets the size of the output file of this page. Valid only when "PageOutput" INI attributes is turned on.

Example:

```
$lOutputSizeTmp = $a2wPagePar->getSize();
```

6.9.19.43 **getWidth**

Parameter:

None

Return value data type:

Integer

Remarks:

Get page width

Example:

```
$iWidthTmp = $a2wPagePar->getWidth();
```

6.9.19.44 isConstantBack**Parameter:**

None

Return value data type:

Boolean

Remarks:

Returns true if page is medium applied constant back

Example:

```
$bConstantBackTmp = $a2wPagePar->isConstantBack();
```

6.9.19.45 isOverlayFound**Parameter:**

Overlay name of type string

Return value data type:

Boolean

Remarks:

Check whether given overlay is included in page or not

Example:

```
$bOverlayFoundTmp = $a2wPagePar->isOverlayFound("01MEDF01");
```

6.9.19.46 new

Parameter:

Page type (Constants defined in "[a2w::PageConstants](#)" module)

Return value data type:

Page instance of type `a2w::Page`

Remarks:

Creates new page instance and returns the same

Example:

```
$a2wPage = new a2w::Page( $a2w::PageConstants::TYPE_AFP );
```

6.9.19.47 setBackgroundColor

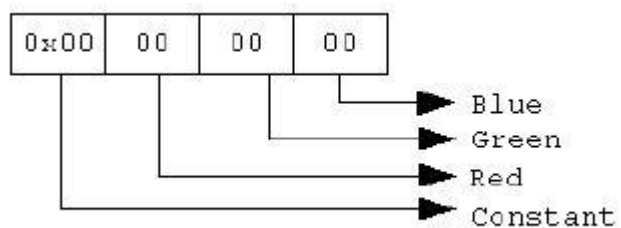
Parameter:

Color space of type byte:

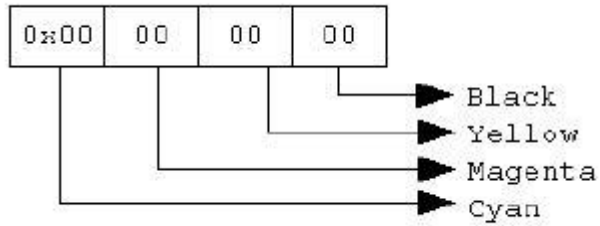
1	RGB color space
2	CMYK color space
	All other values will be ignored

Page color of type integer:

For RGB color space, value will be interpreted as given below:



For CMYK color space value will be interpreted as given below:



Return value data type:

Void

Remarks:

Set page background color.

Example:

```
# Declare color space constants
$COLORSPACE_RGB = 1;
$COLORSPACE_CMYK = 2;
# Set red color in RGB color space
$a2wPagePar->setBackgroundColor( $COLORSPACE_RGB, 0x00FF0000 );
# Set cyan color in CMYK color space
$a2wPagePar->setBackgroundColor( $COLORSPACE_CMYK, 0xFF000000 );
```

6.9.19.48 setHeight

Parameter:

Page height of type integer

Return value data type:

Void

Remarks:

Set page height, value depends upon "ScriptUnitBase" INI attribute

Example:

```
$a2wPagePar->setHeight( 584 );
```

6.9.19.49 **setOutputFileName**

Parameter:

Output filename of type string

Return value data type:

Void

Remarks:

Sets the name of the output file for this page. Valid only when "PageOutput" INI attribute is turned on.

Example:

```
$a2wPagePar->setOutputFilename( "AFP2webPageDoc.tif" );
```

6.9.19.50 **setResolution**

Parameter:

Page resolution of type integer

Return value data type:

Void

Remarks:

Set page resolution

Example:

```
$a2wPagePar->setResolution( 240 );
```

6.9.19.51 **setRotation**

Parameter:

Rotation of type integer

Return value data type:

Void

Remarks:

Sets the page rotation in clockwise (0, 90, 180, 270). By default value is 0.

Example:

```
$a2wPagePar->setRotation( 180 );
```

6.9.19.52 setWidth**Parameter:**

Page width of type integer

Return value data type:

Void

Remarks:

Set page width, value depends upon "ScriptUnitBase" INI attribute

Example:

```
$a2wPagePar->setWidth( 840 );
```

6.9.20 a2w::PageConstants [NEW]

"a2w::PageConstants" module defines various constants that can be used to create "a2w::Page" module provided "new" API.

6.9.20.1 Page Type Constants

Constants meant for creating page type using "a2w::Page::new" API

Constant	Value	Type	Description
\$TYPE_AFP	1	Integer	AFP (Advanced Function Presentation) Page
\$TYPE_LPD	2	Integer	LPD (Line Printer Data) Page
\$TYPE_TIFF	3	Integer	TIFF (Tagged Image File Format) Page
\$TYPE_PDF	4	Integer	PDF (Portable Document File) Page

\$TYPE_RASTER	7	Integer	RASTER (All images) Page
\$TYPE_MMD	12	Integer	MMD (Mixed Mode Data) Page

6.9.21 a2w::PSEG

a2w::PSEG module provides page applied page segment handle and exposes the following interfaces to the AFP2web Scripting Facility. Valid only for AFP input.

6.9.21.1 getIncludedXPosition

Parameter:

None

Return value data type:

Integer

Remarks:

Returns included X position of this page segment in page.

Example:

```
$iInclXPosTmp = $a2wPSEG->getIncludedXPosition();
```

6.9.21.2 getIncludedYPosition

Parameter:

None

Return value data type:

Integer

Remarks:

Returns included Y position of this page segment in page

Example:

```
$iInclYPosTmp = $a2wPSEG->getIncludedYPosition();
```

6.9.21.3 **getName**

Parameter:

None

Return value data type:

String

Remarks:

Returns page segment name

Example:

```
$sNameTmp = $a2wPSEG->getName();
```

6.9.22 **a2w::Text [EXTENDED]**

"a2w::Text" module provides text object handle and expose following interfaces to scripting facility

6.9.22.1 **getAngle**

Parameter:

None

Return value data type:

Integer

Remarks:

Get angle (rotation) in clockwise (0,90,180,270)

Example:

```
$iAngleTmp = $a2wText->getAngle();
```

6.9.22.2 getColor

Parameter:

None

Return value data type:

Integer

Remarks:

Get text color of 4 s length (as given in below format)



Example:

```
$iColorTmp = $a2wText->getColor();
```

6.9.22.3 getEBCDICText

Parameter:

None

Return value data type:

String

Remarks:

Get EBCDIC value of text object

Example:

```
$sEBCDICTextTmp = $a2wText->getEBCDICText();
```

6.9.22.4 **getFont**

Parameter:

None

Return value data type:

Font instance of type a2w::Font

Remarks:

Returns font used to render this text.

Example:

```
$a2wFontTmp = $a2wText->getFont();
```

6.9.22.5 **getMappedFontLocalId**

Parameter:

None

Return value data type:

Integer

Remarks:

Get text-applied font's local map id

Example:

```
$iFontLocalIdTmp = $a2wText->getMappedFontLocalId();
```

6.9.22.6 **getText**

Parameter:

None

Return value data type:

String

Remarks:

Get value of text object

Example:

```
$sTextTmp = $a2wText->getText();
```

6.9.22.7 getTextLen**Parameter:**

None

Return value data type:

Integer

Remarks:

Get value length of text object

Example:

```
$iTextLenTmp = $a2wText->getTextLen();
```

6.9.22.8 getXPos**Parameter:**

None

Return value data type:

Integer

Remarks:

Get X position of text object

Example:

```
$iXPosTmp = $a2wText->getXPos();
```

6.9.22.9 **getYPos**

Parameter:

None

Return value data type:

Integer

Remarks:

Get Y position of text object

Example:

```
$iYPosTmp = $a2wText->getYPos();
```

6.9.22.10 **new**

Parameter:

None

Return value data type:

Text instance of type a2w::Text

Remarks:

Allocate new text instance

Example:

```
$a2wText = new a2w::Text();
```

6.9.22.11 **remove**

Parameter:

None

Return value data type:

Void

Remarks:

Removes text from presentation, but positioning text object on output coordinate system is done still

Example:

```
$a2wText->remove();
```

6.9.22.12 setAngle**Parameter:**

Angle of type integer

Return value data type:

Void

Remarks:

Set angle (rotation) of text object in clockwise (0,90,180,270).

Example:

```
$a2wText->setAngle( 90 );
```

6.9.22.13 setColor**Parameter:**

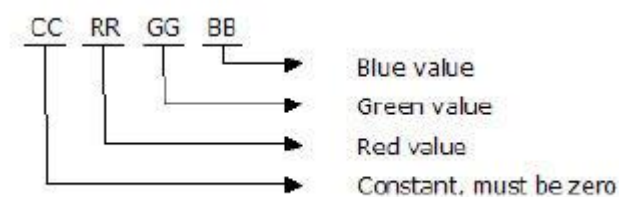
Color of type integer

Return value data type:

Void

Remarks:

Set color of text object. Value is interpreted in RGB color space as given below



Example:

```
$a2wText->setColor( 0x00FF0000 ); # Set red color
```

6.9.22.14 setEBCDICText [NEW]**Parameter:**

1. Binary string of EBCDIC characters
2. Length of binary string

Return value data type:

Void

Remarks:

Set value of text object using given EBCDIC text

Example:

```
# Pack "Maas Holding GmbH" string as EBCDIC
$sPackedEBCDICString = pack( "CCCCCCCC", 0xD4, 0x81, 0x81, 0xA2, 0x40, 0xC8, 0x96, 0x93, 0x84,
0x89, 0x95, 0x87, 0x40, 0xC7, 0x94, 0x82, 0xC8 );
$iLength = 11;

# Set packed EBCDIC string as text on object
$a2wText->setEBCDICText( $sPackedEBCDICString, $iLength );
```

6.9.22.15 setFont**Parameter:**

Font instance of type a2w::Font

Return value data type:

Void

Remarks:

Set font of text object

Example:

```
#---- Create font
$a2wFontTmp = new a2w::Font();
#---- Set font details
$a2wFontTmp->setName( "Verdana" );
```

```
...
#---- Set above created font as font in text
$a2wText->setFont( $a2wFontTmp );
```

6.9.22.16 **setText**

Parameter:

Text of type string

Return value data type:

Void

Remarks:

Set value of text object

Example:

```
$a2wText->setText( "Maas Holding GmbH" );
```

6.9.22.17 **setTextLen**

Parameter:

Text length of type integer

Return value data type:

Void

Remarks:

Set value length of text object

Example:

```
$a2wText->setTextLen( 9 );
```

6.9.22.18 **setXPos**

Parameter:

X position of type integer

Return value data type:

Void

Remarks:

Set X position of text object

Example:

```
$a2wText->setXPos( 100 );
```

6.9.22.19 setYPos**Parameter:**

Y position of type integer

Return value data type:

Void

Remarks:

Set Y position of text object

Example:

```
$a2wText->setYPos( 200 );
```

6.9.23 a2w::UserData

"a2w::UserData" module provides user defined data object handle to pass back user's own info to Kernel which can later be accessed through Java SDK callback interface and "a2w::UserData" module expose following interfaces to scripting facility.

6.9.23.1 getData**Parameter:**

None

Return value data type:

Buffer of type byte

Remarks:

Gets the data of user data.

Example:

```
#---- Fetch the user data  
$byDataTmp = $a2wUserDataTmp->getData();
```

6.9.23.2 getDataLength**Parameter:**

None

Return value data type:

Integer

Remarks:

Gets the user data length.

Example:

```
#---- Fetch the user data length  
$iDataLengthTmp = $a2wUserDataTmp->getDataLength();
```

6.9.23.3 getId**Parameter:**

None

Return value data type:

String

Remarks:

Gets the user data id.

Example:

```
#---- Fetch the user data id  
$sDataIdTmp = $a2wUserDataTmp->getId();
```

6.9.23.4 new

Parameter:

None

Return value data type:

UserData instance of type a2w::UserData

Remarks:

Allocate new user data instance

Example:

```
#--- Create a new UserData
$a2wUserDataTmp = new a2w::UserData();
```

6.9.23.5 setData

Parameter:

User data of type String

User data length of type Integer

Return value data type:

Void

Remarks:

Sets the user data.

Example:

```
#---- Create user data object
$a2wUserDataTmp = new a2w::UserData();
#---- Fetch first Page Index
$a2wPageIndexTmp = $a2wPage->getFirstIndex();
while ( $a2wPageIndexTmp != 0 ){
    #---- Add Index Record
    $sIndexDataTmp .= $a2wPageIndexTmp->getName() . "=" .
    $a2wPageIndexTmp->getValue() . "\r\n";
    #---- Fetch next Page Index
    $a2wPageIndexTmp = $a2wPage->getNextIndex();
}
#---- Set current page index as data
$a2wUserDataTmp->setData( $sIndexDataTmp, length( $sIndex-
DataTmp ) );
```

6.9.23.6 setDataLength

Parameter:

User data length of type Integer

Return value data type:

Void

Remarks:

Set the length of user data.

Example:

```
#---- Create user data object
$a2wUserDataTmp = new a2w::UserData();
#---- Set user data length
$a2wUserDataTmp->setDataLength( 10 );
```

6.9.23.7 setId

Parameter:

User data id of type String

Return value data type:

Void

Remarks:

Set the user data id.

Example:

```
#--- Create a new UserData
$a2wUserData = new a2w::UserData();
#---- Set document id as user data id
$a2wUserData->setId( "DOCINDEX" );
```

6.9.24 a2w::Vector [NEW]

a2w::Vector module provides vector object handle of page content and exposes the following interfaces to the AFP2web Scripting Facility.

6.9.24.1 getAngle

Parameter:

None

Return value data type:

Integer

Remarks:

Get angle (rotation) in clockwise (0,90,180,270)

Example:

```
$iAngleTmp = $a2wVector->getAngle();
```

6.9.24.2 getColor

Parameter:

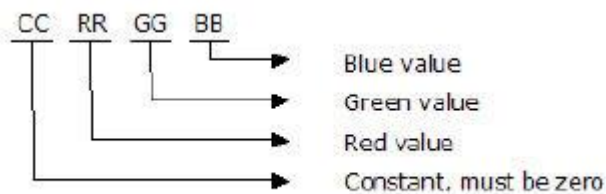
None

Return value data type:

Integer

Remarks:

Returns the color of the vector. Value should be interpreted in RGB color space as given below



Example:

```
$iColorTmp = $a2wVector->getColor();
```

6.9.24.3 **getHeight**

Parameter:

None

Return value data type:

Integer

Remarks:

Get vector object height

Example:

```
$iHeightTmp = $a2wVector->getHeight();
```

6.9.24.4 **getUniqueColorsCount**

Parameter:

iMaxColorPar, maximum colors to be analyzed. Beyond this limit makes no sense to count. Default is -1, means count all unique colors.

Return value data type:

Integer

- In case of error, < 0
- In case of success, Maximum colors counted considering the parameter. The value could be
when iMaxColorsPar = -1, Full count of unique colors
when iMaxColorsPar >= 0, Actual count of unique colors if vector has <= Max colors or Max colors[+ some count], if vector has > Max colors

Remarks:

Get unique colors count of vector object

Example:

```
$iUniqueColorCountTmp = $a2wVector->getUniqueColorsCount();
```

6.9.24.5 getWidth

Parameter:

None

Return value data type:

Integer

Remarks:

Get vector object width

Example:

```
$iWidthTmp = $a2wVector->getWidth();
```

6.9.24.6 getXPos

Parameter:

None

Return value data type:

Integer

Remarks:

Get X position of vector object

Example:

```
$iXPosTmp = $a2wVector->getXPos();
```

6.9.24.7 getYPos

Parameter:

None

Return value data type:

Integer

Remarks:

Get Y position of vector object

Example:

```
$iYPosTmp = $a2wVector->getYPos();
```

6.9.24.8 **remove**

Parameter:

None

Return value data type:

Void

Remarks:

Removes vector from presentation, but positioning vector object on output coordinate system is done still

Example:

```
$a2wVector->remove();
```

6.10 **AFP2web Messages**

6.10.1 **General Information About Error Messages, Return Codes, and Error Files**

afp2web.exe delivers a return code on completion.

AFP2web also writes error messages to a separate text file. You will find this file in the log directory. The file name begins with "error_" and is completed with the processing timestamp. For example: error_20000413_101127.txt

AFP2web passes the return code when issuing messages:

- The return code 0 indicates that no errors occurred. The prefix "I" in the message ID indicates an informational message.

- A negative return code indicates a processing error. The value corresponds with the numerical part of the Message ID in the error message. For example, the return code -23 indicates that the error E023 occurred. The prefix "E" in the message ID indicates an error message, the prefix "W" indicates a warning message.
- The return code > 0 indicates that the conversion has been completed with omission of some input contents. For example, if PDF input spool contains Multi-Media content, AFP2web will convert the input spool ignoring Multi-Media content.

6.10.2 AFP2web-specific Messages

6.10.2.1 Informational Messages

Return_Code	Name	Content
0	I000: Process completed	Process successfully terminated.
0	I001: End of spool:<Spool Filename>	Spool successfully processed.
2	I002: Process completed. Unsupported objects occurred in input spool are ignored	Spool successfully converted, but occurs if input spool contains objects that are not supported by AFP2web. Applicable now, only for PDF inputs.
0	I022: User requested no. of pages parsed. Ignoring other pages	Occurs when the command option -pp is used. No solution required.
0	I083: User requested no. of documents parsed. Ignoring other documents	Occurs when the command option -ed is used. No solution required.
0	I159: Credit operations used with Unlimited Volume ignoring requested operation	Occurs when the processing mode of "CreateCredit" or "DisplayCredit" used with "Unlimited Volume" License It occurs for command line options "-crc" and "-crd" used with "Unlimited Volume" License No Solution Required. Customer can directly invoke the conversion without credit file
0	I199: Reading bytes beyond the bounds	Occurs after input spool or buffer is completely read. No solution required.

6.10.2.2 Warning Messages

Note: These messages do not abort the process.

Return_Code	Name	Content
-------------	------	---------

	W016: Missing mandatory value : <Value>.	MO:DCA-specific warning. Check the AFP document. If the problem cannot be solved, please contact the Support at support@oxseed.com .
	W060: Invalid SFI. <Stream Location and Name>, Offset=<Hex Offset>	MO:DCA specific warning. Check the AFP document.
	W062: Invalid Modca ID. <Stream Location> Offset=<Hex Offset>.	Specified key value is missing under specified Section name. Check AFP input spool for any Structured Field that starts without MO:DCA class identifier "D3". If the problem cannot be solved, please contact the Support at support@oxseed.com .
	W093: Insufficient Font Pattern data for GCGID <GCGID Name> for Resource <Resource Name>. Available Bytes: <Available Bytes>, required Bytes: <Required Bytes>.	FOCA-Specific warning Check the Specified Font Resource document.
	W154: Invalid Index Path (<Info Text>)	Path for writing index files is invalid. Please ensure that the index path specified exists.

6.10.2.3 Error Messages

Return_Code	Name	Content
-1	E001: Unknown error.	Program failure. Please contact the Support at support@oxseed.com .
-2	E002: Unknown exception occurred, leaving...	Program failure. Please contact the Support at support@oxseed.com .
-3	E003: AFP2web is not licensed for <OS name Input format Output format Scripting Facility>.	AFP2web is not licensed for one of the following options: Requested operating system, input format, output format, or Scripting Facility. Please contact the Support at support@oxseed.com .
-4	E004: <File Type> File <Filename> not found.	AFP2web does not find the resource or index file. Please make sure that external resource or index file exists in the specified path.
-5	E005: Missing End Page Group SFI. File Name: <Filename>	MO:DCA specific error. Check the AFP document.

-6	E006: Missing End Document SFI. File Name: <Filename>	MO:DCA specific error. Check the AFP document.
-7	E007: Illegal reserved value. (at <Hex Offset>)	MO:DCA specific error. Check the AFP document.
-8	E008: AFP font representation object is null.	FOCA specific error. Check the AFP font resources
-9	E009: Missing Mandatory Triplet. (at <Hex Offset>)	MO:DCA specific error. Check the AFP document.
-10	E010: Valid spool begining not found. File Name: <Filename>	MO:DCA specific error. Check the AFP document.
-11	E011: Character metrics missing for <Degree> degree rotation; character set is: <Character set name>	FOCA specific error. Check the FOCA AFP document.
-12	E012: AFP2web is not built for <Format Type> <Input and Output formats>.	The combination of key values in the INI file are not valid. Please contact the Support at support@oxseed.com .
-13	E013: <Image Library Error>	Error occurred when image is processed by Maas Image Library. Check Maas Image Library Error Messages.
-14	E014: Missing End Page SFI; filename is: <Filename>	MO:DCA specific error. Check the AFP document.
-15	E015: Missing End Overlay SFI; filename is: <FileName>	MO:DCA specific error. Check the AFP document.
-16	E016: <Compression> compressed <Image Type> image mask is not supported.	Image mask with specified image type/Compression type is not yet supported. Please contact the Support at support@oxseed.com .
-17	E017: Message id <MsgID> define into ErrorCode enum structure, but Error message missing within achErrorMsg[]. Check code.	Program failure. Please contact the Support at support@oxseed.com .
-18	E018: <Compression> compressed <Image Type> images is not supported.	AFP2web does not support specified compression for the image type. Please contact the Support at support@oxseed.com .

-19	E019:< Bits/Pixel> Bits/Pixel <Image Type> image is not supported.	AFP2web does not support specified image type having specified Bits/Pixel (bpp). Please contact the Support at support@oxseed.com .
-20	E020: Unable to find any substitutable external font for [CFT=<Coded Font Title>] [CF=<Coded Font>][CST=<Character Set name Title>][CS=<Character Set>][TFT=<Typeface Title>][TF=<Typeface>]	AFP2web cannot find the substitutable external font. Please check whether the external font file exists for the given Typeface within the external font path (passed using INI attribute "ExtFontPath" or command line argument "-fp")
-21	E021: Java Exception occurred while reading stream (<Stream Name>). Msg:<Info Text>	Java input stream could not be read by AFP2web for the reason given in <InfoText>. Please check if source of input stream is readable.
-22	E022: Maximum Inline image count exceeds the limit <Inline Image Limit>	PDF input spool has inline images that exceeds the maximum allowed. Please contact the Support at support@oxseed.com .
-23	E023: Input file (<Info Text>) not found.	Input file cannot be found. Please check your processing options for afp2web.exe or those in the afp2web.ini.
-24	E024: Invalid Output Format: <Output format>	Invalid output file format requested for conversion. Please check whether valid output format is specified while invoking AFP2web
-25	E025: Insufficient Memory - Unable to create display Device context.	The resolution selected might be too high for the conversion to TIFF. Increase the size of your computer's RAM.
-26	E026: Insufficient Memory - Unable to create Memory Device context.	The resolution selected might be too high for the conversion to TIFF. Increase the size of your computer's RAM.
-27	E027: Insufficient memory while allocating Space for Bitmap data.	The resolution selected might be too high for the conversion to TIFF. Increase the size of your computer's RAM.
-28	E028: Unable to create bitmap for a page (ErrorCode: <Last Error Code of Graphic API> Msg: <Error Msg>)	Output resolution or page size could be too high for RASTER output conversions. Increase the memory size of your computer or reduce the output resolution (-pr option).
-29	E029: Could not install outline font to system; character set is: <Character set >, code page:<Code Page>, Reason:<Reason>	AFP2web could not install the type 1 font created from AFP outline font to the system. Check if user has rights to install external fonts to the system.

-30	E030: Unable to read a TIFF header of <Filename> (Bad TIFF).	This message may occur when trying to convert a TIFF document. The TIFF document cannot be read. Check the TIFF file.
-31	E031: Could not install external font to system. Font File: <Filename>, Reason:<Reason>	AFP2web could not install type1/truetype font to the system. Check if user has rights to install external fonts to the system.
-32	E032: Input file (<Info Text>) is not a TIFF file.	AFP2web auto detects the input file format and identified that the input file is not a TIFF. Ensure that the input file is of TIFF format.
-33	E033: Unsupported compression for image (<Image Name>, <Image Type>)	AFP2web does not support specified compression. Please contact the Support at support@oxseed.com .
-34	E034: Color/Grayscale input image <Image Type, Image Name, Compression> is not supported for <Output Format> output	AFP2web does not support color input images of specified compression for the specified output format. Please contact the Support at support@oxseed.com .
-35	E035: Java Input Stream is null for (<Stream Name>)	NULL Input stream is passed to AFP2web. Please check if valid input stream is passed to JAVA SDK
-36	E036: Java <Method Name> Method Handle is null for (<Stream Name>)	AFP2web could not get specified Method handle from Java Runtime environment. Please check installed Java version is same as or higher than the specified Java version in AFP2web user guide.
-37	E037: Input stream does not support random access. Setting offset failed for stream (<Stream Name>).	If input stream contains some invalid AFP data, AFP2web tries to find valid AFP data within the stream. In such case, it sets the offset directly in the input stream. This exception is thrown when the input stream passed to AFP2web does not support setting an offset within the stream (random access). Please check if AFP input stream contains valid data and that it supports random access
-38	E038: Unable to create bitmap for rasterization of AFP font <Character Set>.(Error Code: <Error Code>, Message: <Error Message>)	AFP2web is not able to create bitmap representation of AFP Font. Please check error message returned by AFP2web and take needed action, else Please contact the Support at support@oxseed.com .
-39	E039: Conversion of font to font not yet supported	AFP2web does not support font conversion of specified font types.

		Please contact the Support at support@oxseed.com .
-40	E040: Conversion of font to font failed. Font Name: Reason: <Reason>	Check if Input font is valid. Please contact the Support at support@oxseed.com .
-41	E041: Unique System Id generation failed. (Error Code: <Error Code>, Message:<Message>)	AFP2web could not get information required to generate "Unique System Id" Please contact the Support at support@oxseed.com .
-42	E042: Invalid Operating System specified on Licensee ("<Licensee>") in <Ini Path>\<Ini Filename>.	The Licensee text in afp2web.ini file is invalid. Ensure Licensee text is entered properly in afp2web.ini
-43	E043: Invalid AFP2web Edition Id: <Edition Id>.	The Licensee text in afp2web.ini file is invalid. Please contact the Support at support@oxseed.com .
-52	E052: Error reading Repeated Triplets (at <Hex Offset>)	MO:DCA specific error. Check the AFP document.
-53	E053: NULL Mapcodedfont object in MCFTriplets()	Memory allocation error. Increase the memory limit in the parameter DocLimit.
-54	E054: Illegal Resource Type in MCF Resource Local Identifier Triplet (at <Hex Offset>)	MO:DCA specific error. Check the AFP document.
-55	E055: Illegal Local id value in MCF Resource Local Identifier Triplet (at <Hex Offset>)	MO:DCA specific error. Check the AFP document.
-56	E056: Error Illegal Page Overlay Type - <Overlay Type> (at <Hex Offset>)	MO:DCA specific error. Check the AFP document.
-57	E057: Not Able to Get PDF Doc Handle. Exiting...	Memory allocation error. Increase the memory limit in the parameter DocLimit.
-58	E058: Resource (<Resource Name>) not found	Path to resource file is incorrect. Check the INI file and command options. Ensure that the correct path is used for resource files.
-59	E059: Null Data Buffer	System-internal failure. Please contact the Support at support@oxseed.com .
-61	E061: Input Stream Open Failed	System-internal failure. Please contact the Support at support@oxseed.com .
-63	E063: SFI data is null	System-internal failure.

		Please contact the Support at support@oxseed.com .
-65	E065: [CHARSET] Section: Missing or Invalid Font Height (<Info Text>) in "mapping.def"	Entries in the mapping.def have invalid values. Check the mapping.def.
-67	E067: Error Invalid Interchange Set Triplet Id <Triplet Id> (at <Hex Offset>)	MO:DCA specific error. Check the AFP document.
-68	E068: File read error	Error occurred when trying to read one of the following files: INI file, mapping.def file, ASCII codepage file (*.cp). Ensure that the required files exist.
-69	E069: Error while writing <Filename> file	Error when trying to write specified file. Ensure that AFP2web has write access to the path of the specified file.
-70	E070: Unable to move a file pointer to a specified location. Filename: <Filename>	File operation error Please contact the Support at support@oxseed.com .
-72	E072: File (<Info Text>) is empty.	Error when reading the specified file. The file is empty. Please check the specified file.
-73	E073: Unable to read data from the stream: File <Filename>, Error Message: <Info Text>	AFP2web could not read the specified file. Ensure that AFP2web has read access to the specified file.
-74	E074: SFI does not begin with 5A. <Stream Location and Name> Offset=<Hex Offset>	AFP2web distinguish 2 kinds of AFP files. The one having an 0x5A Hex value preceding each and every Structured Field Identifier (SFI). And the one not having it. This message occurs when at least the very first SFI has an 0x5A Hex value preceding it but some others in the same file not. Please check the AFP data.
-75	E075: Unable to write data into the stream: File <Filename>, Error Message: <Info Text>	AFP2web could not write the specified file. Ensure that AFP2web has write access to the specified file.
-76	E076: Not able to find a valid SFI within the 4 Kb of data. <Stream Location and Name> Offset=<Hex Offset>	AFP2web could not find any valid Structured Field Introducer (SFI) within the 4 Kb of data following offset <HexOffset>. Process has been aborted. Please check the input spool is valid and contains valid Structured Fields.
-77	E077: Missing or Invalid Licensee(<Licensee>) or Serial	The combination of key values in the INI file are not valid.

	Number(<Serial Number>) in <Ini Path>\<Ini Filename>.	Please ensure that you specify the correct values for Licensee and Serial number in the INI file. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-84	E084: Required Resource (<Resource Name>) of Type (<Resource Type>) not found	Specified resource of specified type is referenced by input AFP spool but does not exist during conversion. Please check the resource file path, inline resource group and ensure to add the missing resource on appropriate path or inline resource group.
-86	E086: PDF Library Error: <Info Text>	Maas C PDF Library error encountered Please contact the Support at support@oxseed.com .
-87	E087: TIFF Library Error <Info Text>	LIBTIFF Library error encountered. Please contact the Support at support@oxseed.com .
-88	E088: Scripting Facility Error (rc=<Error Code>): <Info Text>	Occurs when AFP2web Scripting Facility interface fails to process the script. Check the script file. Please contact the Support at support@oxseed.com if the problem cannot be resolved.
-90	E090: Image Library Error: <Info Text>	Maas Image Library error encountered while decompressing IOCA and IOB images. Please contact the Support at support@oxseed.com with error and log files.
-91	E091: Output buffer of IOCA/IOB object (<Object Name>) is null.	Image Library error encountered while decompressing IOCA and IOB images. Please contact the Support at support@oxseed.com .
-92	E092: Unsupported image type <Image Type> for AFP Container Objects.	Unsupported image type occurred in AFP Container Object. Please contact the Support at support@oxseed.com .
-94	E094: AFP font representation object for Character Set <Character Set Name> is null.	FOCA specific error. Check the AFP font resources
-95	E095: Used code point information is missing. Character Set:<Character Set Name>, Code Page:<Code Page Name>	Internal failure. Please contact the Support at support@oxseed.com .
-96	E096: Error while creating Type 3 output font representation object for Character Set <Character Set Name> and Code Page <Code Page Name>.	FOCA specific error. Check the AFP font resources

-97	E097: Type 3 character bitmap array is null for Character Set <Character Set Name> and Code Page <Code Page Name>.	FOCA specific error. Check the AFP font resources.
-98	E098: Unable to open file <Filename>. Reason: <Info Text>	AFP2web could not open the specified file Please check the reason given in the exception. If you could not solve the problem, please contact the Support at support@oxseed.com .
-99	E099: Invalid AFP file.	AFP2web auto detects the format of input spools. In some cases it may fail and this error is raised on such case. Please check the input file format and ensure it is AFP.
-100	E100: Memory allocation error : <Info Text>	Insufficient memory. Increase the size of your computer's RAM.
-101	E101: Invalid INI File Path (<Info Text>)	Path to INI file is not correct. Check the command line option/ini property specifying the path to afp2web.ini.
-102	E102: Error while opening Statistics file <Filename>	Path for statistics file is not correct. Check the settings in the INI file and command options. Ensure that the path for writing the statistics file exists.
-103	E103: Invalid Output path (<Info Text>).	Path for output files incorrect. Check the INI file and command line parameters. Ensure that the correct path is used for the output files.
-104	E104: INI File (<Ini Filename>) not found in path list <Path List>.	INI file not found in given path list. Ensure that the INI file (afp2web.ini) exists in any directory given in the path list. If not, ensure to configure proper ini path or place a copy of the INI file (afp2web.ini) into a directory on this path.
-105	E105: [<Section Name>] Section not found in the INI File (<Filename>)	A required section is missing in the afp2web.ini file. Check the INI file.
-106	E106: Type 3 Encoding stream is null for Character Set <Character Set Name> and Code Page <Code Page Name>.	FOCA specific error. Check the AFP font resources.
-107	E107: Missing or Invalid Spool File Type parameter	Invalid Spool type specified Check spool type parameter specified in INI file or as command line option

-108	E108: Missing Input File parameter	Missing input file. Specify at least one input file to be processed.
-110	E110: <Section Name> Section: <Key> Key is missing	Specified key value is missing under specified Section name. Please ensure that specified key exists under given section.
-111	E111: <Section Name> Section: <Key> Key: Missing or invalid <Parameter Name> Parameter	Missing or invalid parameter for the specified key under given section. Please ensure that specified parameter exists and is valid for the given key under given section
-112	E112: <Section Name> Section: Missing or Invalid Value <Info Text> in the mapping.def.	Values missing or incorrect in specified section in mapping.def. Please check the specified section in mapping.def.
-113	E113:<Section Name> Section: Duplicate value <Info Text> in the mapping.def	The font map entry exists more than once in the given section of the "mapping.def". Please check specified section in mapping.def and remove the duplicate entry. If you need some assistance to remove duplicate entry, please contact the Support at support@oxseed.com .
-114	E114: Error while setting font. Character Set: <Character Set Name> Reason: <InfoText>	AFP2web could not set font to output. Please check the reason given in the exception. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-115	E115: Product version expired. Please contact AFP2web Support.	Product version expired. Please check the for reason given in the exception and ensure AFP2web license time limit is not exceeded. If you could not solve the problem, please contact the Support at support@oxseed.com .
-117	E117: Error while generating Unique System Id: Error Code: <Error Code> <Error Message>	AFP2web could not generate unique system identifier by collecting required info from system. Please contact the Support at support@oxseed.com with error code and error message.
-118	E118: Output font representation is null for Character Set=<Character Set Name>, Code Page=<Code Page Name> and Output Font Type=<Output Font Type>	Corrupted or invalid external font(s). Please check the external font files in "extfont" path. If you could not solve the problem, please contact the Support at support@oxseed.com .
-119	E119: Invalid Type1 PFM File <Filename>	Corrupted or invalid PFM file.

		Please check the specified PFM file. If you could not solve the problem, please contact the Support at support@oxseed.com .
-120	E120: Error while parsing True Type font file <Filename>, Reason: <Info Text>.	Corrupted or invalid True Type font file. Please check the specified TrueType file. If you could not solve the problem, please contact the Support at support@oxseed.com .
-155	E155: Error while opening the Index File (<Filename>), leaving...	AFP2web could not open the index file for writing. Please ensure that the path exists and that AFP2web has write access to the path of the index file. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-158	E158: File Pointer is null	Internal failure. Please contact the Support at support@oxseed.com .
-159	E159: License Manager Error (rc=<Error Code>): <Info Text>	License verification or Credit file verification or Voucher file verification failed in Transaction Volume Based Conversion, Please contact the Support at support@oxseed.com with error and log files
-167	E167: Conversion from <Input File Type> to <Output File Type> is not supported	Unsupported file conversion. No solution required. Please check whether latest AFP2web release has been extended with required conversions.
-190	E190: Internet Connection failed. Reason:<Info Text>	AFP2web could not create an Internet session. Please check the reason specified in the exception. If you could not solve the problem, please contact the Support at support@oxseed.com .
-191	E191: FTP Connection failed. Reason: <Info Text>	AFP2web could not establish a FTP connection. Please check the reason specified in the exception. If you could not solve the problem, please contact the Support at support@oxseed.com .
-192	E192: Unable to open FTP file <Filename>, Reason: <Info Text>.	AFP2web could not open a file over FTP connection. Please check the reason specified in the exception. If you could not solve the problem, please contact the Support at support@oxseed.com .
-193	E193: Unable to read FTP file <Filename>, Reason: <Info Text>.	AFP2web could not read a file over FTP connection. Please check the reason specified in the exception. If you could not solve the problem, please contact the Support at support@oxseed.com .

-198	E198: Null FTP Connection.	Internal failure. Please contact the Support at support@oxseed.com .
-200	E200: Null Input Buffer	It occurs when null input buffer is passed to AFP2web C/Java SDK Please make sure that input buffer passed to AFP2web C/Java SDK contains valid data.
-201	E201: AFP library Error: <AFP Library Error Message>	Error occurred in AFP library while creating AFP output. Please contact the Support at support@oxseed.com .
-207	E207: Output file name is null	Output file name not given in the command line options. Please ensure that output file name is given in command line options.
-208	E208: Null DocList.	Empty document list is passed to AFP2web Java SDK Merge process API. Please assert document list (that contain Document id) before calling AFP2web Java SDK Merge process API.
-209	E209: Null FileList.	Empty file list is passed to AFP2web Java SDK API for 1-to-1 or 1-to-n conversion. Please assert file list (that contain filename) before calling AFP2web Java SDK API.
-231	E231: libXPDF Error (rc=<Error Code>): <Error Message> at <API Name>	Occurs while parsing PDF files that contains invalid entries. Check if PDF file is valid and viewable in Acrobat Reader.
-232	E232: Text Rasterization failed. Font: , Text:<Info Text>, Reason:<Reason>	External font file for the specified font not found. Please ensure to add the specified font file to external font directory.

6.10.3 Maas Image Library Messages

Return_Code	Name	Content
-1	MIL001: Unknown error.	Please contact the Support at support@oxseed.com .

-2	MIL002: Message id <Message Id> define into ErrorCode enum structure, but Error message missing within achImgLibErrorMsg[]. Check code.	Program failure. Please contact the Support at support@oxseed.com .
-5	MIL005: Image header information is null	Program failure. Please contact the Support at support@oxseed.com .
-6	MIL006: DIB buffer is null	Program failure. Please contact the Support at support@oxseed.com .
-7	MIL007: Error while reading file <Filename>. Return Code:<Return Code> Message:<Reason>	Maas Image Library could not read the specified file. Ensure that the Maas Image Library has read access to the file. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-8	MIL008: File name parameter is null or empty	File name passed to Maas Image Library is null. Ensure that right file name is passed to the Maas Image Library.
-9	MIL009: Unsupported image type <Image Type>	Maas Image Library does not support the specified image type. Please contact the Support at support@oxseed.com .
-10	MIL010: Decompression of <BPP> Bits/Pixel <Compression> <Image Type> images is not supported	Unsupported decompression requested. Please contact the Support at support@oxseed.com .
-12	MIL012: Source image buffer is null or buffer length parameter is zero	Invalid parameters passed to Maas Image Library. Ensure that proper image buffer is passed to Maas Image Library. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-13	MIL013: Invalid compression parameter id <Parameter Id>	Invalid compression parameter passed to Maas Image Library. Ensure valid compression parameters are passed to Maas Image Library.
-14	MIL014: IOCA Library handle is null	Program failure. Please contact the Support at support@oxseed.com .
-15	MIL015: IOCA Library Error: Return Code:<Return Code> Message:<Info Text>	Error occurred in IOCALib Library. Please check the input image file/stream for message given in the exception. If you cannot solve

		the problem, please contact the Support at support@oxseed.com .
-16	MIL016: Decompression of <SPP> Samples/Pixel <Image Type> images is not supported	Unsupported decompression requested. Please contact the Support at support@oxseed.com .
-17	MIL017: CxImage Library Error: <Info Text>	Error occurred in CxImage Library. Please check the input image file/stream for the message given in the exception. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-18	MIL018: Error while opening file <Filename>. Return Code:<Return Code> Message:<Reason>	Maas Image Library could not open the specified file. Ensure that the specified file exists.
-19	MIL019: Error while creating TIFF handle for <Filename "Memory Stream">	Program failure. Please contact the Support at support@oxseed.com .
-20	MIL020: TIFF Library Error: <Info Text>	Error occurred in TIFF Library. Please check the input image file/stream for the message given in the exception. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-21	MIL021: Memory allocation error	Insufficient Memory. Increase the size of your computer's RAM.
-22	MIL022: Compression <Compression> is not supported for <Output Format>	Unsupported compression requested for a specified output format. Please contact the Support at support@oxseed.com .
-23	MIL023: <Image Type> Image encoder is null	Program failure. Please contact the Support at support@oxseed.com .
-24	MIL024: <Image Type> Image decoder is null	Program failure. Please contact the Support at support@oxseed.com .
-25	MIL025: JPEG compression failed: <Info Text>	Program failure. Please check the message given in the exception. If you cannot solve the problem, please contact the Support at support@oxseed.com .

-26	MIL026: <Method Name> not implemented for <Class Name>	Program failure. Please contact the Support at support@oxseed.com .
-27	MIL027: JPEG Library handle is null	Program failure. Please contact the Support at support@oxseed.com .
-28	MIL028: Compression parameter array is null	Program failure. Please contact the Support at support@oxseed.com .
-29	MIL029: Cannot convert IOCA <Color Space> image planes to <Color Space> color space	Program failure. Please contact the Support at support@oxseed.com .
-30	MIL030: CMKY image buffer passed to CMYK Converter is null	Program failure. Please contact the Support at support@oxseed.com .
-31	MIL031: IOCA RGB image plane buffer is null	Occurs while parsing AFP IOCA sample that contains invalid data. Check if AFP IOCA contains valid data
-32	MIL032: Unsupported file format <File Type>	Maas Image Library does not support the specified file type. Please contact the Support at support@oxseed.com .
-33	MIL033: Grayscale conversion of<Bits/Pixel> Bits/Pixel images is not supported	Maas Image Library does not support Grayscale conversion of images having specified bits/pixel. Please contact the Support at support@oxseed.com .
-34	MIL034: Black/White conversion of <Bits/pixel> Bits/Pixel images is not supported	Maas Image Library does not support B/W conversion of images having specified bits/pixel. Please contact the Support at support@oxseed.com .
-35	MIL035: Zlib Library Error: <Error Message>	Error occurred in Zlib library, while compressing image data using "FlateEncode" algorithm. Please check the message given in the exception. It may occur due to insufficient memory. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-36	MIL036: TIFF image with tiles is not supported	TIFF image with Tiles are not yet supported.

		Please contact the Support at support@oxseed.com .
-37	MIL037: IOCA logical resolution unit not supported	Maas Image Library does not support IOCA Logical resolution unit. Please contact the Support at support@oxseed.com .
-38	MIL038: Decompression of <Compression Type> <Image Type> images is not supported	Unsupported decompression requested. Please contact the Support at support@oxseed.com .
-39	MIL039: Error while writing file <Filename>	Maas Image Library could not write the specified file. Ensure that Maas Image Library has write access to the specified path. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-40	MIL040: Incorrect call to buffer access methods	Program failure. Please contact the Support at support@oxseed.com .
-41	MIL041: Rotation Failed	Program failure. Please contact the Support at support@oxseed.com .
-42	MIL042: Unable to prepare header for OJPEG TIFF data	Invalid/Corrupted TIFF file/TIFF Image buffer. Please check the tiff header of tiff file. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-43	MIL043: Error while extracting TIFF image data from strip <StripNumber>	Invalid/Corrupted TIFF file/TIFF Image buffer. Please check the TIFF File.If you cannot solve the problem, please contact the Support at support@oxseed.com .
-44	MIL044: JPEG Library Error: <Info Text>	Error occurred in JPEG Library. Please check the input image file/stream for the message given in the exception. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-45	MIL045: Images with <Bits/Pixel> Bits/Pixel can not converted to <Image Type>	Unsupported conversion requested. Please contact the Support at support@oxseed.com .
-46	MIL046: TIFF Library handle is null	Program failure.

		Please contact the Support at support@oxseed.com .
-47	MIL047: Error while writing <SFI> SFI	Program failure. Please contact the Support at support@oxseed.com .
-48	MIL048: Image data is null	Program failure. Please contact the Support at support@oxseed.com .
-49	MIL049: Error while decoding image. Name=<Image Name>, Line Number=<Line Number>	Invalid or corrupted image file/buffer. Please check if image file contains valid data and ensure that Maas Image Library has read access to the specified image(filename). If you cannot solve the problem, please contact the Support at support@oxseed.com .
-50	MIL050: <Image Type> compression is not supported	Maas Image Library does not support specified compression. Please contact the Support at support@oxseed.com .
-51	MIL051: Error while decompressing IOCA <Color Space> planes of image data	Invalid IOCA Image Data. Please check if IOCA has valid Image data. If you cannot solve the problem, please contact the Support at support@oxseed.com .
-52	MIL052: Decompression of <BPP> Bits/Pixel <Image Type> images is not supported.	Unsupported decompression requested. Please contact the Support at support@oxseed.com .
-53	MIL053: Decompression of <Image Type> images having <Photometric Name> Photometric Interpretation is not supported	Unsupported decompression requested. Please contact the Support at support@oxseed.com .
-55	MIL055: Insufficient data read from <Stream Name>. Expected Bytes=<Expected Bytes>, Bytes Read=<Bytes Read>	Image data stream has lesser number of bytes available than needed for a complete scanline. Check if input image file/stream has valid data.
-57	MIL057: TIFF Images with Planar format (separate planes of data) are not supported	TIFF Images with Planar format(separate planes of data) are not supported. Please contact the Support at support@oxseed.com .
-58	MIL058: Image Cropping is not yet supported for images with <Bits/Pixel> bits per pixel	Image Cropping is not yet supported for images with specified bits per pixel. Please contact the Support at support@oxseed.com .

-59	MIL059: Image Mixing is not yet supported for images with <Bits/Pixel> bits per pixel	Image Mixing is not yet supported for images with specified bits per pixel. Please contact the Support at support@oxseed.com .
-60	MIL060: <Encoding Name> Encoding failed. <Error Message>	Error occurred while compressing image data using the specified encoding algorithm. Please contact the Support at support@oxseed.com giving the error message.
-61	MIL061: Decompression of <Bits/Pixel> Bits/Pixel image mask is not supported	Decompression of image masks is not yet supported with specified bits per pixel. Please contact the Support at support@oxseed.com .
-62	MIL062: Decompression of <Image Type><CompressionType>" image mask is not supported	Decompression of image mask is not supported for the given image type/compression type. Please contact the Support at support@oxseed.com .
-63	MIL063: Could not create new <Context> Context for an image. Maximum <Context> Contexts limit reached	It occurs either when Maas Image Library tries to use greater than maximum contexts allowed or when image contexts are not properly freed due to the handling of corrupted images. Please contact the Support at support@oxseed.com .
-64	MIL064: <Image Type> Encoder is not initialized	It occurs when Maas Image Library tries to use an image encoder to compress images without initializing the encoder. Please contact the Support at support@oxseed.com .

6.11 AFP2web Conversion Process

6.11.1 Auto image grouping

For PDF2AFP conversion, when the input PDF contains consecutively placed images having either 1 pixel width or height, then they will be automatically merged together as one image in order to have better image presentation quality and size.

6.11.2 Windows maximum memory usages limitation per conversion process

Windows Operating system (32-bit) limits any process to utilize a maximum of 2 GB (even though system has more than 2 GB physical memory) only. This implies that AFP2web can not use more than 2 GB memory when executed on Windows Operating System. i.e. Conversion which use "DOC_COLD" option and don't split documents will have problem of memory allocation error as AFP2web parses all the pages into memory before writing in such case.

Reference:

<http://support.microsoft.com/kb/555223>

6.11.3 AFP output, MOD:CA IS1 /IS2 extended feature

If AFP input of type MO:DCA IS1 or IS2 contain font mapping using GCSGID (Global Character Set Global ID) or FGID (Font Global ID) instead of AFP coded font name or AFP character set/code page name pair then AFP2web can handle this font mapping using new "GCSGID" section on "mapping.def". This kind of mapping process is a new extension to AFP2web.

6.11.4 Creating Color, Grayscale and B/W output

AFP2web creates Color, Grayscale and Black/White output based on configuration. This section describes what options should be used for various output formats to achieve Color or Grayscale or B/W output.

Output Format	Color Output	GrayScale Output	Black/White Output
PDF	-c or Color=on	Color=off	-- na --
AFP	-c or Color=on	Color=off	-afp:bw or Color=off FormatType=BW
TIFF	-c -tiff[:lzw jpg pack uncompressed] or Color=on OutputFormat=TIFF FormatType=[lzw jpg pack uncompressed]	-tiff[:lzw jpg pack uncompressed] or Color=off OutputFormat=TIFF FormatType=[lzw jpg pack uncompressed]	-tiff[:g3 g4] or Color=off OutputFormat=TIFF FormatType=[g3 g4]

7 Appendix

This appendix includes:

- Tips for migrating the Scripting Facility modules to AFP2web Version 4.x
- Guide to the Scripting Facility examples
- Frequently asked questions

7.1 Migrating Scripting Facility Modules to Latest Version

The following table shows the differences between scripting modules from the previous version 2.1 to those of Version 4.x. This can be used as a checklist for migration.

Version 2.1	Version 4.x
use a2w::mhtIni;	use a2w::Config;
use a2w::mhtDocument;	use a2w::Document;
use a2w::mhtPtocaPage;	use a2w::Page;
use a2w::mhtTextObject;	use a2w::Text;
use a2w::mhtNOP;	use a2w::NOP;
use a2w::mhtResOverlay;	use a2w::Overlay;
use a2w::mhtAttributeElement;	use a2w::Index;
use a2w::mhtAFPFont;	use a2w::Font;
use a2w::mhtPagePageGroupIndex;	use a2w::Index;
use a2w::mhtIndexElement;	use a2w::Index;
sub initialize(mhtIni)	sub initialize(a2w::Config, a2w::Kernel)
\$a2wIniPar->setAutosplit(\$TRUE);	\$a2wConfigPar->setAttribute("Autosplit", "on");

<code>\$svIndexPathTmp = \$a2wConfig->getIndexPath();</code>	<code>\$svIndexPathTmp = \$a2wConfigPar->getAttribute("IndexPath");</code>
<code>\$svOutputFilePathTmp = \$a2wConfig->getOutputFilePath();</code>	<code>\$svOutputFilePathTmp = \$a2wConfigPar->getAttribute("OutputFilePath");</code>
<code>\$svScriptArgsTmp = \$a2wConfig->getScriptArgs();</code>	<code>\$svScriptArgsTmp = \$a2wConfigPar->getAttribute("ScriptArgument");</code>
<code>\$a2wPageTmp = \$a2wDocument->getFirstPageObject();</code>	<code>\$a2wPageTmp = \$a2wDocumentPar->getFirstPage();</code>
<code>\$a2wPageTmp = \$a2wDocument->getNextPageObject();</code>	<code>\$a2wPageTmp = \$a2wDocumentPar->getNextPage();</code>
<code>\$a2wPage->getFirstTextObject();</code>	<code>\$a2wPagePar->getFirstText();</code>
<code>\$a2wPage->getNextTextObject();</code>	<code>\$a2wPagePar->getNextText();</code>
<code>\$a2wPageGroupIndexTmp = \$a2wDocument->getFirstIndexObject();</code>	<code>\$a2wIndexTmp = \$a2wDocumentPar->getFirstIndex();</code>
<code>\$a2wPageGroupIndexTmp = \$a2wDocument->getNextIndexObject();</code>	<code>\$a2wIndexTmp = \$a2wDocumentPar->getNextIndex();</code>
<code>\$a2wIndexTmp = \$a2wPageGroupIndexTmp->getFirstAttributeElement();</code>	directly access to index thru: <code>\$a2wIndexTmp = \$a2wDocumentPar->getFirstIndex();</code> or <code>\$a2wIndexTmp = \$a2wPagePar->getFirstIndex();</code>
<code>\$a2wIndexTmp = \$a2wPageGroupIndexTmp->getNextAttributeElement();</code>	directly access to index thru: <code>\$a2wIndexTmp = \$a2wDocumentPar->getNextIndex();</code> or <code>\$a2wIndexTmp = \$a2wPagePar->getNextIndex();</code>
<code>\$a2wIndex->setAttributeName("Insured");</code>	<code>\$a2wIndex->setName("Insured");</code>
<code>\$a2wIndex->setAttributeValue("John Doe");</code>	<code>\$a2wIndex->setValue("John Doe");</code>
<code>\$svName = \$a2wIndex->getAttributeName();</code>	<code>\$svName = \$a2wIndex->getName();</code>

<code>\$svValue = \$a2wIndex->getAttributeValue();</code>	<code>\$svValue = \$a2wIndex->getValue();</code>
<code>\$sIndexNameTmp = \$a2wIndexTmp->getAttributeName();</code>	<code>\$sIndexNameTmp = \$a2wIndexTmp->getName();</code>
<code>\$sIndexValueTmp = \$a2wIndexTmp->getAttributeValue();</code>	<code>\$sIndexValueTmp = \$a2wIndexTmp->getValue();</code>
<code>\$a2wOverlay->getFirstTextObject();</code>	<code>\$a2wOverlay->getFirstText();</code>
<code>\$a2wOverlay->getNextTextObject();</code>	<code>\$a2wOverlay->getNextText();</code>
<code>\$a2wDocument->addIndex("Insured", "JohnDoe");</code>	<code>\$a2wIndexTmp = new a2w::Index();</code>
	<code>\$a2wIndexTmp->setName("Insured");</code>
	<code>\$a2wIndexTmp->setValue("John Doe");</code>
	<code>\$a2wDocumentPar->addIndex(\$a2wIndexTmp);</code>
<code>\$a2wPage->addIndex("Insured", "John Doe");</code>	<code>\$a2wIndexTmp = new a2w::Index();</code>
	<code>\$a2wIndexTmp->setName("Insured");</code>
	<code>\$a2wIndexTmp->setValue("John Doe");</code>
	<code>\$a2wPagePar->addIndex(\$a2wIndexTmp);</code>
<code>\$a2wPage->includeObject(0x92, "BGFORM", \$XPos, \$YPos, \$Rotation, "JPEG");</code>	<code>\$a2wPagePar->addImage("BGFORM", \$XPos, \$YPos, \$svWidth, \$svHeight, \$Rotation);</code>
<code>\$a2wPage->includeObject(0xDF, "O1BGFORM", \$XPos, \$YPos, \$Rotation);</code>	<code>\$a2wPagePar->addOverlay("O1BGFORM", \$XPos, \$YPos);</code>

<code>\$a2wPage->includeObject(0x5F, "S1BGFORM", \$XPos, \$YPos, \$Rotation);</code>	<code>\$a2wPagePar->addPSEG("S1BGFORM", \$XPos, \$YPos);</code>
<code>\$a2wPage->addObject(\$a2wTextTmp);</code>	<code>\$a2wPagePar->addText(\$a2wTextTmp);</code>
<code>\$a2wTextTmp->setColor(0x00BBGGRR);</code>	<code>\$a2wTextTmp->setColor(0x00RRGGBB);</code>

7.2 Tutorials: Using the Scripting Facility

This appendix serves as a quick-start guide to using the Scripting Facility examples, which are delivered with AFP2web Version 4.x. Each Scripting Facility example performs a small, basic task. The samples have inline comments for further help.

7.2.1 Introduction to the Scripting Facility Tutorials

7.2.1.1 Environment For Running the Scripting Facility Examples

AFP2web Version 4.x is quickly installed. For the general operating system installation steps and requirements, please refer to the AFP2web user guide.

The Scripting Facility examples are written in Perl, the files with the file extension *.pm, which are located in the root folder of the AFP2web installation.

Note: It is not necessary to install PERL for the tutorial examples. AFP2web delivers a Perl DLL, which will be used if it is found in the current working directory or if it is found in the environment variable PERLLIB. If you plan to use your own Perl environment, please refer to the system prerequisites in the installation chapter of the AFP2web User Guide.

AFP2web derives its run time parameters from the afp2web.ini file (INI file). Any command line option entered will override the corresponding parameter in the INI file. By default, output is written to the subfolder /pdf and in case of logging messages to the subfolder /log. For a detailed documentation of parameters and command line options, please refer to the AFP2web user guide.

7.2.1.2 Running a Scripting Facility Example

Selecting Commands in the demo batch command file

The in-line header of each Perl module shows how to start AFP2web and to run the script. For example, the following listing shows how to start AFP2web with the Scripting Facility example sortPageTextObjs.pm:

Demo batch file (Starting AFP2web with a Script):

On Windows:

```
afp2web.exe -q -c -doc_cold -sp:sortPageTextObjs.pm samples\insure.afp
```

On Unix:

```
./afp2web -q -c -doc_cold -sp:sortPageTextObjs.pm samples/insure.afp
```

The command line options and their meanings:

Parameters	Description
-q	Run AFP2web in quiet mode
-doc_cold	Activate the AFP2web Scripting Facility
-sp	The name of the Scripting Facility file to use, if you do not want to use the default afp2web.pm
-sa	Arguments to pass to the script, if your script requires any parameters.

For more information on INI parameters and command line options, please refer to the AFP2web user guide. For example, you might want to use the command line option `-ll` to create a processing log.

The following lines show how you can trap and display the return code. If you want to use it, just insert these lines before the exit command.

Demo.bat (Trapping an Error Level):

```
IF %ERRORLEVEL% EQU 0 (
@ECHO AFP2web: Done %ERRORLEVEL%
) ELSE (
@ECHO AFP2web: Error %ERRORLEVEL%
)
```

Running a script with demo.bat

Step	Description
1	<p>Edit the batch command file demo.bat, resp. demo. Enter the appropriate command to start AFP2web for the script example.</p> <p>To start AFP2web, just start the demo batch command file. Under Windows, just double-click this file in the Windows Explorer.</p> <p>The system console is opened showing messages issued by AFP2web.</p> <p>If you turned AFP2web's logging on, then AFP2web will log information into a log file in the <code>/log</code> sub folder by default.</p>
2	View the results.

7.2.1.3 The Basic Structure of a Script

When you formulate a Scripting Facility module, please use the template afp2web.pm. It shows where to implement basic functions (such as including packages, reading start parameters, logging, indexing and document page splitting).

A script must have the following subroutines:

- sub afp2web()
- sub initialize()
- sub initializeDoc()
- sub initializePage()
- sub finalizePage()
- sub finalizeDoc()
- sub finalize()

The purpose of each routine is to trap processing events of AFP2web.

Each script requires packages to be included, which provide the Scripting Facility functionality. For example:

```
use a2w::Config;
use a2w::Kernel;
```

Additional packages provided add functionality to the script module.

In this chapter, we focus on the particular subroutines. For details about the coding, please refer to the module itself. There are plenty of in-line comments which make the module self explanatory.

7.2.2 addAnnotations.pm: Adding a Hypertext Annotation

7.2.2.1 Understanding addAnnotations.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub initialize()	Used to set initial values for the hyperlink appearance.
sub afp2web()	Main entry for AFP2web. In our example, this routine will check for the first page of a document and will add a hyperlink annotation to the text found in predefined position.

The subroutine initialize() is used to set initial values for the hyperlink appearance.

addAnnotations.pm - initialize():

```
sub initialize(){
    # Assuming width of a character as 2.53 millimeter
    # It is used to calculate Hyper Link Rectangle Width as below
    # Hyper Link Rect Width (in millimeters)=Hyper Link Text Length*$CharWidth
    $CharWidth = 2.53;
    # Hyper Link Rectangle Height values in millimeters ---
    $HPLHeight= 2.6;
    # Hyper Link Rectangle Border Width value in millimeters (approximately
    1.0 Point)
    #--- Set BorderWidth to zero if border is not needed.
    $BorderWidth = 0.35;
    #--- Hyper Link Action URL
    $UrlText = "http://afp2web.com";
    # Hyper Link Border color as RGB value (default color blue is assigned
    here)
    $HPLColor=0x000000FF; #---- Blue (0x00RRGGBB)
    # $HPLColor=0x00FF0000; #---- Red
    # $HPLColor=0x0000FF00; #---- Green
    # $HPLColor=0x00FFFFFF; #---- White
    # $HPLColor=0x00000000; #---- Black
}
```

The routine `afp2web()` is used to process each document page. This routine check for the first page of a document and will add a hyperlink annotation to the text found in predefined position. In the following we only show the command that inserts the annotation.

addAnnotations.pm - `afp2web()` - Command used to insert the annotation:

```
$a2wPagePar->addAnnotation( $CheckXPosTmp,#---- Upper left X position of
annotation box (in AFP page units)
$CheckYPosTmp,#---- Upper left Y position of annotation box (in AFP page
units)
$WidthTmp,#---- Width of annotation box (in AFP page units)
$HeightTmp,#---- Height of annotation box (in AFP page units)
$UrlText,#---- URL
$BorderWidthTmp,#---- Annotation border width
$HPLColor);#---- Annotation border color (Format=0x00BBGGRR)
```

At this point we do not want to repeat the information, which you can find in the script itself. Please refer to the in-line comments of the script. There, explanations have been added for each attribute.

7.2.2.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

```
On Windows:
afp2web.exe -q -c -doc_cold -sp:addAnnotations.pm samples\insure.afp
On Unix:
./afp2web -q -c -doc_cold -sp:addAnnotations.pm samples/insure.afp
```

7.2.2.3 Results of addAnnotations.pm

You will find the output file in the output sub folder (the default will be in the subfolder /pdf). The hypertext annotation will look like:

Insurance Specialists Company will pay the benefits provider

We have issued this Policy to You in consideration of the pa
Your application. Your application is part of Your Policy.

Insured

[Beth N McShine](#)



<http://afp2web.com>

Policy Number **332-5767288-77**

7.2.3 addBookmarks.pm: Adding Bookmarks to the PDF output document

`addBookmarks.pm` generates bookmarks in the PDF document.

7.2.3.1 Understanding addBookmarks.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub initializeDoc()	Saves the current document object. Resets page numbering Adds a bookmark for the document
sub afp2web()	Main entry for AFP2web. The return code in this routine tells AFP2web to begin a new document with this page or not, to skip the current page, or to stop AFP2web processing. Adds a bookmark for the page.
sub initialize()	This routine is also used to set the flags for AutoSplit and PDFBookmark. We do not show this routine in this document.

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document. We use the Perl variable @_ to save the document, which is passed as argument . And we reset the page counter to zero. The following lines show how we set the bookmarks for the document.

addBookmarks.pm - initializeDoc():

```
#---- Add document level bookmarks ----#
if ($a2wDocumentPar->getId() == 1){
    #---- Add creator
    $a2wDocumentPar->addBookmark( "Creator", "Maas Holding GmbH" );
    #---- Add creation date & time
    my $svLocalTimeTmp = localtime;
    $a2wDocumentPar->addBookmark( "Created On", $svLocalTimeTmp );
    #---- Add spool filename
    $a2wDocumentPar->addBookmark( "Spool", $svSpoolFilename );
    return 0;
}
```

It is equally important that we save the current page object so that we can later access its elements. This is done in the subroutine initializePage().

addBookmarks.pm - initializePage():

```
sub initializePage(){
#---- Get Parameter of initializePage( Par: a2w::Page )
($a2wPagePar) = @_;
```

The routine afp2web() is used to process each document page. This routine formulates the return code value to be passed to AFP2web. Bookmarks are added to each page.

addBookmarks.pm - afp2web() - Adding Bookmarks:

```
#-----
# Main entry method
# Return values:
# < 0:error
# 0:append page to Current Document
# 1:skip page
# 2:first page / new document
#-----
sub afp2web(){
    if ( $bLog == $TRUE ){
        print "afp2web(): PageId " . $a2wPagePar->getParseId() . "\n";
```

```

}
$APPEND= 0;# append page to Current Document
$SKIP= 1;# skip page
$NEWDOC= 2;# new document
#---- Set default return value
my $svRetTmp = $APPEND; # default: append page
#---- Add page level bookmarks ----#
if ($a2wDocumentPar->getId() == 1){
    #---- Add page number (in spool)
    $a2wPagePar->addBookmark( "Page", $a2wPagePar->getParseId() );
}
return $svRetTmp;
}

```

7.2.3.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

On Windows:

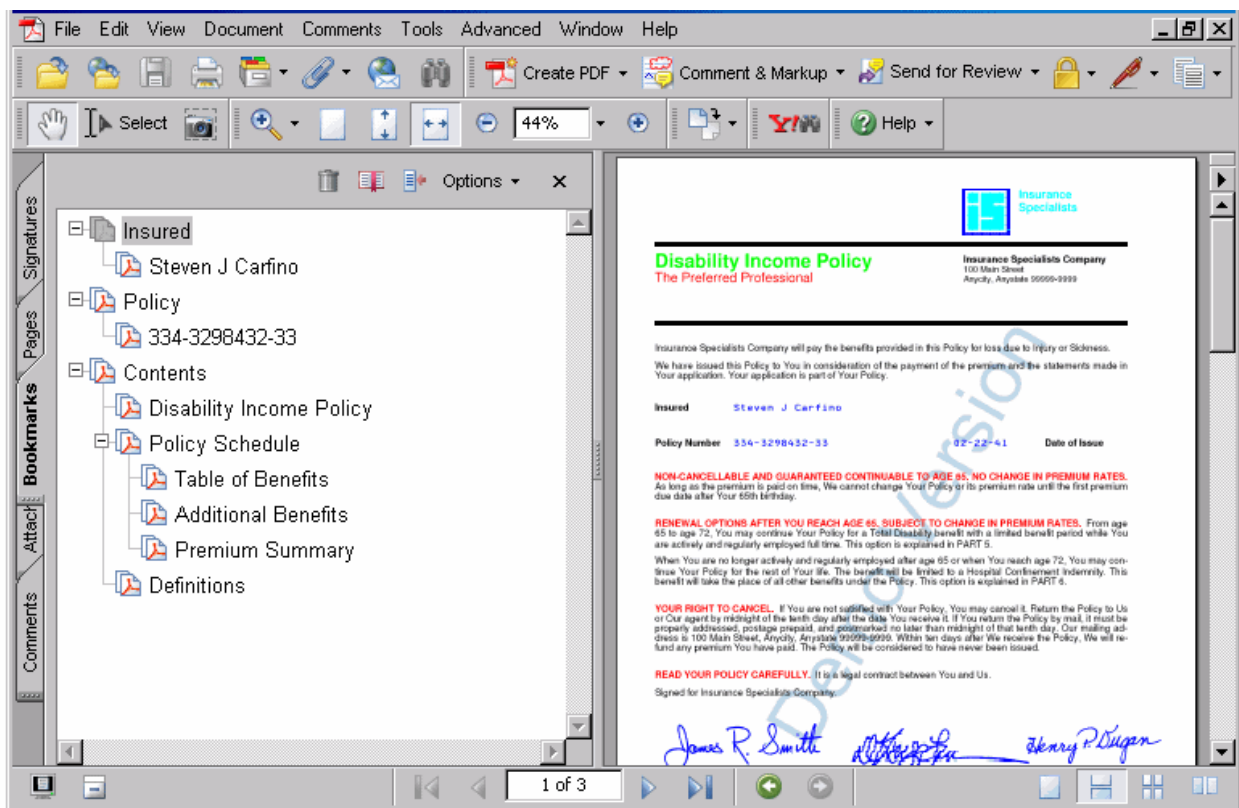
```
afp2web.exe -q -c -doc_cold -sp:addBookmarks.pm samples\insure.afp
```

On Unix:

```
./afp2web -q -c -doc_cold -sp:addBookmarks.pm samples/insure.afp
```

7.2.3.3 Results of addBookmarks.pm

You will find the output PDF file in the output sub folder (the default will be in the sub folder /pdf). When opened in Acrobat Reader, you can find the bookmarks as follows:



7.2.4 addUserData.pm: Adding User Data to the Document

addUserData.pm shows how to create user data and add it to the Document object. You can add your own data such as indexes or any other type of information to adapt your documents according to your own information requirements.

You use the perl module "addUserData.pm" mostly together with the A2W JAVA SDK.

AFP2web gives access to the UserData information in the „A2WDocumentWriter::write“ method of the A2W JAVA SDK callback handler .

7.2.4.1 Understanding addUserData.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub initializeDoc()	Fetches the document indexes. Creates UserData object with fetched document index. Adds user data to document.
sub afp2web()	Fetches the page indexes. Creates UserData object with fetched page index. Adds user data to document.

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document. This routine calls the subroutine fetchDocumentIndexes() to get document index information, creates a UserObject with the document index information and adds the newly created UserData to the document object.

addUserData.pm - initializeDoc():

```
#INFO: This is a sample for fetching the document indexes as one String
my $sIndexDataTmp = fetchDocumentIndexes();
#INFO: This is a sample for adding user data to document
#---- Create user data object
my $a2wUserDataTmp = new a2w::UserData();
#---- Set document id as user data id
$a2wUserDataTmp->setId( "DOCINDEX" );
#---- Set document index as data
$a2wUserDataTmp->setData( $sIndexDataTmp, length( $sIndexDataTmp ) );
#---- Add data to document
$a2wDocumentPar->addUserData( $a2wUserDataTmp );
```

The routine fetchDocumentIndexes() gets the index information of the document to be added in the UserObject.

addUserData.pm - fetchDocumentIndexes():

```
#-----
# Fetch Document Indexes
#-----
sub fetchDocumentIndexes(){
    if ( $bLog == $TRUE ){
        print "fetchDocumentIndexes()\n";
    }
}
```

```

#---- Prepare Document Id as index
my $sRetIndexDataTmp = "DocId=" . $a2wDocumentPar->getId() . "\r\n";
#---- Fetch first Document Index
my $a2wDocIndexTmp = $a2wDocumentPar->getFirstIndex();
#---- Add PageGroup Name
if ( $a2wDocIndexTmp != 0 ){
    #---- Add PageGroup Name
    $sRetIndexDataTmp .= "PageGroup=" .
        $a2wDocIndexTmp->getIndexedObjectName() . "\r\n";
}
while ( $a2wDocIndexTmp != 0 ){
    #---- Add Index Record
    $sRetIndexDataTmp .= $a2wDocIndexTmp->getName() . "=" .
        $a2wDocIndexTmp->getValue() . "\r\n";
    #---- Fetch next Document Index
    $a2wDocIndexTmp = $a2wDocumentPar->getNextIndex();
}
return $sRetIndexDataTmp;
}

```

The routine `afp2web()` is used to process each document page. This routine calls the subroutine `fetchPageIndexes()` to get page index information, creates a `UserData` with page index information and adds newly created `UserData` to the document object.

addUserData.pm - afp2web() - Adding UserData:

```

#-----
# Main entry method
# Return values:
# < 0:error
# 0:append page to Current Document
# 1:skip page
# 2:first page / new document
#-----
sub afp2web(){
    if ( $bLog == $TRUE ){
        print "afp2web(): PageId " . $a2wPagePar->getParseId() . "\n";
    }
    $APPEND= 0;# append page to Current Document
    $SKIP= 1;# skip page
    $NEWDOC= 2;# new document
    #---- Set default return value
    my $svRetTmp = $APPEND; # append page to Current Document
    #---- Increment Page Id
    $PageId++;
    #INFO: This is a sample for fetching the page indexes as one String
    my $sIndexDataTmp = fetchPageIndexes();
    #INFO: This is a sample for adding user data to document
    #---- Create user data object
    my $a2wUserDataTmp = new a2w::UserData();
    #---- Set document and page id as user data id
    $a2wUserDataTmp->setId( "PAGE" . "_" . $PageId . "_" . "INDEX" );
    #---- Set current page index as data
    $a2wUserDataTmp->setData( $sIndexDataTmp, length( $sIndexDataTmp ) );
    #---- Add data to document
    $a2wDocumentPar->addUserData( $a2wUserDataTmp );
    return $svRetTmp;
}

```

The routine `fetchPageIndexes()` fetches the indexing information of the corresponding page to be added in the user object.

addUserData.pm - fetchPageIndexes() - Adding UserData:

```
#-----
# Fetch Page Indexes
#-----
sub fetchPageIndexes(){
    if ( $bLog == $TRUE ){
        print "fetchPageIndexes()\n";
    }
    #---- Prepare Page Id
    my $sRetIndexDataTmp = "PageId=$PageId\r\n";
    #---- Fetch first Page Index
    my $a2wPageIndexTmp = $a2wPagePar->getFirstIndex();
    while ( $a2wPageIndexTmp != 0 ){
        #---- Add Index Record
        $sRetIndexDataTmp .= $a2wPageIndexTmp->getName() . "=" .
            $a2wPageIndexTmp->getValue() . "\r\n";
        #---- Fetch next Page Index
        $a2wPageIndexTmp = $a2wPagePar->getNextIndex();
    }
    return $sRetIndexDataTmp;
}
```

7.2.4.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

On Windows:

```
afp2web.exe -q -c -doc_cold -sp:addUserData.pm samples\insure.afp
```

On Unix:

```
./afp2web -q -c -doc_cold -sp:addUserData.pm samples/insure.afp
```

7.2.4.3 Accessing UserData in A2W Java SDK CALLBACK Handler

`A2WDocumentWriter` is a java interface class which you need to implement. An example of an implementation is the class `CustomDocumentWriter`. `CustomDocumentWriter` has a `write` method, which gets a user data list from the object "a2wDocObjectPar" of type `A2WDocumentObject`, loops through the user data list, and writes user data to separate files.

For more information, please refer to the Java SDK documentation that comes with A2W JDK release package.

```
public class CustomDocumentWriter implements A2WDocumentWriter{
    /**
     * Callback to write document with default definition
     * <p>
     * @param a2wDocObjectPar Document object that has output buffer, index
     * list and user data list
     * <br>
     * @return
     * <PRE>
     * 0 on Success
     * <0 on Error
     * </PRE>
     * </p>
     */
    public int write( A2WDocumentObject a2wDocObjectPar ){
```

```

//---- Assert parameters ----//
if ( a2wDocObjectPar == null ){
    return -1;
}
//---- Fetch user data list ----//
LinkedList lstDocUserDataTmp = a2wDocObjectPar.getUserDataList();
//---- Process UserData details ----//
if ( lstDocUserDataTmp != null ){
    .....
    String sUserDataBaseFilenameTmp = "userdata";
    String sUserDataFilenameTmp = null;
    try {
        A2WUserData a2wUserDataTmp = null;
        FileOutputStream fosUserDataTmp = null;
        ListIterator lstitrUserDataTmp=lstDocUserDataTmp.listIterator( 0 );
        if ( lstitrUserDataTmp != null ){
            while ( lstitrUserDataTmp.hasNext() ){
                a2wUserDataTmp =
(A2WUserData)lstitrUserDataTmp.next();

                // Write User Data to a File
                if ( a2wUserDataTmp != null
                    && a2wUserDataTmp.getData() != null ){
                    // Build filename to store User Data
                    sUserDataFilenameTmp =
sUserDataBaseFilenameTmp + "_" +

                    a2wUserDataTmp.getId() + ".dat";
                    // Create file stream for User data
                    fosUserDataTmp=new FileOutputStream(
sUserDataFilenameTmp );

                    fosUserDataTmp.write( a2wUserDataTmp.getData()

);

                    fosUserDataTmp.close();
                    fosUserDataTmp = null;
                } //if
            } // while
        } //if
    } //try
    catch ( Exception eFileErrorPar ){
        System.out.println( eFileErrorPar );
    } }

```

7.2.5 addWatermark.pm: Adding Watermark Text to Pages

This example shows how to add text to a page as watermark.

7.2.5.1 Understanding addWatermark.pm

In this example, we focus on the subroutine:

Subroutine	Description
sub afp2web()	Main entry for AFP2web.

	We show how <code>afp2web()</code> adds text as watermark.
--	--

The following commands are used to add watermark text.

addWatermark.pm - sub afp2web():

```
#---- Add Watermark ----#
#---- Create Helvetica 60 points Font (Watermark font)
$a2wFontTmp = new a2w::Font();
$a2wFontTmp->setName("Helvetica");
$a2wFontTmp->setHeight(60);
$a2wFontTmp->setBold($TRUE);
#---- Create Text Object ----#
my $a2wWatermarkTmp = new a2w::Text();
$a2wWatermarkTmp->setText("Confidential");
$a2wWatermarkTmp->setXPos(81);# X=81mm
$a2wWatermarkTmp->setYPos(204);# Y=204mm
$a2wWatermarkTmp->setAngle(60);# Rotation Angle = 60 degrees
$a2wWatermarkTmp->setColor(0xBDDBE7);# RRGGBB color value (BD=>189,
DB=>219, E7=>231 is light blue)
$a2wWatermarkTmp->setFont($a2wFontTmp);# set font
#---- Add Text to Page
$a2wPagePar->addText( $a2wWatermarkTmp );
```

7.2.5.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

```
On Windows:
afp2web.exe -q -c -doc_cold -sp:addWatermark.pm samples\insure.afp
On Unix:
./afp2web -q -c -doc_cold -sp:addWatermark.pm samples/insure.afp
```

7.2.5.3 Results of addWatermark.pm

This example produces PDF files with watermarks added to each page. Please note that the demo version of AFP2web adds its own "Demo Version" watermark to the result.

7.2.6 afp2xml.pm: Script Used to Produce XML Out of AFP

Producing XML from AFP requires two steps to solve the problem: The first step is to use the Scripting Facility to extract anything you want from the AFP spool file. The second step is to define your XML. Using a DTD, you define which XML elements and attributes are valid and using basic XML functionality of Perl, you produce your XML.

`afp2xml.pm` shows how to extract text and index data from AFP documents to create text files in XML format.

7.2.6.1 Understanding afp2xml.pm

This script requires that following packages be included.

afp2xml.pm - Packages to Include:

```

use a2w::Config;
use a2w::Document;
use a2w::Font;
use a2w::Index;
use a2w::Kernel;
use a2w::Line;
use a2w::MediumMap;
use a2w::NOP;
use a2w::Overlay;
use a2w::Page;
use a2w::PSEG;
use a2w::Text;
#---- Required Perl Modules for XML output
use IO::File;
use XML::Writer;

```

This script uses XML::Writer, a simple Perl module for writing XML documents, which you can obtain at the Perl Comprehensive Network (www.cpan.org). We place this module in a new subfolder of the Perl installation lib\XML\.

A typical sequence of methods used to create an XML document: `my $writer = new XML::Writer(); $writer->startTag('greeting', 'type' => 'simple'); $writer->characters("Hello, world!"); $writer->endTag('greeting'); $writer->end();`

Please refer to the documentation of the XML::Writer, which you can find on the official CPAN website. There, you will find more information on the following methods:

Function	Description
<code>\$xmlDocWriter->xmlDecl</code>	Add an XML declaration to the beginning of an XML document. The version will always be ``1.0".
<code>\$xmlDocWriter->doctype</code>	Add a DOCTYPE declaration to an XML document. The declaration must appear before the beginning of the root element.
<code>\$xmlDocWriter->comment</code>	Add a comment to an XML document.
<code>\$xmlDocWriter->startTag</code>	Add a start tag to an XML document. Any arguments after the element name are assumed to be name/value pairs for attributes: the module will escape all '&', '<', '>', and '' characters in the attribute values using the predefined XML entities
<code>\$xmlDocWriter->endTag</code>	Add an end tag to an XML document. The end tag must match the closest open start tag, and there must be a matching and properly-nested end tag for every start tag
<code>\$xmlDocWriter->dataElement</code>	Print an entire element containing only character data.
<code>\$xmlDocWriter->characters</code>	Add character data to an XML document. All '<', '>', and '&' characters in the \$data argument will automatically be escaped using the predefined XML entities
<code>\$xmlDocWriter->end</code>	Finish creating an XML document. This method will check that the document has exactly one document element, and that all start tags are closed

The basic subroutines which must be located in the script are:

- `sub afp2web()` - the main entry point for AFP2web
- `sub initialize()`
- `sub initializeDoc()`
- `sub initializePage()`

- sub finalizePage()
- sub finalizeDoc()
- sub finalize()

In this example, we focus on the subroutines:

Subroutine	Description
sub initializeDoc()	Creates the start tag for the Document element
sub afp2web()	Main entry for AFP2web. It creates start tags for PageList and Page and produces XML output for elements on this page.
sub finalizePage()	Creates the end tag for the current element Page
sub finalizeDoc()	Creates end tags for the elements PageList and Document

The routine initializeDoc() defines the root element for the document.

afp2xml.pm - initializeDoc() - Defining the Root Element of the XML Document:

```
#---- Generating FileHandle for XML Writer (needed since a2wDocumentPar-
>getSimpleFilename() not available from initializeDoc())
$xmlDocIdFilename = $svOutputFilePath . $DocIdTmp . ".xml";
$xmlDocOutput = new IO::File( ">" . $xmlDocIdFilename );
#---- Generating XML Writer Object
$xmlDocWriter = new XML::Writer(OUTPUT=> $xmlDocOutput,
NAMESPACES=> 1,
);
#---- Processing Instruction
$xmlDocWriter->xmlDecl("ISO-8859-1");
$xmlDocWriter->comment("Generated by $svScriptProc " . $Version . " , " .
localtime());
$xmlDocWriter->doctype('Document', "", 'afp2xml.dtd');
#---- Starting with root element
$xmlDocWriter->startTag('Document', 'Id'=> $DocIdTmp,# Doc Id element
'Name'=> $a2wDocumentPar->getName(),
'Spool'=> $svSpoolFilename,
'PID'=> $a2wDocumentPar->getPID(),
);
```

The routine afp2web() is called for each document page and creates start tags for PageList , Page and produces XML output for elements on this page.

afp2xml.pm - afp2web():

```
#---- Add Document Indexes to Index List
if ($PageId == 1){
    addIndexes( $a2wDocumentPar );
    $xmlDocWriter->startTag('PageList');# <PageList>
}
#---- Starting with Element Page
$xmlDocWriter->startTag('Page', 'Id'=> $PageId,# Page Id element
'Width'=> $a2wPagePar->getWidth(),
'Height'=> $a2wPagePar->getHeight(),
'Resolution'=> $a2wPagePar->getResolution())
```

```
);
#---- Add Page Applied Medium Map Info
addPageMediumMapInfo();
#---- Add Page Applied Medium Overlay List
addPageMediumOverlayList();
#---- Add Page Included Overlay List
addIncludedOverlayList( $a2wPagePar );
#---- Add Page Included PageSegment List
addIncludedPageSegmentList( $a2wPagePar );
#---- Add Page Indexes to IndexList of Page
addIndexes( $a2wPagePar );
#---- Add NOPS to NOPList of Page
addPageNOPs();
#---- Add Text Objects to TextObjectList of Page
addTextObjects( $a2wPagePar );
#---- Add Line Objects to LineObjectList of Page
addLineObjects( $a2wPagePar );
```

For how to use the XML::Writer to create additional XML elements, please refer to the subroutines in the script example.

7.2.6.2 Starting the Script Example

Note: To run this example, you need to install Perl. Please refer to the system prerequisites in the installation chapter of the AFP2web User Guide.

The following listing shows a simple method of adding directories to the Perl search path. Please note that these directories must be modified for your environment.

Important: This statement must come as the very first statement in the main module loaded by AFP2web (this is `afp2xml.pm`).

Adding directories to the Perl search path in the internal Perl array @INC:

```
BEGIN {
unshift(@INC, ('C:\Perl\lib', 'C:\Perl\site\lib', 'C:\temp'));
}
```

As an alternative, instead of modifying the Perl search path in the routine as described above, you can do this directly on the command line when invoking AFP2web:

On Windows:

```
FOR /F %i IN ( 'perl -e "${$,=';'; print @INC;}"' ) DO @SET PERLLIB=%i
afp2web.exe -q -c -doc_cold -sp:afp2xml.pm samples\insure.afp
```

On Unix:

```
export PERLLIB=`perl -e "${$,=' '; print @INC;}"`
./afp2web -q -c -doc_cold -sp:afp2xml.pm samples/insure.afp
```

7.2.6.3 Results of afp2xml.pm

One PDF file is created per AFP document and one XML file with document information.

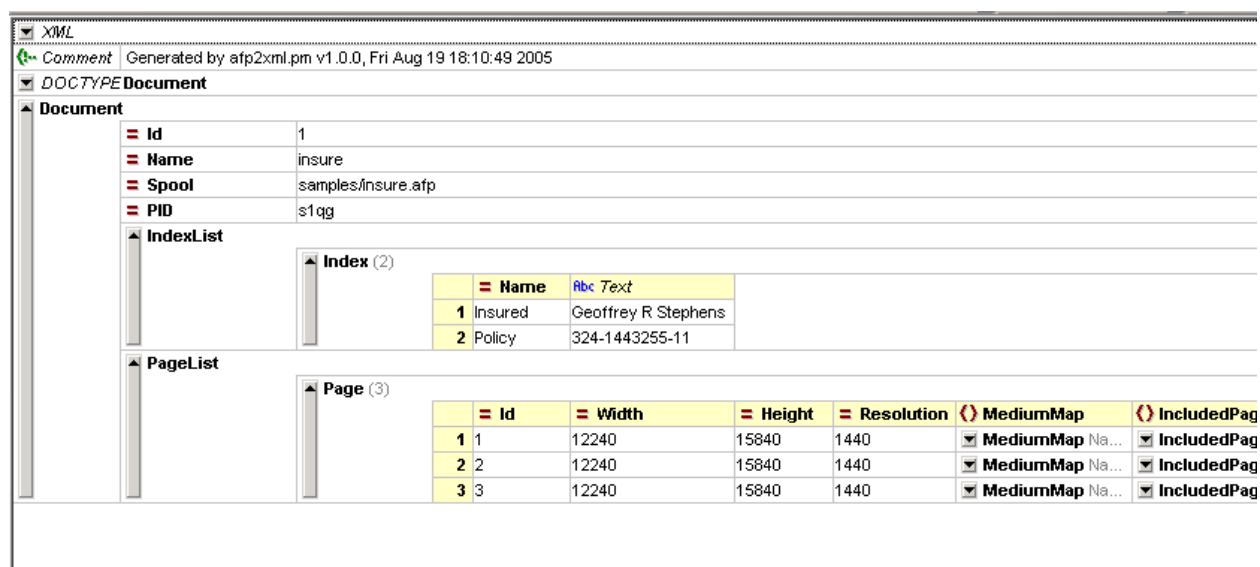
The XML representation of the AFP document will come as a text file without any formatting for readability:

```

0      10      20      30      40      50      60      70
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!-- Generated by afp2xml.pm v1.0.0, Fri Aug 19 18:10:49 2005 -->
3  <!DOCTYPE Document SYSTEM "mhtDoc_V3.0.dtd">
4  <Document Id="1" Name="insure" Spool="samples/insure.afp" PID="s1qg"><Index
.   " YPos="6176" CodedFont="" CharacterSet="C1DOGT10" CodePage="T1DOBASE" Type
.   trikeover="no">If You are not satisfied with Your Policy, You may cancel it
.   ents">Additional Benefits</Index><Index Name="Contents">Premium Summary</I
.   h="-1" Weight="Bold" Slant="Medium" Underline="no" Strikeover="no">.....
.   ace="GOTHIC" Height="100" Width="-1" Weight="Normal" Slant="Medium" Underl:
.   T1GIO361" Typeface="HELVETICA LATIN1" Height="100" Width="-1" Weight="Bold'
.   -1" Weight="Normal" Slant="Medium" Underline="no" Strikeover="no">--</Text(
.   h="-1" Weight="Bold" Slant="Medium" Underline="no" Strikeover="no">Part 1 .
.   Height="100" Width="-1" Weight="Normal" Slant="Medium" Underline="no" Stril
.   61" Typeface="TIMES NEW ROMAN LATIN1" Height="100" Width="-1" Weight="Bold'
.   100" Width="-1" Weight="Normal" Slant="Medium" Underline="no" Strikeover="
.   >2.</TextObject><TextObject XPos="4152" YPos="11407" CodedFont="" Character
5

```

If you want a formatted view, we recommend using an XML editor. For this purpose, we have placed DTDs for the document and index data in the sub folder /pdf:



7.2.7 any2html.pm: Module to build HTML output

The Script Facility example any2html.pm demonstrates how to build HTML output.

7.2.7.1 Understanding any2html.pm

The Scripting Facility sample any2html.pm helps to build HTML output using following idea,

AFP2web kernel will take care of rasterizing NON-TEXT contents using PNG output and all TEXT contents will be processed by Scripting Facility and will be stored as HTML output. AFP2web kernel generated PNG output will then be used as HTML background.

NOTE: Scripting Facility will remove the TEXT contents so that AFP2web kernel will present only NON-TEXT contents on PNG output

In addition to above processing, Scripting Facility will also format the input page content in case of LPD input.

In this example, we focus on the following subroutines:

Subroutine	Description
sub initializePage()	Declare the variable hash to collect page font table
sub afp2web()	Calls following sub routines, _buildHeaderMetaInfo() to build the HTML header meta information. _collectAndRemoveTextContent() to collect and remove text contents in page from presentation. And builds HTML header and font table style. _formatPage() is additionally invoked for LPD input to format text contents
sub _buildHeaderMetaInfo	Builds HTML header meta info.
sub _collectAndRemoveTextContent	Used to Collect and remove text content from page. Calls the subroutine, _isFontBold(), _isFontItalic(), and _fetchHTMLFont()
sub _isFontBold	Asserts whether font weight is BOLD and returns TRUE if it is
sub _isFontItalic	Asserts whether font slant is ITALIC and returns TRUE if it is
sub _fetchHTMLFont	Search for text font equivalent HTML font name and returns it

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document.

any2html.pm – afp2web() - Calling _formatPage:

```
....
#---- Format page, for LPD input ----#
if ( lc( $sSpoolType ) eq "lpd" ){
    my ( $iRcTmp, $sMsgTmp ) = _formatPage( $a2wPagePar );
    if ( $iRcTmp < 0 ){
        return ( $iRcTmp, $sMsgTmp );
    }
}
....
```

The subroutine _formatPage() is called when the input spool type is LPD to format page text contents.

any2html1.pm - _formatPage():

```
#-----
# Format page
```

```

#
# Parameter(s)
# 1. Page (of type a2w::Page)
#
# Returns
# >0 in case of success
# <0 (+ the reason message) in case of error
#
# Remarks
# Processes each text (aka line text) of input and formats it based on
# ASA carriage controls
#
# *** IMPORTANT NOTE ON PROCESSING LINES ***:
# AFP2web will process each line of input and will make a text object
# out of each line of text. So processing a text object mean processing
# a line from input.
#
# Also the text (aka line) will have Carriage Control character at the
# beginning which can be processed in below function for formatting the
# lines
#
#-----
sub _formatPage(){
    .....
    #---- Get parameter(s)
    #
    # 1. Page (of type a2w::Page)
    #
    my $a2wCurrentPageTmp = shift;

    #---- Fetch page resolution
    my $iPageResTmp = $a2wCurrentPageTmp->getResolution();

    #---- Set new page width/height
    $a2wCurrentPageTmp->setHeight( $iPageHeight );
    $a2wCurrentPageTmp->setWidth( $iPageWidth );

    #---- Evaluate line height
    my $fFactorTmp = $iPageResTmp / 72;

    if ( lc( $sScriptUnitBase ) eq "mm" ){
        $fFactorTmp = 25.4 / 72;
    }

    #---- Initialize LineId
    my $iLineIdTmp = 0;

    #---- Origin is at the bottom left corner
    my $iLeftMarginTmp = $iLeftMargin * $fFactorTmp;
    my $iTopMarginTmp = $iTopMargin * $fFactorTmp;

    #---- Modify boundary evaluation based on Top-Left co-ordinate system
    my $iXPosTmp = $iLeftMarginTmp;    # X Position starts from Left Margin
    my $iYPosTmp = $iTopMarginTmp;     # Y Position is "Top Margin"

    #---- Process and format page content ----#
    my $a2wTextTmp = $a2wCurrentPageTmp->getFirstText();

```

```

#---- Format each and every text (aka line text)
my $sTextTmp = "";
my $cCCCharTmp = "";
my $cTRCharTmp = "";
my $sTextWithoutCCTmp = "";

#---- Index Field Insured Name
my $sInsuredNameTmp = "";

while ( $a2wTextTmp != 0 ){
    $iLineIdTmp++;          # Increment line number
    $iXPosTmp = $iLeftMarginTmp; # X Position starts from Left Margin

    #---- Assert Y position
    #---- Modify boundary evaluation based on Top-Left co-ordinate system
    if ( $iYPosTmp > $iPageHeight ){
        if ( $bLog == $TRUE ){
            printf STDERR "\tERROR! Spool File Type ( " . $sSpoolFileType . "), Line=" .
$iLineIdTmp . ": Not enough space on page.\n";
        }

        return ( -98, "Spool File Type ( " . $sSpoolFileType . "), Line=" . $iLineIdTmp . ": Not
enough space on page." );
    }

    #---- Process current line and add texts appropriately
    $sTextWithoutCCTmp = "";
    $sTextTmp = $a2wTextTmp->getText();

    #---- Collect current page's index fields line
    if ( $iLineIdTmp == $iIndexFieldsLineNr ){
        $sIndexFieldsLine = $sTextTmp;
    }

    if ( $bTRCExist == $TRUE ){
        if ( $sTextTmp =~ /(.) (.*)/ ){
            $cCCCharTmp      = $1; # Contains CC
            $cTRCharTmp      = $2; # Contains TRC
            $sTextWithoutCCTmp = $3; # Contains line data
        }
    }
    else {
        if ( $sTextTmp =~ /(.) (.*)/ ){
            $cCCCharTmp      = $1; # Contains CC
            $sTextWithoutCCTmp = $2; # Contains line data
        }
    }

    #----- ASA Carriage Control action is applied before printing current line
    #       so process apply actions for given CC character
    #
    if ( $cCCCharTmp eq "+" ){
        # Do not space
        # Overwrite existing line
        $iYPosTmp -= $iLineSpacing;
    }
    elsif ( $cCCCharTmp eq " " ){
        # Space one line
        # One line space is already evaluated,
        # so nothing to do here
    }
}

```

```

}
elseif ( $cCCCharTmp eq "0" ){          # Space two lines
    # One line space is already evaluated,
    # so leave only one line space now
    $iYPosTmp += $iLineSpacing;
}
elseif ( $cCCCharTmp eq "-" ){          # Space three lines
    # One line space is already evaluated,
    # so leave only two line space now
    $iYPosTmp += $iLineSpacing;
    $iYPosTmp += $iLineSpacing;
}
elseif ( $cCCCharTmp eq "1" ){          # Skip to channel 1/New page
    # A2W kernel itself identifies new page,
    # so nothing to do here
}
elseif ( $cCCCharTmp eq "2" ){          # Skip to channel 2
}
elseif ( $cCCCharTmp eq "3" ){          # Skip to channel 3
}
elseif ( $cCCCharTmp eq "4" ){          # Skip to channel 4
}
elseif ( $cCCCharTmp eq "5" ){          # Skip to channel 5
}
elseif ( $cCCCharTmp eq "6" ){          # Skip to channel 6
}
elseif ( $cCCCharTmp eq "7" ){          # Skip to channel 7
}
elseif ( $cCCCharTmp eq "8" ){          # Skip to channel 8
}
elseif ( $cCCCharTmp eq "9" ){          # Skip to channel 9
}
elseif ( $cCCCharTmp eq "A" ){          # Skip to channel 10
}
elseif ( $cCCCharTmp eq "B" ){          # Skip to channel 11
}
elseif ( $cCCCharTmp eq "C" ){          # Skip to channel 12
}

#---- Skip trailing spaces
$sTextWithoutCCTmp =~ s/\s+$/g;

#---- Skip empty lines
if ( length( $sTextWithoutCCTmp ) <= 0 ){
    #---- Evaluate next YPos (YPos = YPos - FontHeight)
    $iYPosTmp += $iLineSpacing;

    #---- Don't present this text on output
    $a2wTextTmp->remove();
}
else {
    if ( $bLog == $TRUE ){
        printf STDERR ( "(" . $iXPosTmp . ", " . $iYPosTmp . ")=>" . $sTextWithoutCCTmp .
" <=\n" );
    }

    #---- Fill in text details
    $a2wTextTmp->setText( $sTextWithoutCCTmp ); # Text value

```

```

        $a2wTextTmp->setXPos( $iXPosTmp );          # Text X position
        $a2wTextTmp->setYPos( $iYPosTmp );          # Text Y position
        $a2wTextTmp->setFont( $a2wDefaultFont );    # Text font

        #---- Evaluate next YPos
        $iYPosTmp += $iLineSpacing;
    }

    #---- Get next text
    $a2wTextTmp = $a2wCurrentPageTmp->getNextText();
}
.....
return 0;
}

```

The processing logic in the routine "_formatPage()" is to process text (aka line text) of input and format it based on ASA carriage controls.

any2html.pm - afp2web() - Calling _buildHeaderMetaInfo:

```

....
#---- Build Header meta info
if ( $sHeaderMetaInfo eq "" ){
    $sHeaderMetaInfo = _buildHeaderMetaInfo();
}
....

```

The subroutine _buildHeaderMetaInfo() is called to build HTML header meta info.

any2html.pm - _buildHeaderMetaInfo:

```

#-----
# Build HTML header meta info
#
#-----
sub _buildHeaderMetaInfo{
    ....
    ....
    my $sMetaInfoTmp = '<META http-equiv="Content-Type" content="text/html; charset=Windows-1252">';
    return $sMetaInfoTmp;
}

```

This function used to build and return the HTML meta info.

any2html.pm - afp2web() - Calling _collectAndRemoveTextContent:

```

....
#---- Collect and remove text content
my $sPageContentTmp = _collectAndRemoveTextContent( $a2wPagePar, $fScaleFactorTmp );
....

```

The subroutine _collectAndRemoveTextContent() is called to collect and remove text contents from page.

any2html.pm - _collectAndRemoveTextContent:

```

#-----
# Collect and remove text content from page
#-----
sub _collectAndRemoveTextContent{
    ....
    #---- Fetch parameter(s)
    #
    # 1. Page (of type a2w::Page)

```

```

# 2. Scale factor (of type float)
#
my $a2wCurrentPagePar = shift;
my $fScaleFactorPar = shift;

#---- Loop through page text content and remove all
my $a2wTextTmp = $a2wCurrentPagePar->getFirstText();

my $fXPosTmp = 0.0;
my $fYPosTmp = 0.0;

my $a2wFontTmp = undef;
my $iFontLocalIdTmp = -1;

my $sFontStyleNameTmp = "";
my $sTextContentTmp = "";

my $iFontHeightTmp = 0;
my $bBoldFontTmp = $FALSE;
my $bItalicFontTmp = $FALSE;

while ( $a2wTextTmp != 0 ){
    #---- Fetch text position
    $fXPosTmp = $a2wTextTmp->getXPos() * $fScaleFactorPar;
    $fYPosTmp = $a2wTextTmp->getYPos() * $fScaleFactorPar;

    #---- Round text position
    $fXPosTmp = int( $fXPosTmp + .5 * ( $fXPosTmp <=> 0 ) );
    $fYPosTmp = int( $fYPosTmp + .5 * ( $fYPosTmp <=> 0 ) );

    #---- Fetch text font
    $a2wFontTmp = $a2wTextTmp->getFont();

    #---- Fetch font local id
    $iFontLocalIdTmp = $a2wTextTmp->getMappedFontLocalId();
    $sFontStyleNameTmp = "ft" . sprintf( "%03d", $iFontLocalIdTmp );

    #---- Evaluate font attributes
    $iFontHeightTmp = int($a2wFontTmp->getHeight() * $fFontScaleFactor);
    $bBoldFontTmp = _isFontBold( $a2wFontTmp );
    $bItalicFontTmp = _isFontItalic( $a2wFontTmp );

    # Check for leading/trailing spaces
    my $bPreFormattedTmp = $FALSE;
    my $sTextTmp = $a2wTextTmp->getText();
    if ( $sTextTmp =~ /\^s+/ || $sTextTmp =~ /\s+$/ ){
        $bPreFormattedTmp = $TRUE;
    }

    #---- Check whether font style is already created or not
    # If not created already, create it
    if ( $hrefPageFontTable->{ $iFontLocalIdTmp } ){ # Already created
        # Do nothing
    }
    else { # Not created already, create NOW
        #---- Build font class style
        $hrefPageFontTable->{ $iFontLocalIdTmp } = "."
            . $sFontStyleNameTmp

```

```

        . '{'
        . 'font-size:' . $iFontHeightTmp . 'px;'
        . 'font-family:' . _fetchHTMLFont(
$a2wFontTmp->getName() ) . ';'
        . '}'
        ;
    }
    #---- Add text to page content
    $sTextContentTmp .= "\n"
        . '<DIV style="position:absolute;'
        . 'left:' . $fXPosTmp . ';'
        . 'top:' . ( $fYPosTmp - $iFontHeightTmp ) . ';'
        . '"'
        . '>'
        . '<SPAN class=" . $sFontStyleNameTmp . "'
        . ' style="color:#' . sprintf( "%06X", $a2wTextTmp->getColor() ) . ';"'
        . '>'
        . ( ( $bBoldFontTmp == $TRUE ) ? '<B>' : '' )
        . ( ( $bItalicFontTmp == $TRUE ) ? '<I>' : '' )
        . ( ( $bPreFormattedTmp == $TRUE ) ? '<PRE>' : '' )
        . $sTextTmp
        . ( ( $bPreFormattedTmp == $TRUE ) ? '</PRE>' : '' )
        . ( ( $bItalicFontTmp == $TRUE ) ? '</I>' : '' )
        . ( ( $bBoldFontTmp == $TRUE ) ? '</B>' : '' )
        . '</SPAN>'
        . '</DIV>'
        ;

    #---- Remove text from presentation
    $a2wTextTmp->remove();

    #---- Fetch next text
    $a2wTextTmp = $a2wCurrentPagePar->getNextText();
}
return $sTextContentTmp;
}

```

The function used to collect and remove text content presentation from page.

any2html.pm : _isFontBold() called by _collectAndRemoveTextContent:

```

#-----
# Is Font Bold
#
# Checks whether given font has BOLD weight and returns true if yes
#
#-----
sub _isFontBold{
    ....
    #---- Fetch parameter(s)
    #
    # 1. Font (of type a2w::Font)
    #
    my $a2wFontPar = shift;

    my $bBoldTmp = $a2wFontPar->isBold();
    if ( $bBoldTmp == $FALSE ){
        #---- Fetch font character set name and decide whether it is bold or not
        my $sCharSetNameTmp = $a2wFontPar->getCharacterSetName();
    }
}

```

```

#---- List of bold character set names
my $hrefBoldCharSetNameTmp = {
    "c0p2ke60" => $TRUE
    , "c0p2ke80" => $TRUE
    , "c0p2ke90" => $TRUE
    , "c0p2ke00" => $TRUE
    , "c0p2kea0" => $TRUE
    , "c0p2keb0" => $TRUE
    , "c0p2ked0" => $TRUE
    , "c0p2kef0" => $TRUE
};

$bBoldTmp = $hrefBoldCharSetNameTmp->{ lc( $sCharSetNameTmp ) };
}
return $bBoldTmp;
}

```

The function is used to evaluate the given font has BOLD weight or not.

any2html.pm : _isFontItalic() called by _collectAndRemoveTextContent:

```

#-----
# Is Font Italic
#
# Checks whether given font has Italic slant and returns true if yes
#
#-----
sub _isFontItalic{
    ....
    #---- Fetch parameter(s)
    #
    # 1. Font (of type a2w::Font)
    #
    my $a2wFontPar = shift;

    my $bItalicTmp = $a2wFontPar->isItalic();
    if ( $bItalicTmp == $FALSE ){
        #---- Fetch font character set name and decide whether it is bold or not
        my $sCharSetNameTmp = $a2wFontPar->getCharacterSetName();

        #---- List of italic character set names
        my $hrefItalicCharSetNameTmp = {
            "c0p2k160" => $TRUE
            , "c0p2k180" => $TRUE
            , "c0p2k190" => $TRUE
            , "c0p2k100" => $TRUE
            , "c0p2k1a0" => $TRUE
            , "c0p2k1b0" => $TRUE
            , "c0p2k1d0" => $TRUE
            , "c0p2k1f0" => $TRUE
        };

        $bItalicTmp = $hrefItalicCharSetNameTmp->{ lc( $sCharSetNameTmp ) };
    }
    return $bItalicTmp;
}

```

The function is used to evaluate the given font has Italic slant or not.

any2html.pm : _fetchHTMLFont() called by _collectAndRemoveTextContent():

```

#-----
# Fetch equivalent HTML font name
#
# Search for text font equivalent HTML font name and returns it
#
#-----
sub _fetchHTMLFont{
    ....
    #---- Fetch parameter(s)
    #
    # 1. Text font name (of type string)
    #
    my $sTextFontNamePar = shift;

    my $sHTMLFontNameTmp = $sTextFontNamePar;
    if ( $sTextFontNamePar eq "FORMAT" ){
        $sHTMLFontNameTmp = "Arial";
    }
    return $sHTMLFontNameTmp;
}

```

The function used to search for text font equivalent HTML font name.

7.2.7.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

Windows:

```
afp2web.exe -q -c -doc_cold -sp:any2html.pm samples\insure.afp
```

Unix:

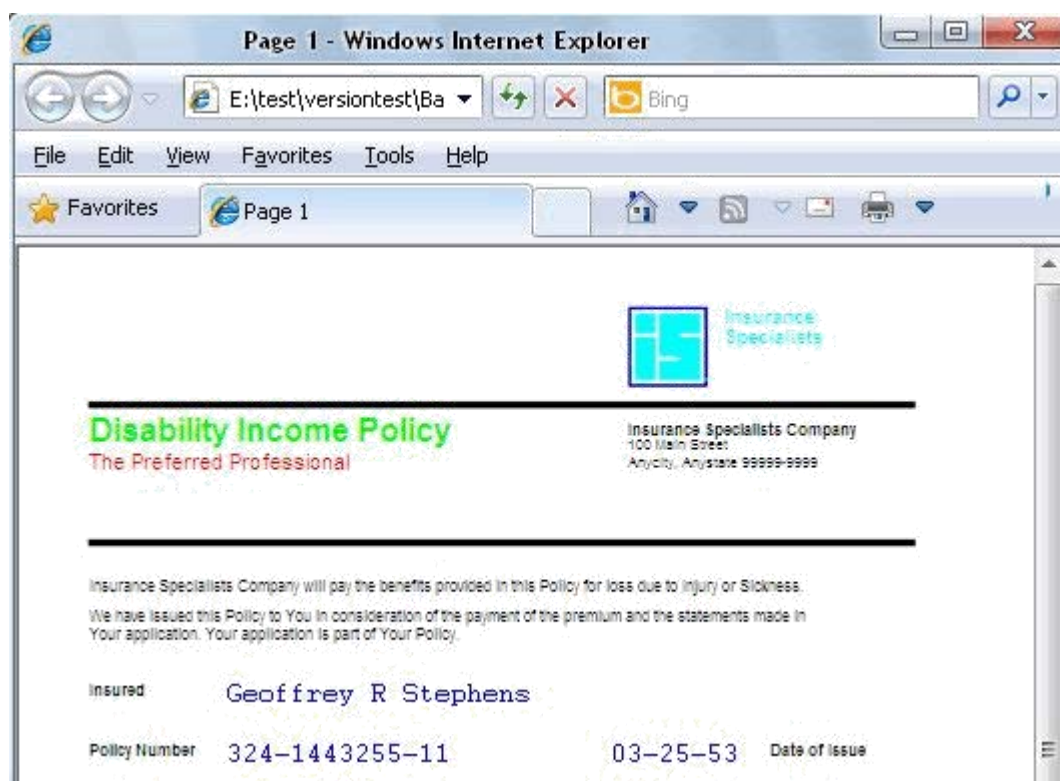
```
./afp2web -q -c -doc_cold -sp:any2html.pm samples/insure.afp
```

7.2.7.3 Results of any2html.pm

You will find the output files in the output sub folder (the default will be in the sub folder "pdf"):

Name	Ext	Size	Date	Attr
[.]	<DIR>		10/18/2010 19:18---	
insure_0001	html	7,036	10/18/2010 15:53-a--	
insure_0001	png	21,544	10/18/2010 15:53-a--	
insure_0002	html	11,239	10/18/2010 15:53-a--	
insure_0002	png	9,913	10/18/2010 15:53-a--	
insure_0003	html	11,707	10/18/2010 15:53-a--	
insure_0003	png	9,806	10/18/2010 15:53-a--	
insure_0004	html	7,030	10/18/2010 15:53-a--	
insure_0004	png	21,542	10/18/2010 15:53-a--	
insure_0005	html	11,233	10/18/2010 15:53-a--	
insure_0005	png	9,913	10/18/2010 15:53-a--	

The resulting HTML output will have a start page, which begins like this:



7.2.8 autosplit.pm: Splitting an AFP Spoolfile into Documents and Pages

AFP documents are normally merged in spool files to be directed to a printer. One AFP spool file can contain any number of documents and pages. Non-visible information in this spool file can be, for example, the index data defined by Tagged Logical Elements (TLEs) of AFP. TLEs are logically related to documents and document pages to which they apply and are thus a means for AFP2web to recognize document and page boundaries.

If your type of AFP spool file has TLEs included, you can let AFP2web handle document and page splitting automatically and still use a script to perform other tasks. All you have to do is set the AutoSplit property to ON and then AFP2web will automatically split an AFP spool file into documents and pages.

This Perl script autosplit.pm shows how to use the AutoSplit functionality of AFP2web.

7.2.8.1 Understanding autosplit.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub afp2web()	<p>Main entry for AFP2web.</p> <p>The return code in this routine tells AFP2web to begin a new document with this page or not, to skip the current page, or to stop AFP2web processing.</p>

sub initialize()	Accesses processing parameters, activates Autosplit, and initializes variables.
sub initializeDoc()	Saving document object, restart page count for this new document
initializePage()	Saving the current page object
addDocumentIndexes()	Collecting index data
addPageIndexes()	Collecting index data
sub finalizeDoc()	Output index data

The routine initialize() is called once at the beginning of the process. This routine is used to access processing parameters. For our example, we activate the AutoSplit functionality and initialize variables.

autosplit.pm - initialize() - Activating the autosplit functionality and initializing variables:

```
...
#---- Reset Page Id
$PageId = 0;
#---- Reset/Create Current Index List
@IndexList = ();
#---- Set AutoSplit to true
$a2wConfigPar->setAttribute( "AutoSplit", "on" );
...
```

AFP2web actively triggers the routine initializeDoc() when it starts processing a new document. Here too, we reset the page count to zero. Most important : we save a reference to the current document in the variable \$a2wDocumentPar. Using this reference, we will later access the document objects. Because we can access the document index data at this point, we call the subroutine addDocumentIndexes() to collect the index data.

autosplit.pm - initializeDoc() - Saving document object, restart page count for this new document:

```
sub initializeDoc(){
    #---- Get Parameter of initializeDoc( Par: a2w::Document )
    ($a2wDocumentPar) = @_ ;
    if ( $bLog == $TRUE ){
        print "initializeDoc(): DocId " . $a2wDocumentPar->getId() . "\n";
    }
    #---- Reset Page Id
    $PageId = 0;
    #---- Add Document Indexes to Index List
    addDocumentIndexes();
    return 0;
}
```

In the routine initializePage(), we save a reference to the current page in the variable \$a2wPagePar. Using this reference, we will later access page index data.

autosplit.pm - initializePage() - Saving the current page object:

```
sub initializePage(){
    #---- Get Parameter of initializePage( Par: a2w::Page )
    ($a2wPagePar) = @_ ;
    if ( $bLog == $TRUE ){
        print "initializePage()\n";
    }
}
```

```

        return 0;
    }

```

The routine `afp2web()` is the main entry point for AFP2web. This routine is called for each document page.

Because AFP2web handles document and page splitting, we don't have much programming logic for this part. For example, we don't have to implement rules for recognizing the start of a new document and we therefore don't need to set the return code to `$NEWDOC`.

For document splitting, all we do is set a default return value (append current page to document) and increment the page count for the current document.

Finally, for each page, we can access the index data of this page. This is done using the subroutine `addPageIndexes()`.

autosplit.pm - sub afp2web():

```

sub afp2web(){
    if ( $bLog == $TRUE ){
        print "afp2web(): PageId " . $a2wPagePar->getParseId() . "\n";
    }
    $APPEND= 0;# append page to Current Document
    $SKIP= 1;# skip page
    $NEWDOC= 2;# new document
    #---- Set default return value
    my $svRetTmp = $APPEND; # default: append page
    #---- Increment Page Id
    $PageId++;
    #---- Add Page Indexes to Index List
    addPageIndexes();
    return $svRetTmp;
}

```

The following lines show our custom routines, which use the functions of either the document and page object to collect index data.

autosplit.pm - addDocumentIndexes():

```

sub addDocumentIndexes(){
    if ( $bLog == $TRUE ){
        print "addDocumentIndexes()\n";
    }
    #---- Define temp variables
    my $IndexPtrTmp = @IndexList;
    #---- Fetch first Document Index
    my $a2wIndexTmp = $a2wDocumentPar->getFirstIndex();
    #---- Add PageGroup Name
    if ( $a2wIndexTmp != 0 ){
        #---- Add PageGroup Name to Index List
        @IndexList[$IndexPtrTmp++] = "PageGroup=" . $a2wIndexTmp->getIndexedObjectName();
    }
    #---- Loop thru Indexes
    while ( $a2wIndexTmp != 0 ){
        #---- Add Index Record to Index List
        @IndexList[$IndexPtrTmp++] = $a2wIndexTmp->getName() . "=" .
            $a2wIndexTmp->getValue();
        #---- Fetch next Document Index
        $a2wIndexTmp = $a2wDocumentPar->getNextIndex();
    }
}

```

autosplit.pm - addPageIndexes():

```

sub addPageIndexes(){
    if ( $bLog == $TRUE ){
        print "addPageIndexes()\n";
    }
    #---- Define temp variables
    my $IndexPtrTmp = @IndexList;
    #---- Add Page Id to Index List
    @IndexList[$IndexPtrTmp++] = "Page=" . $a2wPagePar->getName();
    #---- Fetch first Page Index
    my $a2wIndexTmp = $a2wPagePar->getFirstIndex();
    #---- Loop thru Indexes
    while ( $a2wIndexTmp != 0 ){
        #---- Add Index Record to Index List
        @IndexList[$IndexPtrTmp++] = $a2wIndexTmp->getName() . "=" .
            $a2wIndexTmp->getValue();
        #---- Fetch next Page Index
        $a2wIndexTmp = $a2wPagePar->getNextIndex();
    }
}

```

At the end of each document, that is, in the routine `finalizeDoc()`, we can print the index data collected for this document. The following listing shows a few lines of this routine. For the complete coding, please refer to the script example.

autosplit.pm - sub finalizeDoc() - output index data:

```

#---- Open Index file
my $svFileOpenSuccessTmp = open( fIndexFile, ">$IndexFilenameTmp" );
if ( $svFileOpenSuccessTmp ){
    #---- Write Document Id, Type, Name, PageCount and Size
    print fIndexFile ("DocId=". $a2wDocumentPar->getId() .
        ", DocType=". $svDocType .
        ", DocName=". $svOutputFilePath . $a2wDocumentPar->getOutputFilename() .
        ", PageCount=". $a2wDocumentPar->getPageCount() .
        ", Size=". $a2wDocumentPar->getSize() . "\n");
    #---- Write Indexes
    for(my $IndexPtrTmp = 0; $IndexPtrTmp < $docIndexCountTmp; $IndexPtrTmp++){
        print fIndexFile ("@IndexList[$IndexPtrTmp]\n");
    }
    print fIndexFile ("\n");
    close( fIndexFile );
}

```

autosplit.pm - sub finalizeDoc() - resetting the index collection for the next document:

```

#---- Reset Current Index List
@IndexList = ();

```

7.2.8.2 Running the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

On Windows:

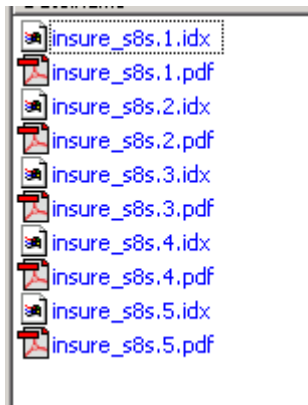
```
afp2web.exe -q -c -doc_cold -sp:autosplit.pm samples\insure.afp
```

On Unix:


```
./afp2web -q -c -doc_cold -sp:autosplit.pm samples/insure.afp
```

7.2.8.3 Results of autosplit.pm

This example produces PDF files for each document and a corresponding text file with index data:



A PDF document will have a start page, which begins like this:

		Insurance Specialists
Disability Income Policy The Preferred Professional		Insurance Specialists Company 100 Main Street Anycity, Anystate 99999-9999
Insurance Specialists Company will pay the benefits provided in this Policy for loss due to Injury or Sickness. We have issued this Policy to You in consideration of the payment of the premium and the statements made in Your application. Your application is part of Your Policy.		
Insured	Geoffrey R Stephens	
Policy Number	324-1443255-11	03-25-53 Date of Issue
NON-CANCELLABLE AND GUARANTEED CONTINUABLE TO AGE 65. NO CHANGE IN PREMIUM RATES.		

The contents of the text file with index data will look like the following.

autosplit.pm - contents of a text file with index data:

```
DocId=1, DocType=PDF, DocName=./pdf/insure_s8s.1.pdf, PageCount=3, Size=126321
PageGroup=324-1443255-110000001
Insured=Geoffrey R Stephens
Policy=324-1443255-11
Page=00000001
Contents=Disability Income Policy
Page=00000002
Contents=Policy Schedule
Contents=Policy Schedule
```

Contents=Table of Benefits
 Contents=Additional Benefits
 Contents=Premium Summary
 Page=00000003
 Contents=Definitions

For this example, we don't want to describe the structure of AFP elements in detail. If you want to investigate your AFP data, you might want to use the command option -ll to create an AFP2web log file.

The following screen shot shows part of the log file created when running the command option -ll:ALL for our example. We can find entries for index data, the first Index is located at the beginning of the document:

```
> samples/insure.afp

03 A8 A7 BDI Begin Document Index
  Fully Qualified Name Triplet
  Fully Qualified Name used (FQNTtype): 0x01
  This GID replaces the first parameter in the structured field that contains
  Fully Qualified Name Triplet
  Fully Qualified Name used (FQNTtype): 0x83
  The triplet contains a GID reference to a Begin Document structured field
  Fully Qualified Name Triplet
  Fully Qualified Name used (FQNTtype): 0x0a
  The triplet contains a GID reference to a Begin Resource Group structured
03 B2 A7 IEL Index Element
  Object Byte Extent Triplet
  Byte Extent of Indexed Object: 0x2ca1
  Direct Byte Offset Triplet
  Offset of the indexed object in bytes: 0x94
  Object Structured Field Extent Triplet
  Extent, in structured fields, to the indexed object: 0x62
  Object Structured Field Offset Triplet
  Offset, in structured fields, to the indexed object: 0x02
  Medium Map Page Number Triplet
  Sequence Number of The Indexed Page: 0x01
  Fully Qualified Name Triplet
  Fully Qualified Name used (FQNTtype): 0x0d
  The triplet contains a GID reference to a Begin Named Page Group structure
03 A0 90 TLE Tag Logical Element
  Sequence Number: 00000000
  Level Number: 00000000
  Fully Qualified Name Triplet
  Fully Qualified Name used (FQNTtype): 0x0b
  The triplet contains the GID of a document attribute: Insured
  Attribute Value Triplet
  Attribute Value: Geoffrey R Stephens
03 A0 90 TLE Tag Logical Element
  Sequence Number: 00000000
  Level Number: 00000000
  Fully Qualified Name Triplet
  Fully Qualified Name used (FQNTtype): 0x0b
  The triplet contains the GID of a document attribute: Policy
  Attribute Value Triplet
  Attribute Value: 324-1443255-11
03 B2 A7 IEL Index Element
```

7.2.9 dumpIniParms.pm: Dumping INI Parameters

The Script Facility example dumpIniParms.pm demonstrates how to access the AFP2web processing parameters. These parameters have the values, which are set in the configuration file (afp2web.ini file) or which have been overridden by a command line option when calling AFP2web.

7.2.9.1 Understanding dumpIniParms.pm

This script requires that following packages be included:

```
use a2w::Config;
use a2w::Kernel;
```

The subroutine initialize() is called at the beginning of the AFP2web process. At this point, we have access to the global parameters. We use the Perl variable @_ to save the arguments, which are passed to this subroutine.

```
#---- Get Parameter of initialize( Par: a2w::Config, a2w::Kernel )
( $a2wConfigPar, $a2wKernelPar ) = @_;
```

The next statement turns the module logging on if the corresponding INI parameter or command line option has been set.

```
#---- Set/Reset Logging
$bLog = $FALSE;
if (index( lc($a2wConfigPar->getAttribute("LoggingLevel")), "sf") >= 0 ){
    $bLog = $TRUE;
}
```

In this context, we can use the functions of a2w::Config and a2w::Kernel. For example, to access any processing parameter, use the getAttribute method of a2w::Config with the name of the INI parameter as argument. The following lines show how to use these functions:

```
my $svScriptProcTmp = $a2wConfigPar->getAttribute("ScriptProcedure");
my $svScriptArgsTmp = $a2wConfigPar->getScriptArgs();
$svIndexFilePath = $a2wConfigPar->getIndexFilePath();
$svOutputFilePath = $a2wConfigPar->getOutputFilePath();
$svSpoolFilename = $a2wKernelPar->getSpoolFilename();
```

The following lines prepare to write output to a separate dump file. The name of this file must have been passed as argument using the command line option -sa. If this command line option is not entered, the script returns an error code to AFP2web and AFP2web will stop processing.

```
#---- Open Dump file
my ($svSpoolFilenamePathTmp, $svDumpFilenameTmp) = ($svSpoolFilename =~ /^(?:(?:.*[:\\\/])?)(.*)/s);
$svDumpFilenameTmp = $svOutputFilePath . $svDumpFilenameTmp . ".txt";
open( fDumpFile, ">$svDumpFilenameTmp" );
print "Running $svScriptProcTmp: Dumping to $svDumpFilenameTmp...\n";
```

In the following code, we build an array containing all the AFP2web processing parameters. The for loop iterates through that array to get each parameter and then writes its value to the output file.

```
my $ptrTmp = 0;
my @iniParmList = ();
@iniParmList[$ptrTmp++] = "Licensee";
@iniParmList[$ptrTmp++] = "SerialNr";
@iniParmList[$ptrTmp++] = "Title";
@iniParmList[$ptrTmp++] = "Subject";
@iniParmList[$ptrTmp++] = "Keywords";
# ----- additional items not listed here....
$iniParmCountTmp = @iniParmList;
print fDumpFile ("===== Ini Parameter List =====\n");
for ( $ptrTmp = 0; $ptrTmp < $iniParmCountTmp; $ptrTmp++ ){
```

```

        print fDumpFile ( "\t" . @iniParmList[$ptrTmp] . ">" . $a2wConfigPar-
        >getAttribute(@iniParmList[$ptrTmp]) . "<=\n" );
    }
    print fDumpFile ("=====\n");
    close(fDumpFile);
    return 0;

```

There are no output documents resulting from this script. Because we return \$SKIP in the afp2web() subroutine, all pages are skipped:

```

sub afp2web(){
    if ( $bLog == $TRUE ){
        print "afp2web(): PageId " . $a2wPagePar->getParseId() . "\n";
    }
    $APPEND= 0;# append page to Current Document
    $SKIP= 1;# skip page
    $NEWDOC= 2;# new document
    $svRetTmp = $SKIP; # skip page
    return $svRetTmp;
}

```

7.2.9.2 Running the Script Example

You use the following command in demo.bat to start AFP2web with the Scripting Facility:

```

@ECHO OFF
:On Windows:afp2web.exe -q -c -doc_cold -sp:dumpIniParms.pm samples\
insure.afp
:On Unix:./afp2web -q -c -doc_cold -sp:dumpIniParms.pm samples/insure.afp

```

7.2.9.3 Results of dumpIniParms.pm

You will find the output file written by dumpIniParms.pm in the output sub folder (the default is the file dumpIniParms.txt in the sub folder /pdf):

```

===== Ini Parameter List =====
Licensee=>Maas Holding GmbH Evaluation Version<=
SerialNr=>E6E5E2A8-6DB7B128<=
Title=>AFP2web v4.0 [Built for Windows 2003/2008 32-bit on Jul 16 2010 at 16:35:20]<=
Subject=>AFP and TIFF Conversions (afp2web.com)<=
Keywords=>AFP LPD PDF TIFF<=
Logging=>off<=
LoggingFont=>off<=
LoggingLevel=>0<=
ExceptionLoggingLevel=>1<=
LogPath=>./log/<=
ResPath=>./samples/resource/<=
CpPath=>./afpcp/<=
CodePage=>T1V10273<=
OutputFilePath=>./pdf/<=
IndexPath=>./pdf/<=
..... additional lines not displayed here....

```

7.2.10 dumpMediumMaps.pm: Output List of Medium Maps to a Dump File

This Perl script dumpMediumMaps.pm shows how to create a dump file with a list of all medium maps applied to a document page.

7.2.10.1 Understanding dumpMediumMaps.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub afp2web()	Main entry for AFP2web. The return code in this routine tells AFP2web to begin a new document with this page or not, to skip the current page, or to stop AFP2web processing. We show how afp2web() accesses and prints information about the current medium map applied to the page.

The routine afp2web() is the main entry point for AFP2web. That is, it will be invoked at each parsing event, which can be trapped by the Scripting Facility. In our case, this routine will be called at the beginning of each document page.

The commands used to print information on the current medium map:

dumpMediumMaps.pm - afp2web():

```
#---- Get Page Id
my $svPageIdTmp = $a2wPagePar->getParseId();
#---- Dump
printf fDumpFile ("===== ");
printf fDumpFile ("Page %06d", $svPageIdTmp);
printf fDumpFile (" =====\n");
#---- Fetch applied medium map
my $a2wMediumMapTmp = $a2wPagePar->getAppliedMediumMap();
if ( $a2wMediumMapTmp != 0 ){
    #---- Dump Formdef Name
    printf fDumpFile ("Formdef: " . $a2wMediumMapTmp->getFormdefName() .
"\n");
    #---- Dump Medium Map Name
    printf fDumpFile ("Medium Map: " . $a2wMediumMapTmp->getName() . ", ");
    #---- Dump Medium Map Width, Height, Resolution, Duplex Control and N-Up
    Control
    printf fDumpFile ("Width=". $a2wMediumMapTmp->getWidth() .
", Height=". $a2wMediumMapTmp->getHeight() .
", Resolution=". $a2wMediumMapTmp->getResolution() .
", Duplex Control=" . $a2wMediumMapTmp->getDuplexControl() .
", N-Up Control=" . $a2wMediumMapTmp->getNupControl() .
"\n");
}
```

7.2.10.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

On Windows:


```
afp2web.exe -q -c -doc_cold -sp:dumpMediumMaps.pm samples\insure.afp
```

On Unix:

```
./afp2web -q -c -doc_cold -sp:dumpMediumMaps.pm samples/insure.afp
```

7.2.10.3 Results of dumpMediumMaps.pm

The resulting PDF document will have a start page, which begins like this:

			Insurance Specialists
<hr/>			
Disability Income Policy The Preferred Professional		Insurance Specialists Company 100 Main Street Anycity, Anystate 99999-9999	
<hr/>			
Insurance Specialists Company will pay the benefits provided in this Policy for loss due to Injury or Sickness. We have issued this Policy to You in consideration of the payment of the premium and the statements made in Your application. Your application is part of Your Policy.			
Insured	Geoffrey R Stephens		
Policy Number	324-1443255-11	03-25-53	Date of Issue
<hr/>			
NON-CANCELLABLE AND GUARANTEED CONTINUABLE TO AGE 65. NO CHANGE IN PREMIUM RATES.			

You will find the dump file in the output sub folder (the default is the sub folder /pdf). In our example, the same medium map is applied to all pages. The first lines of the dump file are as follows.

dumpMediumMaps.pm - contents of the dump file (extract):

```
===== Page 000001
=====
Formdef: F1DEMO
Medium Map: OGL, Width=0, Height=0, Resolution=240, Duplex Control=1, N-Up
Control=1
===== Page 000002
=====
Formdef: F1DEMO
Medium Map: OGL, Width=0, Height=0, Resolution=240, Duplex Control=1, N-Up
Control=1
===== Page 000003
=====
Formdef: F1DEMO
Medium Map: OGL, Width=0, Height=0, Resolution=240, Duplex Control=1, N-Up
Control=1
.... additional lines not shown here....
```

7.2.11 dumpNOPs.pm: Output the Contents of NOP Fields Into a Dump File

NOP fields is a common method of passing non-visible document information. Various applications use their conventions for handling NOP fields - for example for printer controls, for indexing, for processing control.

You might want to view the existing NOP-information in your AFP spool files.

This Perl script dumpPageTextObjs.pm shows how to produce a list of the values passed as NOP fields per page.

7.2.11.1 Understanding dumpNOPs.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub afp2web()	Main entry for AFP2web. The return code in this routine tells AFP2web to begin a new document with this page or not, to skip the current page, or to stop AFP2web processing. What we show is how afp2web() processes all NOP objects of the current page.

The routine afp2web() is the main entry point for AFP2web. That is, it will be invoked at each parsing event, which can be trapped by the Scripting Facility. In our case, this routine will be called at the beginning of each document page.

We show how afp2web() collects all NOP objects of the current page and then prints the NOP values to a dump file.

dumpNOPs.pm - afp2web() - ... printing NOP values to the dump file:

```
#---- Get Page Id
my $svPageIdTmp = $a2wPagePar->getParseId();
#---- Dump
printf fDumpFile ("===== ");
printf fDumpFile ("Page %06d", $svPageIdTmp);
printf fDumpFile (" =====\n");
#---- Fetch 1st NOP
my $a2wNOPTmp = $a2wPagePar->getFirstNOP();
if ( $a2wNOPTmp != 0 ){
    my $svNOPDataTmp = $a2wNOPTmp->getValue();
    print fDumpFile ("NOP=$svNOPDataTmp\n");
    #---- Fetch next NOP
    $a2wNOPTmp = $a2wPagePar->getNextNOP();
    #---- Loop for each and every NOP
    while ( $a2wNOPTmp != 0 ){
        $svNOPDataTmp = $a2wNOPTmp->getValue();
        #---- Write NOP to dump file
        print fDumpFile ("NOP=$svNOPDataTmp\n");
        #---- Fetch next NOP
        $a2wNOPTmp = $a2wPagePar->getNextNOP();
    }; # end-while
}
```

7.2.11.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

On Windows:


```
afp2web.exe -q -c -doc_cold -sp:dumpNOPs.pm samples\insure.afp
```

On Unix:

```
./afp2web -q -c -doc_cold -sp:dumpNOPs.pm samples/insure.afp
```

7.2.11.3 Results of dumpNOPs.pm

The resulting PDF document will have a start page, which begins like this:

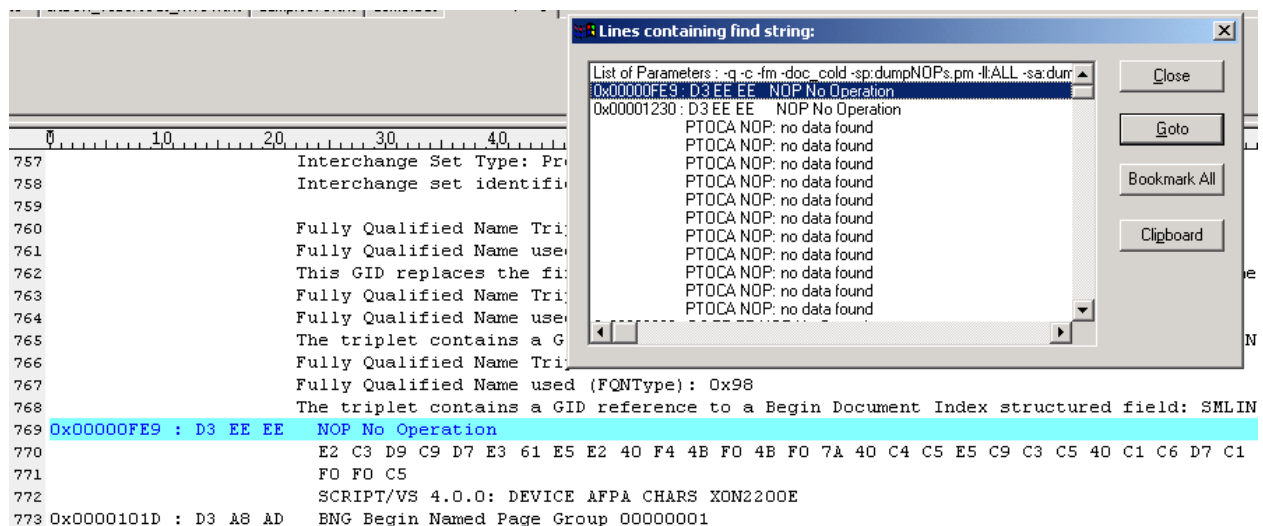
		Insurance Specialists
Disability Income Policy The Preferred Professional		Insurance Specialists Company 100 Main Street Anycity, Anystate 99999-9999
<p>Insurance Specialists Company will pay the benefits provided in this Policy for loss due to Injury or Sickness. We have issued this Policy to You in consideration of the payment of the premium and the statements made in Your application. Your application is part of Your Policy.</p>		
Insured	Geoffrey R Stephens	
Policy Number	324-1443255-11	03-25-53 Date of Issue
NON-CANCELLABLE AND GUARANTEED CONTINUABLE TO AGE 65. NO CHANGE IN PREMIUM RATES.		

You will find the dump file in the output subfolder (the default is the sub folder /pdf). Each line is preceded by the document page number. However, we find only one NOP object.

dumpNOPs.pm - contents of the dump file (extract):

```
===== Page 000001 =====
NOP=SCRIPT/VS 4.0.0: DEVICE AFPA CHARS X0N2200E
===== Page 000002 =====
===== Page 000003 =====
===== Page 000004 =====
===== Page 000005 =====
===== Page 000006 =====
===== Page 000007 =====
===== Page 000008 =====
===== Page 000009 =====
===== Page 000010 =====
===== Page 000011 =====
===== Page 000012 =====
===== Page 000013 =====
===== Page 000014 =====
===== Page 000015 =====
```

Using the command line option -ll:ALL, we create a log file. When looking into the log file created for this example, we see that there is actually only one NOP object:



7.2.12 dumpPageTextObjs.pm: Dumping Text and Font Information

The Perl script dumpPageTextObjs.pm shows how to access all text objects of a page and to write their text values to an output file (the dump file). Whenever a font information changes, this information will be written to the dump file immediately before the text affected by this change.

7.2.12.1 Understanding dumpPageTextObjs.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub initialize	Initializes and creates the dump file.
sub initializeDoc()	Saves the current document object
sub initializePage()	Saves the current document page
sub afp2web()	<p>Main entry for AFP2web.</p> <p>The return code in this routine tells AFP2web to begin a new document with this page or not, to skip the current page, or to stop AFP2web processing.</p> <p>In this routine, our example writes the text of the page to the dump file.</p> <p>Whenever a change in the font information occurs, this information is also written to the dump file.</p>

sub finalizeDoc()	Writes index data to file for this document.
-------------------	--

The subroutine initialize() includes commands to initialize and create the dump file:

dumpPageTextObjs.pm - initializeDoc():

```
sub initialize(){#-- Get Parameter of initialize( Par: a2w::Config,
    a2w::Kernel )
    ( $a2wConfigPar, $a2wKernelPar ) = @_;
    my $svScriptProcTmp = $a2wConfigPar->getAttribute("ScriptProcedure");
    my $svScriptArgsTmp = $a2wConfigPar->getScriptArgs();
    $svIndexFilePath = $a2wConfigPar->getIndexFilePath();
    $svOutputFilePath = $a2wConfigPar->getOutputFilePath();
    $svSpoolFilename = $a2wKernelPar->getSpoolFilename();
    #---- Open Dump file
    my ($svSpoolFilenamePathTmp, $svDumpFilenameTmp) = ($svSpoolFilename =~
    /^(?::.*[:\\\\/])?(.*)/s);
    $svDumpFilenameTmp = $svOutputFilePath . $svDumpFilenameTmp . ".txt";
    open( fDumpFile, ">$svDumpFilenameTmp" );
    print "Running $svScriptProcTmp: Dumping to $svDumpFilenameTmp...\n";
return 0;}
```

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document. We use the Perl variable @_ to save the document, which is passed as argument.

dumpPageTextObjs.pm - initializeDoc():

```
sub initializeDoc(){
    #---- Get Parameter of initializeDoc( Par: a2w::Document )
    ($a2wDocumentPar) = @_;
return 0;}
```

It is equally important that we save the current page object so that we can later access its elements. This is done in the subroutine initializePage().

dumpPageTextObjs.pm - initializePage():

```
sub initializePage(){#---- Get Parameter of initializePage(Par: a2w::Page)
($a2wPagePar) = @_;
return 0;}
```

The heart of the logic is located in the routine afp2web(). The first action is to initial the return code value, which will control AFP2web processing and print the the current page ID (number of the document in the spool file) to the dump file.

dumpPageTextObjs.pm - afp2web() - initializing the return code:

```
sub afp2web(){
    if ( $bLog == $TRUE ){
        print "afp2web(): PageId " . $a2wPagePar->getParseId() . "\n";
    }
    $APPEND= 0; # append page to Current Document
    $SKIP= 1; # skip page
    $NEWDOC= 2; # new document
    $svRetTmp = $APPEND; # default: append page
    #---- Get Page Id
    my $svPageIdTmp = $a2wPagePar->getParseId();
    #---- Dump
    printf fDumpFile ("===== ");
    printf fDumpFile ("Page %06d", $svPageIdTmp);
    printf fDumpFile (" =====\n");
    .... additional lines not shown here....
```

```

        return $svRetTmp;
    }

```

The routine `afp2web()` is called for each document page. We use this routine to loop thru all text objects that can be found on the page and to print the text found to the dump file.

dumpPageTextObjs.pm - afp2web() - writing all text to the dump file:

```

#---- Fetch first Text Object
my $a2wTextTmp = $a2wPagePar->getFirstText();
#---- Loop thru all the Texts
while ( $a2wTextTmp != 0 ){
    .... additional lines... for dumping font information .(see next listing)....
    #---- Build String to be dumped with details
    printf fDumpFile (" @(" . $a2wTextTmp->getXPos() . "," . $a2wTextTmp->getYPos() . ")>" . $a2wTextTmp->getText() . "<\n");
    #---- Get the next Text Object
    $a2wTextTmp = $a2wPagePar->getNextText();
} # end-while

```

While looping thru text objects we investigate the font information and whenever a change occurs, we write new font information to the dump file. We extend this loop with the following lines.

dumpPageTextObjs.pm - afp2web() - writing font information to the dump file:

```

#---- Write font entry with font details
$svTmp = $a2wTextTmp->getMappedFontLocalId();
if ( $svTmp != $svFontLocalIdTmp ){ # Compare Font Local Id against Current
    one
    #---- Save Font Local Id as Current
    $svFontLocalIdTmp = $svTmp;
    #---- Fetch Font Object
    $a2wFontTmp = $a2wTextTmp->getFont();
    #---- Build String to be dumped
    $svFontEntryTmp = "==">;
    #---- Fetch Coded Font name
    $svTmp = $a2wFontTmp->getCodedFontName();
    if ( $svTmp ne "" ){
        $svFontEntryTmp .= "CF=$svTmp, ";
    }
    #---- Fetch Character Set name
    $svTmp = $a2wFontTmp->getCharacterSetName();
    if ( $svTmp ne "" ){
        $svFontEntryTmp .= "CS=$svTmp, ";
    }
    .... additional font attributes ....
    #---- Dump Font Entry to File
    printf fDumpFile (" $svFontEntryTmp\n");
}

```

7.2.12.2 Running the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:


```

#On Windows:
afp2web.exe -q -c -doc_cold -sp:dumpPageTextObjs.pm samples\insure.afp
#On Unix:
./afp2web -q -c -doc_cold -sp:dumpPageTextObjs.pm samples/insure.afp

```

7.2.12.3 Results of dumpPageTextObjs.pm

The resulting PDF document will have a start page, which begins like this:

 Insurance Specialists	
Disability Income Policy The Preferred Professional	Insurance Specialists Company 100 Main Street Anycity, Anystate 99999-9999
<hr/>	
<p>Insurance Specialists Company will pay the benefits provided in this Policy for loss due to Injury or Sickness.</p> <p>We have issued this Policy to You in consideration of the payment of the premium and the statements made in Your application. Your application is part of Your Policy.</p>	
Insured	Geoffrey R Stephens
Policy Number	324-1443255-11
	03-25-53
	Date of Issue
<hr/>	
NON-CANCELLABLE AND GUARANTEED CONTINUABLE TO AGE 65. NO CHANGE IN PREMIUM RATES.	

You will find the dump file in the output sub folder (the default is the sub folder /pdf). The following example shows a few lines of the content you will find in the dump file.

dumpPageTextObjs.pm - contents of the dump file (extract):

```
===== Page 000001 =====
==>CS=C1H400B0, TF=HELVETICA LATIN1, W=B, S=12.00, CP=T1GI0361
@(8496,862)>Insurance<
@(8496,1120)>Specialists<
==>CS=C1H400J0, TF=HELVETICA LATIN1, W=B, S=20.00, CP=T1GI0361
@(792,2308)>Disability Income Policy<
==>CS=C1H200D0, TF=HELVETICA LATIN1, W=R, S=14.00, CP=T1GI0361
@(792,2638)>The Preferred Professional<
==>CS=C1H40000, TF=HELVETICA LATIN1, W=B, S=10.00, CP=T1GI0361
@(7330,2209)>Insurance Specialists Company<
==>CS=C1H20090, TF=HELVETICA LATIN1, W=R, S=9.00, CP=T1GI0361
@(7330,2418)>100 Main Street<
@(7330,2627)>Anycity, Anystate 99999-9999<
==>CS=C1H20000, TF=HELVETICA LATIN1, W=R, S=10.00, CP=T1GI0361
@(792,4104)>Insurance Specialists Company will pay the benefits provided
in this Policy for loss due to Injury or Sickness.<
@(792,4471)>We have issued this Policy to You in consideration of the payment
of the premium and the statements made in<
@(792,4681)>Your application. Your application is part of Your Policy.<
==>CS=C1H40000, TF=HELVETICA LATIN1, W=B, S=10.00, CP=T1GI0361
@(792,5353)>Insured<
```

7.2.13 eyecatcher_lpd.pm: Process and Format LPD text objects

The PERL script "eyecatcher_lpd.pm" shows how to process and format LPD input content. Along with this it also splits up documents based on eye catcher text and extracts required indexes.

AFP2web will make a text object out of each line input line data. So processing a text object mean processing a line from input. Following are the consequences of this processing behavior

- * Each text (aka line) will have the Carriage Control Character at the beginning (Script must use this character to format the text and must stripe this character from output presentation)
- * Each text (aka line) will be translated based on code page specified in "SpoolFileType" section, so the Carriage Control character will also be translated while processing it on Script)

eyecatcher_lpd.pm - Investigating the index field line properties:

Identify the line number of line having index fields using LPD viewers/editors. Set the identified line number into the variable "\$lIndexFieldsLineNr" on "initialize()" routine which will later be used on "_formatPage()" routine.

```
#---- Identify new document ----#
# New document is identified whenever the document id on index fields line is changing
#
# Syntax of index fields line:
# Index: 0      0      0      0...1      1      1      1      1
#         1      2      3      4...0      1      2      3      4
#         0123456789012345678901234567890...01234567890123456789012345678901234567890
#         >      0010000001      ...      324-1443255-11      03-25-53<
#
#         \_/_
#         |---> Document ID
#
```

Here the we used the "Document ID" index field to identify the new document based on the index field change we split the document.

7.2.13.1 Using eyecatcher_lpd.pm to split documents and retrieve index data

AFP2web Scripting Facility module used to process and format LPD input documents and to collect indexes.

In this example, we focus on the subroutines:

Subroutine	Description
sub initialize()	Initialize the initial values.
sub afp2web()	Formats the current page (by calling the sub routine "_formatPage()") and identifies the new document based on eye catcher text
sub _formatPage()	A custom routine that used to process each text (aka line text) of input and formats it based on ASA carriage controls
sub finalizeDoc()	Generates current document indexes (by calling subroutine "_processAndFetchIndexFields()") and writes the indexes on file
sub _processAndFetchIndexFields()	A custom routine that processes given text (aka line) for index fields, splits the found indexes and returns the same.

The subroutine used to declare and initialize page and index fields constants.

eyecatcher_lpd.pm - initialize():

```

sub initialize(){
    ....
    #---- Declare and Initialize Page formatting constants
    ....
    #---- Declare and Initialize index field line of document
    ....
}

```

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document.

eyecatcher_lpd.pm - initializeDoc():

```

sub initializeDoc(){
    #---- Get Parameter of initializeDoc( Par: a2w::Document )
    ($a2wDocumentPar) = @_ ;
    return 0;
}

```

It is equally important that we save the current page object so that we can later access its elements. This is done in the subroutine initializePage().

eyecatcher_lpd.pm - initializePage():

```

sub initializePage(){
    #---- Get Parameter of initializePage( Par: a2w::Page )
    ($a2wPagePar) = @_ ;

    if ( $bLog == $TRUE ){
        printf STDERR ( "initializePage()\n" );
    }

    #---- Initialize page index fields line
    $sIndexFieldsLine = "";

    return 0;
}

```

The routine afp2web() is used to process each document page. This routine formulates the return code value to be passed to AFP2web.

eyecatcher_lpd.pm - afp2web(): - initializing the return code:

```

sub afp2web(){
    #---- Set default return value
    my $iRetTmp = $APPEND; # append page to Current Document

    #---- Format page ----#
    my ( $iRcTmp, $sMsgTmp ) = _formatPage( $a2wPagePar );
    if ( $iRcTmp < 0 ){
        return ( $iRcTmp, $sMsgTmp );
    }

    .... additional lines with processing logic not shown here...

    return $iRetTmp;
}

```

The processing logic in the routine afp2web() handles page formatting (using "_formatPage()" routine) and splitting the spool file into documents. The rule is: The start page of a new document have set of index fields based on that value we set the return code for AFP2web to initiate processing for the end of the previous document and the begin of a new document.

eyecatcher_lpd.pm - _formatPage():

```

#-----
# Format page
#
# Parameter(s)
# 1. Page (of type a2w::Page)
#
# Returns
# >0 in case of success
# <0 (+ the reason message) in case of error
#
# Remarks
# Processes each text (aka line text) of input and formats it based on
# ASA carriage controls
#
# *** IMPORTANT NOTE ON PROCESSING LINES ***:
# AFP2web will process each line of input and will make a text object out of each line of text. So
# processing a text
# object mean processing a line from input.
#
# Also the text (aka line) will have Carriage Control character at the beginning which can be
# processed in below
# function for formatting the lines
#-----
sub _formatPage()
    ...
    #---- Get parameter(s)
    #
    # 1. Page (of type a2w::Page)
    #
    my $a2wCurrentPageTmp = shift;

    #---- Fetch page resolution
    my $iPageResTmp = $a2wCurrentPageTmp->getResolution();

    #---- Set new page width/height
    $a2wCurrentPageTmp->setHeight( $iPageHeight );
    $a2wCurrentPageTmp->setWidth( $iPageWidth );

    #---- Evaluate line height
    my $fFactorTmp = $iPageResTmp / 72;
    if ( lc( $sScriptUnitBase ) eq "mm" ){
        $fFactorTmp = 25.4 / 72;
    }

    #---- Initialize LineId
    my $iLineIdTmp = 0;

    #---- Origin is at the bottom left corner
    my $iLeftMarginTmp = $iLeftMargin * $fFactorTmp;
    my $iTopMarginTmp = $iTopMargin * $fFactorTmp;

    #---- Modify boundary evaluation based on Top-Left co-ordinate system
    my $iXPosTmp = $iLeftMarginTmp;    # X Position starts from Left Margin
    my $iYPosTmp = $iTopMarginTmp;     # Y Position is "Top Margin"

    #---- Process and format page content ----#
    my $a2wTextTmp = $a2wCurrentPageTmp->getFirstText();

```

```

#---- Format each and every text (aka line text)
my $sTextTmp = "";
my $cCCCharTmp = "";
my $cTRCharTmp = "";
my $sTextWithoutCCTmp = "";

#---- Index Field Insured Name
my $sInsuredNameTmp = "";

while ( $a2wTextTmp != 0 ){
    $iLineIdTmp++;          # Increment line number
    $iXPosTmp = $iLeftMarginTmp; # X Position starts from Left Margin

    #---- Assert Y position
    #---- Modify boundary evaluation based on Top-Left co-ordinate system
    if ( $iYPosTmp > $iPageHeight ){
        if ( $bLog == $TRUE ){
            printf STDERR "\tERROR! Spool File Type ( " . $sSpoolFileType . "), Line=" .
$iLineIdTmp . ": Not enough space on page.\n";
        }
        return ( -98, "Spool File Type ( " . $sSpoolFileType . "), Line=" . $iLineIdTmp . ": Not
enough space on page." );
    }

    #---- Process current line and add texts appropriately
    $sTextWithoutCCTmp = "";
    $sTextTmp = $a2wTextTmp->getText();

    #---- Collect current page's index fields line
    if ( $iLineIdTmp == $iIndexFieldsLineNr ){
        $sIndexFieldsLine = $sTextTmp;
    }

    if ( $bTRCExist == $TRUE ){
        if ( $sTextTmp =~ /(.) (.*)/ ){
            $cCCCharTmp      = $1; # Contains CC
            $cTRCharTmp      = $2; # Contains TRC
            $sTextWithoutCCTmp = $3; # Contains line data
        }
    }
    else {
        if ( $sTextTmp =~ /(.) (.*)/ ){
            $cCCCharTmp      = $1; # Contains CC
            $sTextWithoutCCTmp = $2; # Contains line data
        }
    }

    #----- ASA Carriage Control action is applied before printing current line
    #       so process apply actions for given CC character
    #
    if ( $cCCCharTmp eq "+" ){
        # Do not space
        # Overwrite existing line
        $iYPosTmp -= $iLineSpacing;
    }
    elseif ( $cCCCharTmp eq " " ){
        # Space one line
        # One line space is already evaluated,
        # so nothing to do here
    }
    elseif ( $cCCCharTmp eq "0" ){
        # Space two lines

```

```

        # One line space is already evaluated,
        # so leave only one line space now
        $iYPosTmp += $iLineSpacing;
    }
    elseif ( $cCCCharTmp eq "-" ){                # Space three lines
        # One line space is already evaluated,
        # so leave only two line space now
        $iYPosTmp += $iLineSpacing;
        $iYPosTmp += $iLineSpacing;
    }
    elseif ( $cCCCharTmp eq "1" ){                # Skip to channel 1/New page
        # A2W kernel itself identifies new page,
        # so nothing to do here
    }
    elseif ( $cCCCharTmp eq "2" ){                # Skip to channel 2
    }
    elseif ( $cCCCharTmp eq "3" ){                # Skip to channel 3
    }
    elseif ( $cCCCharTmp eq "4" ){                # Skip to channel 4
    }
    elseif ( $cCCCharTmp eq "5" ){                # Skip to channel 5
    }
    elseif ( $cCCCharTmp eq "6" ){                # Skip to channel 6
    }
    elseif ( $cCCCharTmp eq "7" ){                # Skip to channel 7
    }
    elseif ( $cCCCharTmp eq "8" ){                # Skip to channel 8
    }
    elseif ( $cCCCharTmp eq "9" ){                # Skip to channel 9
    }
    elseif ( $cCCCharTmp eq "A" ){                # Skip to channel 10
    }
    elseif ( $cCCCharTmp eq "B" ){                # Skip to channel 11
    }
    elseif ( $cCCCharTmp eq "C" ){                # Skip to channel 12
    }

    #---- Skip trailing spaces
    $sTextWithoutCCTmp =~ s/\s+$/g;

    #---- Skip empty lines
    if ( length( $sTextWithoutCCTmp ) <= 0 ){
        #---- Evaluate next YPos (YPos = YPos - FontHeight)
        $iYPosTmp += $iLineSpacing;
        #---- Don't present this text on output
        $a2wTextTmp->remove();
    }
    else {
        if ( $bLog == $TRUE ){
            printf STDERR ( "(" . $iXPosTmp . ", " . $iYPosTmp . ")=>" . $sTextWithoutCCTmp .
"<=\\n" );
        }
        #---- Get next text
        $a2wTextTmp = $a2wCurrentPageTmp->getNextText();
    }
    ...
    return 0;
}

```

The processing logic in the routine "_formatPage()" is to process text (aka line text) of input and format it based on ASA carriage controls.

eyecatcher_lpd.pm afp2web() - Using an eye catcher for document splitting:

```

.....
#---- Identify new document ----#
# New document is identified whenever the document id on index fields line is changing
#
# Syntax of index fields line:
# Index:
# 0      0      0      0...1      1      1      1      1
# 1      2      3      4...0      1      2      3      4
# 0123456789012345678901234567890...0123456789012345678901234567890
# >      0010000001      ...      324-1443255-11      03-25-53<
#      \ /
#      |---> Document ID
my $sDocIdTmp = "";
if ( $sIndexFieldsLine =~ /^.{18}(.{3}).*/ ){
    $sDocIdTmp = $1;    # $1 will have document id
}

if ( $sCurrentDocumentId ne $sDocIdTmp ){
    #---- Store current document id
    $sCurrentDocumentId = $sDocIdTmp;

    #---- Reset Page Id
    $iPageId = 0;

    $iRetTmp = $NEWDOC;
    if ( $bLog == $TRUE ){
        printf STDERR "Found New Document\n";
    }

    #---- Store index fields of current document
    %hshlstDocIndexFieldsLine->{ $a2wPagePar->getParseId() } = $sIndexFieldsLine;
}

#---- Increment Page Id
$iPageId++;
.....

```

When AFP2web closes a document, it will invoke the subroutine finalizeDoc(). Here, we fetch and output the index data from the document. This simple routine calls the subroutine "_processAndFetchIndexFields()", which is responsible for fetching the index data. You have full control of the document content even during the event of finalizing the document. To get the start page of the document, just use the method "\$a2wDocumentPar->getFirstPage()" and scan that page for your index data.

eyecatcher_lpd.pm - finalizeDoc()

```

sub finalizeDoc(){
    .....
    #---- Process and fetch document indexes ----#
    my $a2wFirstPageTmp = $a2wDocumentPar->getFirstPage();

    #---- Fetch page id
    my $iPageIdTmp = $a2wFirstPageTmp->getParseId();

    #---- Fetch current document index fields
    my $sIndexFieldsTmp = %hshlstDocIndexFieldsLine->{ $iPageIdTmp };
}

```

```

#---- Process index fields
my $hshIndexesTmp = _processAndFetchIndexFields( $sIndexFieldsTmp );

#---- Reset current document index fields
%hshlstDocIndexFieldsLine->{ $iPageIdTmp } = "";

#---- Assert processed index value
if ( $hshIndexesTmp == undef ){
    return ( -999, "Unable to fetch indexes for document " . $a2wDocumentPar->getId() );
}

#---- Write index to file ----#
#---- Build Index Filename
my $sSimpleFilenameTmp = $a2wDocumentPar->getSimpleFilename(); # get document simple filename
if ( $sSimpleFilenameTmp eq "" ){
    return ( -997, "Invalid Simple Filename(" . $sSimpleFilenameTmp . ") for document " .
$a2wDocumentPar->getId() );
}

my $IndexFilenameTmp = $sIndexFilePath . $sSimpleFilenameTmp . ".idx"; # build Index Filename
if ( $blog == $TRUE ){
    printf STDERR ( "Writing Index File ==> $IndexFilenameTmp\n" );
}

#---- Open Index file
my $bFileOpenSuccessTmp = open( fIndexFile, ">$IndexFilenameTmp" );
if ( !$bFileOpenSuccessTmp ){
    return ( -998, "Unable to open index file $IndexFilenameTmp, rc=" . $bFileOpenSuccessTmp .
"msg=" . $! );
}

#---- Fetch index name list
my @arrIndexNameListTmp = sort keys( %{ $hshIndexesTmp } );
for ( my $i = 0; $i < @arrIndexNameListTmp; $i++ ){
    print fIndexFile ( $arrIndexNameListTmp[ $i ] . "=" . $hshIndexesTmp->{
$arrIndexNameListTmp[ $i ] } . "\n" );
}
close( fIndexFile );

return 0;
}

```

This routine used to write index data to file.

eyecatcher_lpd.pm : processAndFetchIndexFields() called by finalizeDoc():

```

#-----
# Process index fields
#
# Parameter(s)
# 1. String having index fields
#
# Returns
# - undef in case of error
#
# - Hash reference having indexes in case of success, hash elements would
# be as given below
# %hshCurrentDocIndexes = {
#     'I_DOC_ID'      => ""          # Current document id (e.g. 001)
#

```

[illegible]

You use the following command to start AFP2web conversion for LPD input with the Scripting Facility:

On Windows:

```
afp2web.exe -q -c -lpdin -doc_cold -sp:eyecatcher_lpd.pm -sft:ASA_EBCDIC_FIXED_148
samples\lpdsample.lpd
```

On Unix :

```
./afp2web -q -c -lpdin -doc_cold -sp:eyecatcher_lpd.pm -sft:ASA_EBCDIC_FIXED_148
samples/lpdsample.lpd
```

7.2.13.3 Results of eyecatcher_lpd.pm

You will find the output files in the output sub folder (the default will be in the sub folder "pdf"):

e:\A2W_4.x\TestEnvironment\pdf*.*			
↑ Name	Ext	Size	Date
↑ [..]		<DIR>	07/31/2010 13:26
lpdsample_sgs.1	idx	97	07/31/2010 13:24
lpdsample_sgs.1	pdf	8,635	07/31/2010 13:24
lpdsample_sgs.2	idx	91	07/31/2010 13:24
lpdsample_sgs.2	pdf	8,636	07/31/2010 13:24
lpdsample_sgs.3	idx	94	07/31/2010 13:24
lpdsample_sgs.3	pdf	8,638	07/31/2010 13:24

The contents of the index file:

eyecatcher_lpd.pm - contents of an index file:

```
I_DOC_ID=001
I_INSURED=Geoffrey R Stephens
I_ISSUED_DATE=03-25-53
I_POLICY_NR=324-1443255-11
```

The resulting PDF document will have a start page, which begins like this:

```

Disability Income Policy, The Preferred Professional
                                                                                               *****
                                                                                               x   F R O N T P A G E   x
                                                                                               *****

*****
x Insured :                               x Policy Number :                               x Insurance Specialists Company       x
x Geoffrey R Stephens                     x 324-1443255-11                               x 100 Main Street                   x
x                                                                                               x Any City, Any State 99999-9999    x
*****
x                                                                                               x
x                                                                                               x Insurance Specialists Company will pay the benefits x
x                                                                                               x provided in this Policy for loss due to Injury or Sickness. x
x                                                                                               x
x                                                                                               x We have issued this Policy to You in consideration of the x
x                                                                                               x payment of the premium and statements made in Your        x
x                                                                                               x application. Your application is part of Your Policy.      x
x                                                                                               x
x                                                                                               x *****
x                                                                                               x NON-CANCELLABLE AND GUARANTEED CONTINUABLE TO AGE 65. NO  x
x                                                                                               x CHANGE IN PREMIUM RATES.                                  x
x                                                                                               x As long as the premium is paid on time, We cannot change  x
x                                                                                               x Your Policy or its premium rate until the first premium due x
x                                                                                               x date after Your 65th birthday.                             x
x                                                                                               x
x                                                                                               x *****
x                                                                                               x RENEWAL OPTIONS AFTER YOU REACH AGE 65. SUBJECT TO CHANGE x
x                                                                                               x IN PREMIUM RATES.                                          x
x                                                                                               x From age 65 to age 72, You may continue Your Policy for a x
x                                                                                               x Total Disability benefit with a limited benefit period     x
x                                                                                               x while You are actively and regularly employed full time.  x
x                                                                                               x This option is explained in PART 5.                         x
x                                                                                               x
x                                                                                               x When You are no longer actively and regularly employed    x
x                                                                                               x after age 65 or when You reach age 72, You may continue  x
x                                                                                               x Your Policy for the rest of Your life. The benefit will be x
x                                                                                               x limited to a Hospital Confinement Indemnity. This benefit  x
x                                                                                               x will take the place of all other benefits under the Policy. x
x                                                                                               x This option is explained in PART 6                          x
x                                                                                               x
x                                                                                               x *****
x                                                                                               x YOUR RIGHT TO CANCEL.                                       x
x                                                                                               x If You are not satisfied with Your Policy, You may cancel x
x                                                                                               x it. Return the Policy to Us or Our agent by midnight of the x
x                                                                                               x tenth day after the date You receive it. If You return the x
x                                                                                               x Policy by mail, it must be properly addressed, postage    x
x                                                                                               x prepaid, and postmarked no later than midnight of that    x
x                                                                                               x tenth day. Our mailing address is 100 Main Street, Anycity, x
x                                                                                               x Anystate 99999-9999. Within ten days after We receive the x
x                                                                                               x Policy, We will refund any premium You have paid. The     x
x                                                                                               x Policy will be considered to have never been issued.      x
x                                                                                               x
x                                                                                               x *****
x                                                                                               x READ YOUR POLICY CAREFULLY.                                 x
x                                                                                               x It is a legal contract between You and Us Signed for     x
x                                                                                               x Insurance Specialists Company.                              x
x                                                                                               x
x                                                                                               x
x                                                                                               x
*****
0010000001 Geoffrey R Stephens                                     324-1443255-11    03-25-53

```

7.2.14 eyecatcher_mmd.pm: Find and Extract MMD Text Objects

The PERL script `eyecatcher_mmd.pm` shows how to find and extract predefined text strings on a page. Each text string is located within a given y/x coordinate range.

- A particular text string indicates the first page in a mmd document.

7.2.14.1 Using `eyecatcher_mmd.pm` to find text coordinates

First, we want to determine the page position of these text elements. We run the AFP2web Scripting Facility with the following commands in the subroutine `afp2web()`.

eyecatcher_mmd.pm - Investigating the text positioning properties:

```

sub afp2web(){
    #---- Set default return value
    my $iRetTmp = $APPEND; # default: append page

    #---- Fetch first Text Object

```


sub _processAndFetchIndexFields()	A custom routine that processes given index fields line, splits indexes and returns the same.
--------------------------------------	---

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document.

eyecatcher_mmd.pm - initializeDoc():

```
sub initializeDoc(){
    #---- Get Parameter of initializeDoc( Par: a2w::Document )
    ($a2wDocumentPar) = @_;
    return 0;
}
```

It is equally important that we save the current page object so that we can later access its elements. This is done in the subroutine initializePage().

eyecatcher_mmd.pm - initializePage():

```
sub initializePage(){
    #---- Get Parameter of initializePage( Par: a2w::Page )
    ($a2wPagePar) = @_;
    return 0;
}
```

The routine afp2web() is used to process each document page. This routine formulates the return code value to be passed to AFP2web.

eyecatcher_mmd.pm - afp2web() - initializing the return code:

```
#-----
# Main entry method
# Return values:
# < 0:error
# 0:append page to Current Document
# 1:skip page
# 2:first page / new document
#-----
sub afp2web(){
    ....
    #---- Set default return value
    my $iRetTmp = $APPEND; # default: append page

    .... additional lines are with processing logic not shown here...

    return $iRetTmp;
}
```

The processing logic in the routine afp2web() handles splitting the spool file into documents. The rule is: The start page of a new document always has the text "001 von 001" on a fixed position on that page. If this text value is found, we set the return code for AFP2web to initiate processing for the end of the previous document and the begin of a new document.

eyecatcher_mmd.pm afp2web() - Using an eyecatcher_mmd for document splitting:

```
.....
#---- Identify new document ----#
# Rule:
# "001 von" in following text
#
# @(181,385)>      AAAAAAAAAAAAAA      001 von 001      <
#
#---- Fetch first text
```

```

my $a2wTextTmp = $a2wPagePar->getFirstText();

#---- Loop through all texts
while ( $a2wTextTmp != 0 ){
    #---- Assert whether document eyecatcher exists
    if ( $a2wTextTmp->getXPos() >= 176
        && $a2wTextTmp->getXPos() <= 186
        && $a2wTextTmp->getYPos() >= 380
        && $a2wTextTmp->getYPos() <= 390
        && ((substr ($a2wTextTmp->getText(), 75, 3) eq "001" )) # Still identify substring
        "001" from the "001 von 001" string
    ){
        #---- Reset Page Id
        $iPageId = 0;

        $iRetTmp = $NEWDOK;
        last; # leave while loop
    }

    #---- Fetch next Text Object
    $a2wTextTmp = $a2wPagePar->getNextText();
}

#---- Increment Page Id
$iPageId++;
.....

```

When AFP2web closes a document, it will invoke the subroutine "finalizeDoc()". Here, we fetch and output the index data from the document. This simple routine calls the subroutine "_processAndFetchIndexFields()", which is responsible for fetching the index data. You have full control of the document content even during the event of finalizing the document. To get the start page of the document, just use the method "\$a2wDocumentPar->getFirstPage()" and pass it to the subroutine then scan that page for your index data.

eyecatcher_mmd.pm - _processAndFetchIndexFields() called by finalizeDoc():

```

sub _processAndFetchIndexFields(){
    #---- Get parameter(s)
    #
    # Page (of type a2w::Page)
    #
    my $a2wFirstPagePar = shift;

    #---- Return value
    my $hshIndexFieldsTmp = undef;

    #---- Assert parameter(s)
    if ( $a2wFirstPagePar == 0 ){
        return $hshIndexFieldsTmp;
    }

    #---- Define the list of Eyecatcher positions we should look for
    #
    # @(181,265)>      Stuttgart                                     <
    # @(181,295)>      70173-70794                                   <
    # @(181,325)>      Geoffrey R Stephens                          13.03.2009    <
    #
    my @arrEyeCatcherIndexListTmp = ( ( 181,265 ), # Branch (Filiale)
                                       ( 181,295 ), # Bank code number (Bankleitzahl)
                                       ( 181,325 )  # Account holder name & Date of statement

```

```

    );

#---- Define temp variables
my $arrECILLenTmp = @arrEyeCatcherIndexListTmp / 2;
my $sTextTmp      = "";
my $iTextXPosTmp  = 0;
my $iTextYPosTmp  = 0;
my $iArrPosTmp    = 0;
my $iIdxTextXTmp  = 0;
my $iIdxTextYTmp  = 0;
my $sIndexValueTmp = "";

#---- Get the Index list count
my $iIndexPtrTmp = @arrIndexList;

#---- Fetch first text
my $a2wTextTmp = $a2wFirstPagePar->getFirstText();

#---- Loop thru all the Text Objects
while ( $a2wTextTmp != 0 ){
    #---- Fetch text info
    $sTextTmp      = $a2wTextTmp->getText();
    $iTextXPosTmp  = $a2wTextTmp->getXPos();
    $iTextYPosTmp  = $a2wTextTmp->getYPos();

    #---- Search all the defined Eyecatchers
    for (my $i = 0; $i < $arrECILLenTmp; $i++){
        $iArrPosTmp = $i * 2;
        $iIdxTextXTmp = $arrEyeCatcherIndexListTmp[ $iArrPosTmp ];
        $iIdxTextYTmp = $arrEyeCatcherIndexListTmp[ $iArrPosTmp + 1 ];

        #---- If true ==> we found one of the defined Eyecatchers
        if (    $iTextXPosTmp >= ( $iIdxTextXTmp - 10 )
            && $iTextXPosTmp <= ( $iIdxTextXTmp + 10 )
            && $iTextYPosTmp >= ( $iIdxTextYTmp - 10 )
            && $iTextYPosTmp <= ( $iIdxTextYTmp + 10 )
        ){
            #---- Fetch indexes
            if ( $i == 0 ){
                $sTextTmp =~ s/^\s+//g;    # strip leading spaces
                $sTextTmp =~ s/\s+$//g;    # strip trailing spaces

                if ( $hshIndexFieldsTmp == undef ){
                    $hshIndexFieldsTmp = {};
                }
                $hshIndexFieldsTmp->{ 'I_BRANCH' } = $sTextTmp;
            } elseif ( $i == 1 ){
                $sTextTmp =~ s/^\s+//g;    # strip leading spaces
                $sTextTmp =~ s/\s+$//g;    # strip trailing spaces

                if ( $hshIndexFieldsTmp == undef ){
                    $hshIndexFieldsTmp = {};
                }
                $hshIndexFieldsTmp->{ 'I_BRANCH_CODE' } = $sTextTmp;
            } elseif ( $i == 2 ){
                $sTextTmp =~ s/^\s+//g;    # strip leading spaces
                $sTextTmp =~ s/\s+$//g;    # strip trailing spaces
            }
        }
    }
}

```

```

        if ( $sTextTmp =~ /(.{30}).{27}(.{10})/ ){
            if ( $hshIndexFieldsTmp == undef ){
                $hshIndexFieldsTmp = {};
            }
            $hshIndexFieldsTmp->{ 'I_AC_NAME' } = $1;
            $hshIndexFieldsTmp->{ 'I_ST_DATE' } = $2;

            $hshIndexFieldsTmp->{ 'I_AC_NAME' } =~ s/\s+$//g;    # strip trailing spaces
        }
    }
}

#---- Fetch next text
$a2wTextTmp = $a2wFirstPagePar->getNextText();
}

if ( $hshIndexFieldsTmp != undef
    && $bLog == $TRUE
){
    printf STDERR ( "Index:"
        . " Branch=" . $hshIndexFieldsTmp->{ 'I_BRANCH' }
        . " BranchCode=" . $hshIndexFieldsTmp->{ 'I_BRANCH_CODE' }
        . " AccountName=" . $hshIndexFieldsTmp->{ 'I_AC_NAME' }
        . " StatementDate=" . $hshIndexFieldsTmp->{ 'I_ST_DATE' }
        . "\n"
    );
}

return $hshIndexFieldsTmp;
}

```

Processes given index fields line, splits indexes and returns the same.

7.2.14.3 Starting the Script Example

You use the following command to start AFP2web with the Scripting Facility:

On Windows:

```
afp2web.exe -q -c -mmdin -doc_cold -sp:eyecatcher_mmd.pm -sft:ASA_EBCDIC_FIXED_133 -fd:f1a2w -
pd:p1a2w samples\mmdsample.mmd
```

On Unix:

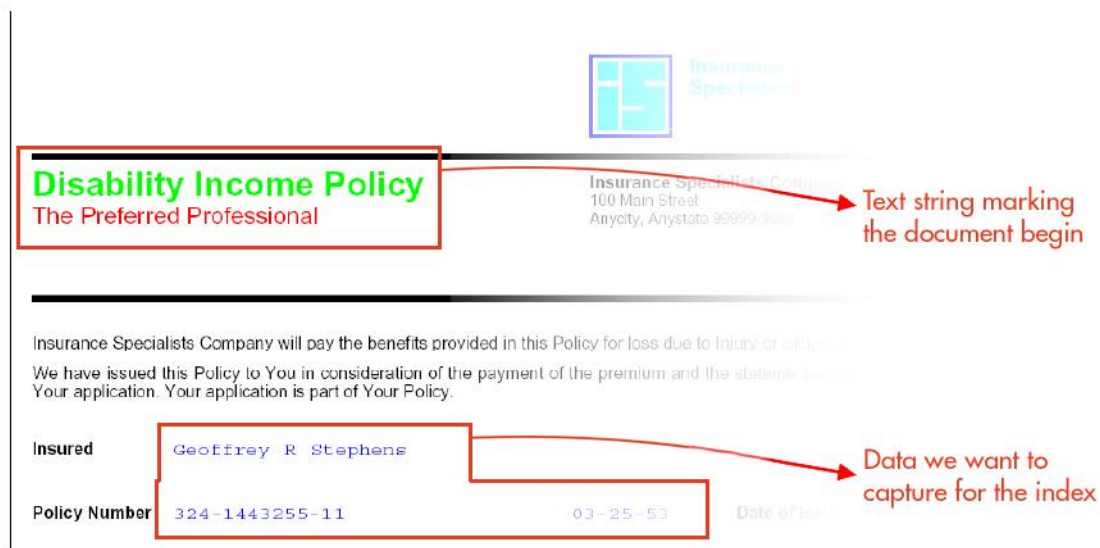
```
./afp2web -q -c -mmdin -doc_cold -sp:eyecatcher_mmd.pm -sft:ASA_EBCDIC_FIXED_133 -fd:f1a2w -
pd:p1a2w samples/mmdsample.mmd
```

7.2.14.4 Results of eyecatcher_mmd.pm

You will find the output file in the output sub folder (the default will be in the sub folder "pdf"):

e:\test\versiontest\Batch\V3.4\Test\lpdmmdtest_rtr\pdf*.*			
↑ Name	Ext	Size	Date
↑ [..]		<DIR>	07/31/2010 13:39
mmdsample_sms.1	idx	100	07/31/2010 13:39
mmdsample_sms.1	pdf	128,670	07/31/2010 13:39
mmdsample_sms.2	idx	94	07/31/2010 13:39
mmdsample_sms.2	pdf	129,572	07/31/2010 13:39
mmdsample_sms.3	idx	97	07/31/2010 13:39
mmdsample_sms.3	pdf	129,590	07/31/2010 13:39

- Three text strings deliver the values for the document index, which shall be written to an output file



7.2.15.1 Using eyecatcher.pm to find text coordinates

First, we want to determine the page position of these text elements. We run the AFP2web Scripting Facility with the following commands in the subroutine afp2web().

eyecatcher.pm - Investigating the text positioning properties:

```
sub afp2web(){
    $APPEND = 0; # append page to Current Document
    $SKIP = 1; # skip page
    $NEWDOC = 2; # new document
    #---- Set default return value
    my $svRetTmp = $APPEND; # default: append page
    #---- Fetch first Text Object
    my $a2wTextTmp = $a2wPagePar->getFirstText();
    #---- Define temp variables
    my $svTextTmp = "";
    my $svTextXPosTmp = 0;
    my $svTextYPosTmp = 0;
    #---- Loop thru all the Text Objects
    while ( $a2wTextTmp != 0 ){
        $svTextTmp = $a2wTextTmp->getText();
        $svTextXPosTmp = $a2wTextTmp->getXPos();
        $svTextYPosTmp = $a2wTextTmp->getYPos();
        print $svTextXPosTmp . "," . $svTextYPosTmp . ">" . $svTextTmp . "\n";
        #---- Fetch next Text Object
        $a2wTextTmp = $a2wPagePar->getNextText();
    }
}
```

```

#---- Increment Page Id
$PageId++;
return $svRetTmp;
}

```

We use the following command in demo.bat to start AFP2web with the Scripting Facility:

```
afp2web.exe -q -c -doc_cold -sp:eyecatcher.pm samples\insure.afp > dumpfile.txt
```

The script produces the following output, which helps us determine the x/y-coordinates of the text strings:

eyecatcher.pm - x/y coordinates found:

```

8496,862>Insurance
8496,1120>Specialists
792,2308>Disability Income Policy
792,2638>The Preferred Professional
7330,2209>Insurance Specialists Company
7330,2418>100 Main Street
7330,2627>Anycity, Anystate 99999-9999
792,4104>Insurance Specialists Company will pay the benefits provided in
this Policy for loss due to Injury or Sickness.
792,4471>We have issued this Policy to You in consideration of the payment
of the premium and the statements made in
792,4681>Your application. Your application is part of Your Policy.
792,5353>Insured
2448,5353>Geoffrey R Stephens
792,6097>Policy Number
2448,6097>324-1443255-11
7114,6097>03-25-53
....additional lines not shown here....

```

7.2.15.2 Using eyecatcher.pm to split documents and retrieve index data

In this example, we focus on the subroutines:

Subroutine	Description
sub initializeDoc()	Saves the current document object. Resets page numbering
sub afp2web()	The return code in this routine tells AFP2web to begin a new document with this page or not, to skip the current page, or to stop AFP2web processing. In this routine, our example recognizes the start page of a new document. Calls the subroutine addPageEyeCatcherIndexes() for the first page of the document.
sub finalizeDoc()	Writes index data to file for this document.
sub addPageEyeCatcherIndexes()	A custom routine that extracts three text values from the current page and saves it as index data.

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document. We use the Perl variable @_ to save the document, which is passed as argument . And we reset the page counter to zero.

eyecatcher.pm - initializeDoc():

```

sub initializeDoc(){
    #---- Get Parameter of initializeDoc( Par: a2w::Document )
    ($a2wDocumentPar) = @_;
    return 0;
}

```

It is equally important that we save the current page object so that we can later access its elements. This is done in the subroutine initializePage().

eyecatcher.pm - initializePage():

```

sub initializePage(){
    #---- Get Parameter of initializePage( Par: a2w::Page )
    ($a2wPagePar) = @_;

```

The routine afp2web() is used to process each document page. This routine formulates the return code value to be passed to AFP2web.

eyecatcher.pm - afp2web() - initializing the return code:

```

# Main entry method
# Return values:
# < 0:error
# 0:append page to Current Document
# 1:skip page
# 2:first page / new document
#-----
sub afp2web(){
    $APPEND= 0;# append page to Current Document
    $SKIP= 1;# skip page
    $NEWDOC= 2;# new document
    $svRetTmp = $APPEND; # default: append page
    .... additional lines with processing logic not shown here...
    return $svRetTmp;
}

```

The processing logic in the routine afp2web() handles splitting the spool file into documents. The rule is: The start page of a new document always has the text "Disability Income Policy" on a fixed position on that page. If this text value is found, we set the return code for AFP2web to initiate processing for the end of the previous document and the begin of a new document.

eyecatcher.pm afp2web() - Using an eyecatcher for document splitting:

```

.....
#---- Fetch first Text Object
my $a2wTextTmp = $a2wPagePar->getFirstText();
#---- Loop thru all the Text Objects
while ( $a2wTextTmp != 0 ){
    #---- If true ==> new doc
    if ( $a2wTextTmp->getXPos() >= 782 &&
        $a2wTextTmp->getXPos() <= 802 &&
        $a2wTextTmp->getYPos() >= 2298 &&
        $a2wTextTmp->getYPos() <= 2318 &&
        $a2wTextTmp->getText() eq "Disability Income Policy" ){
        #---- Reset Page Id
        $PageId = 0;
        $svRetTmp = $NEWDOC;
        last; # leave while loop
    }
    #---- Fetch next Text Object
    $a2wTextTmp = $a2wPagePar->getNextText();
}

```

```
#---- Increment Page Id
$PageId++;
```

When AFP2web closes a document, it will invoke the subroutine `finalizeDoc()`. Here, we fetch and output the index data from the document. This simple routine calls the subroutine `addFirstPageEyecatcherIndexes()`, which is responsible for fetching the index data. You have full control of the document content even during the event of finalizing the document. To get the start page of the document, just use the method `$a2wDocumentPar->getFirstPage()` and scan that page for your index data.

eyecatcher.pm - addFirstPageEyecatcherIndexes() called by finalizeDoc():

```
sub addFirstPageEyecatcherIndexes(){
    #---- Fetch the first Page of Document
    my $a2wFirstPageTmp = $a2wDocumentPar->getFirstPage();
    #---- If a Page exists...
    if ( $a2wFirstPageTmp != 0 ){
        #---- Define the list of Eyecatcher positions we should look for
        my @arrEyeCatcherIndexListTmp = ( ( 2448,5353 ), # Insured
        @(2448,5353)>Geoffrey R Stephens<
        ( 2448,6097 ), # Policy Number @(2448,6097)>324-1443255-11<
        ( 7114,6097 ) ); # Date of Issue @(7114,6097)>03-25-53<
        #---- Define temp variables
        my $arrECILLenTmp = @arrEyeCatcherIndexListTmp / 2;
        .... additional lines not shown here...

        #---- Get the Index list count
        my $IndexPtrTmp = @IndexList;
        #---- Fetch first Text Object
        my $a2wTextTmp = $a2wFirstPageTmp->getFirstText();
        #---- Loop thru all the Text Objects
        while ( $a2wTextTmp != 0 ){
            $svTextTmp = $a2wTextTmp->getText();
            $svTextXPosTmp = $a2wTextTmp->getXPos();
            $svTextYPosTmp = $a2wTextTmp->getYPos();
            #---- Search all the defined Eyecatchers
            for (my $i = 0; $i < $arrECILLenTmp; $i++){
                $svArrPosTmp = $i * 2;
                $svIdxTextXTmp = $arrEyeCatcherIndexListTmp[ $svArrPosTmp ];
                $svIdxTextYTmp = $arrEyeCatcherIndexListTmp[ $svArrPosTmp + 1 ];
                #---- If true ==> we found one of the defined Eyecatchers
                if ( $svTextXPosTmp >= ( $svIdxTextXTmp - 10 ) &&
                $svTextXPosTmp <= ( $svIdxTextXTmp + 10 ) &&
                $svTextYPosTmp >= ( $svIdxTextYTmp - 10 ) &&
                $svTextYPosTmp <= ( $svIdxTextYTmp + 10 ) ){
                    if ( $i == 0 ){
                        $svIndexValueTmp = "Insured=" . $svTextTmp;
                    } elsif ( $i == 1 ){
                        $svIndexValueTmp = "Policy Number=" . $svTextTmp;
                    } elsif ( $i == 2 ){
                        $svIndexValueTmp = "Date of Issue=" . $svTextTmp;
                    }
                    #---- Add Eyecatcher to Index List
                    @IndexList[$IndexPtrTmp++] = $svIndexValueTmp;
                    if ( $bLog == $TRUE ){
                        print "$svIndexValueTmp\n";
                    }
                }
            }
        }
        #---- Fetch next Text Object
```

```

        $a2wTextTmp = $a2wFirstPageTmp->getNextText();
    }
}
}

```

7.2.15.3 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

On Windows:

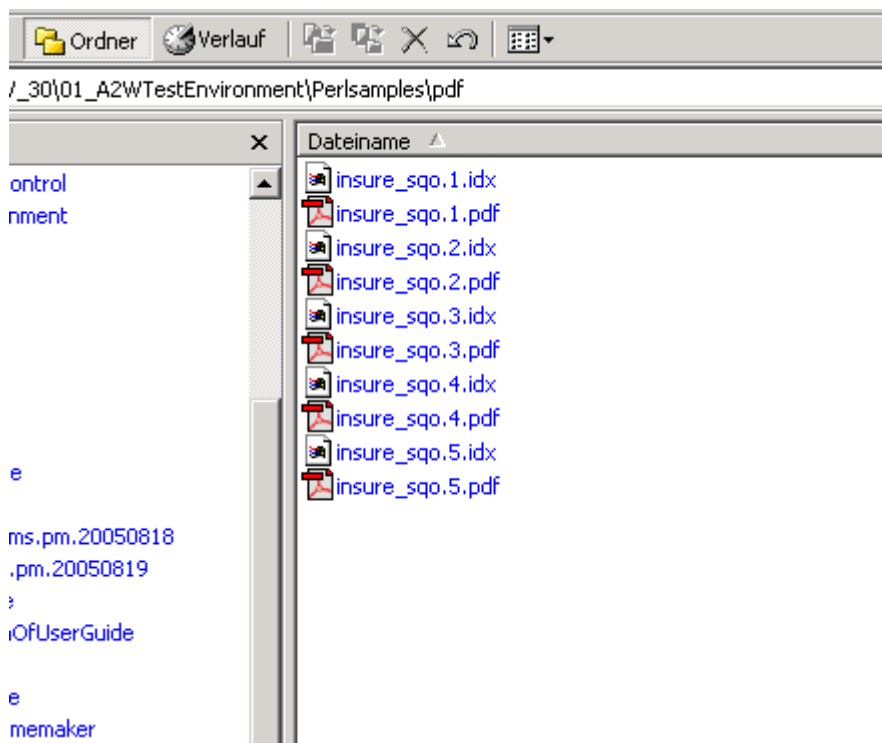
```
afp2web.exe -q -c -doc_cold -sp:eyecatcher.pm samples\insure.afp
```

On Unix:

```
./afp2web -q -c -doc_cold -sp:eyecatcher.pm samples/insure.afp
```

7.2.15.4 Results of eyecatcher.pm

You will find the output file in the output sub folder (the default will be in the sub folder /pdf):



The contents of the index file:

eyecatcher.pm - contents of an index file:

```

DocId=1, DocType=PDF, DocName=./pdf/insure_s2qc.1.pdf, PageCount=3, Size=114669
Insured=Geoffrey R Stephens
Policy Number=324-1443255-11
Date of Issue=03-25-53

```

The resulting PDF document will have a start page, which begins like this:



Disability Income Policy

The Preferred Professional

Insurance Specialists Company
100 Main Street
Anycity, Anystate 99999-9999

Insurance Specialists Company will pay the benefits provided in this Policy for loss due to Injury or Sickness.
We have issued this Policy to You in consideration of the payment of the premium and the statements made in Your application. Your application is part of Your Policy.

Insured Geoffrey R Stephens

Policy Number 324-1443255-11 03-25-53 Date of Issue

NON-CANCELLABLE AND GUARANTEED CONTINUABLE TO AGE 65. NO CHANGE IN PREMIUM RATES.

7.2.16 processDocInfo.pm: Access/Process document information

processDocInfo.pm shows how to access/process input spool's info like properties, bookmarks, metadata etc using Scripting Facility exposed a2w::Document and a2w::Page objects. It also prints input spool's info on console. This example is valid only for PDF input format for now.

7.2.16.1 Understanding processDocInfo.pm

In this example, we focus on the following subroutines:

Subroutine	Description
sub initializeDoc()	Calls the subroutine _processDocumentProperties(), _processDocumentMetadata() and _processDocumentBookmarks() to process input spool's info
sub _processDocumentProperties	Processes the properties like Author, Keyword etc of current document
sub _processDocumentMetadata	Processes the metadata of current document
sub _processDocumentBookmarks	Processes the bookmark of current document

sub _processDocumentBookmarkChildren	Processes the child bookmarks of current document bookmark
sub _printBookmark	Prints the bookmark details

The subroutine initializeDoc() is called when AFP2web signals the beginning of a new document.

processDocInfo.pm – initializeDoc() - Calling _processDocumentProperties:

```
_processDocumentProperties( $a2wDocumentPar );
```

The subroutine _processDocumentProperties() fetches all document properties and process them (i.e, prints them on console)

processDocInfo.pm – _processDocumentProperties:

```
#-----
# Processes document properties
#
# Parameters:
#     a2w::Document instance
#
# Return Value
#     None
#-----
sub _processDocumentProperties{
    if ( $bLog == $TRUE ){
        printf STDERR ( "_processDocumentProperties()\n" );
    }

    #---- Get Parameter
    #
    # a2w::Document
    #
    my $a2wDocPar = shift;

    #---- Assert parameter
    if ( $a2wDocPar == 0 ){
        if ( $bLog == $TRUE ){
            printf STDERR ( "\tInvalid document handle to process properties\n" );
        }

        return;
    }

    printf STDERR( "*** " . $a2wDocPar->getName() . " document properties ****\n");

    #---- Get property name list (as defined in "a2w::DocumentConstants")
    my @arrConstantListTmp = sort keys( %a2w::DocumentConstants:: );
    my $sPropNameTmp = "";
    my $sPropValueTmp = "";

    #---- Loop through each property and process them ----#
    for ( my $i = 0; $i < @arrConstantListTmp; $i++ ){
        #---- Print only constants whose name starts with "PROP" keyword
        if ( @arrConstantListTmp[ $i ] =~ /^PROP/ ){
            #---- Fetch the property details
            $sPropNameTmp = ${ %a2w::DocumentConstants::->{ @arrConstantListTmp[ $i ] } };
            $sPropValueTmp = $a2wDocumentPar->getProperty( $sPropNameTmp );
        }
    }
}
```

```

        if ( $sPropValueTmp ne "" ){
            print STDERR ( $sPropNameTmp
                          . "="
                          . $sPropValueTmp
                          . "\n"
                          );
        } # if ( $sPropValueTmp ne "" ){
    } # if ( @arrConstantListTmp[ $i ] =~ /^PROP/ ){
}

printf STDERR ( "***** End document properties *****\n\n" );
}

processDocInfo.pm - initializeDoc() - Calling _processDocumentMetadata:

_processDocumentMetadata( $a2wDocumentPar );

The subroutine _processDocumentMetadata() fetch document metadata content and process it (i.e, print
it on console).

processDocInfo.pm - _processDocumentMetadata:

#-----
# Processes document metadata
#
# Parameters:
#     a2w::Document instance
#
# Return Value
#     None
#-----
sub _processDocumentMetadata{
    if ( $bLog == $TRUE ){
        printf STDERR ( "_processDocumentMetadata()\n" );
    }

    #---- Get Parameter
    #
    # a2w::Document
    #
    my $a2wDocPar = shift;

    #---- Assert parameter
    if ( $a2wDocPar == 0 ){
        if ( $bLog == $TRUE ){
            printf STDERR ( "\tInvalid document handle to process metadata\n" );
        }

        return;
    }

    printf STDERR ( "***** " . $a2wDocPar->getName() .
" document metadata *****\n\n" );

    #----- Pprocess document meta data -----#
    my $sDocMetadataTmp = $a2wDocumentPar->getMetaDataStream();
    if ( $sDocMetadataTmp ne "" ){
        printf STDERR ( $sDocMetadataTmp . "\n" );
    }
}

```

```

    printf STDERR ( "***** End document metadata *****\n\n" );
}

```

processDocInfo.pm - initializeDoc() - Calling _processDocumentBookmarks:

```

_processDocumentBookmarks( $a2wDocumentPar );

```

The subroutine `_processDocumentBookmarks()` navigates through the top level Document bookmarks, calls `_printBookmark()` to print bookmark details and calls `_processDocumentBookmarkChildren()` to process the bookmark's children.

processDocInfo.pm - _processDocumentBookmarks:

```

#-----

# Processes document bookmarks
#
# Parameters:
# a2w::Document instance
#
# Return Value
# None
#-----

sub _processDocumentBookmarks{
    if ( $bLog == $TRUE ){
        printf STDERR ( "_processDocumentBookmarks()\n" );
    }
    #---- Get Parameter
    #
    # a2w::Document
    #
    my $a2wDocPar = shift;
    #---- Assert parameter
    if ( $a2wDocPar == 0 ){
        if ( $bLog == $TRUE ){
            printf STDERR ( "\tInvalid document handle to process bookmarks\n" );
        }
        return;
    }
    printf STDERR ( "***** " . $a2wDocPar->getName() . " document bookmarks
*****\n" );
    #---- Fetch first bookmark
    my $a2wBookmarkTmp = $a2wDocPar->getFirstBookmark();
    #---- Loop through all bookmarks
    while( $a2wBookmarkTmp ){
        #---- Print bookmark
        _printBookmark( $a2wBookmarkTmp );
        #---- Process bookmark's child(ren), if any exist
        if ( $a2wBookmarkTmp->getFirstChild() ){
            _processDocumentBookmarkChildren( $a2wBookmarkTmp );
        }
        #---- Fetch next bookmark
        $a2wBookmarkTmp = $a2wDocPar->getNextBookmark();
    }
    printf STDERR ( "***** End *****\n" );
}

```

The subroutine `_printBookmark()` prints bookmark details such as bookmark title and the number of the target page. It prints an indentation according to the bookmark nesting level. The information printed in the the following format:

```

<Spaces based on level number ><BookmarkTitle><(TargetPageNumber)>

processDocInfo.pm - _printBookmark():

if ( $bLog == $TRUE ){
    printf STDERR ( "_printBookmark()\n" );
}
#---- Get Parameter
#
# a2w::Bookmark instance
#
my $a2wBookmarkPar = shift;
#---- Assert parameter
if ( $a2wBookmarkPar == 0 ){
    if ( $bLog == $TRUE ){
        printf STDERR ( "\tInvalid bookmark to print\n" );
    }
}
#---- Fetch bookmark info
my $sBookmarkTitleTmp = $a2wBookmarkPar->getTitle(); # Fetch bookmark
title
my $iLevelNrTmp = $a2wBookmarkPar->getLevelNumber(); # Fetch bookmark
level number
my $sTargetTmp = $a2wBookmarkPar->getTargetName(); # Fetch bookmark
target name
#---- Print Bookmark details
#---- Level number is used for alignment of child bookmarks
if ( $iLevelNrTmp > 0 ){
    printf STDERR ( '%*s%s (%s)%s',
        $iLevelNrTmp * 2,
        " ",
        $sBookmarkTitleTmp,
        $sTargetTmp,
        "\n"
    );
}

```

7.2.16.2 Starting the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

Windows:

```
afp2web.exe -q -c -afp -doc_cold -sp:processDocInfo.pm samples\insure.pdf
```

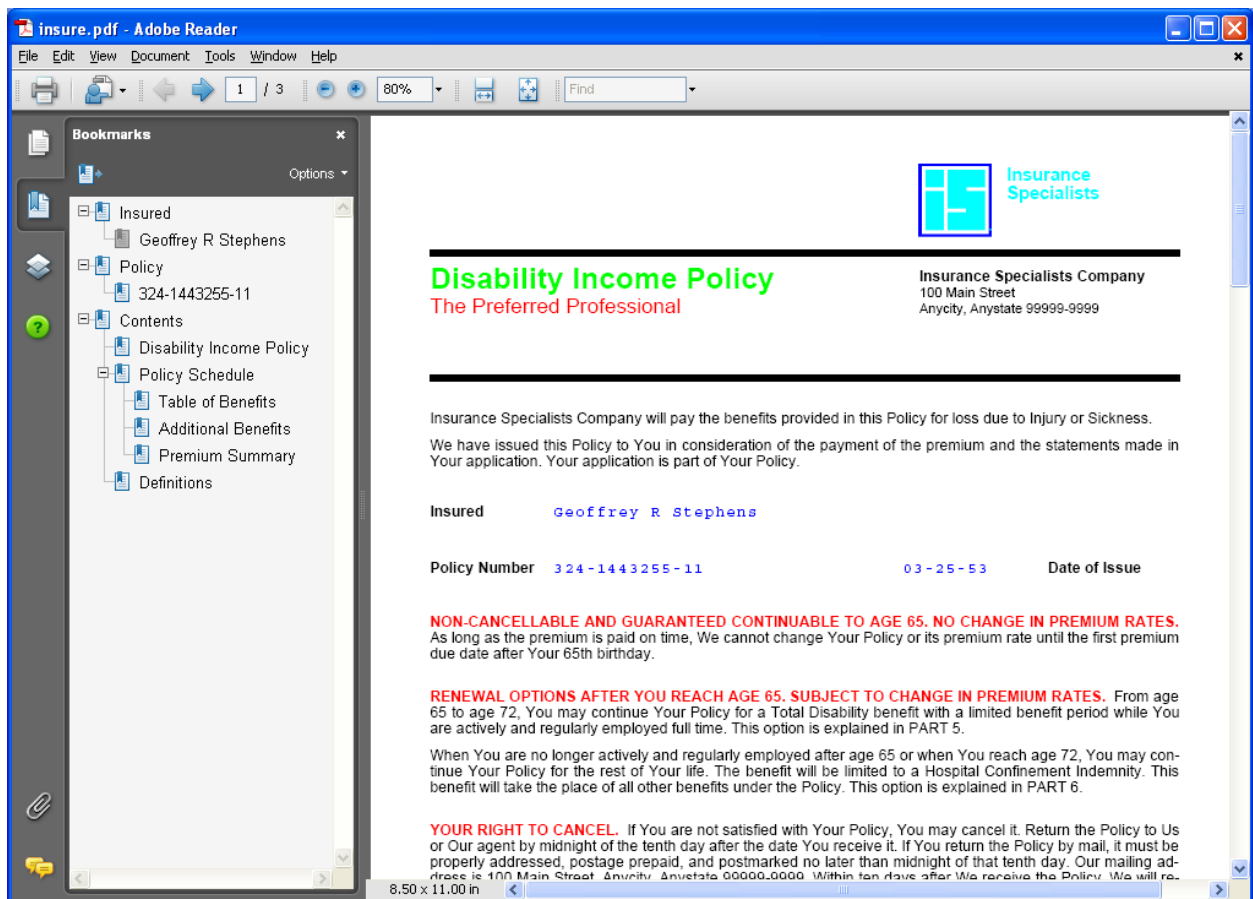
Unix:

```
./afp2web -q -c -afp -doc_cold -sp:processDocInfo.pm samples/insure.pdf
```

7.2.16.3 Results of processDocInfo.pm

The bookmark details of the PDF input document and the corresponding output from AFP2web are given below.

The bookmark details of the input file insure.pdf as shown by Acrobat Reader:



Output of AFP2web when processDocInfo.pm is used as script procedure for the same input file insure.pdf. The document information's prints on the console as given below and the number in parenthesis is the target page number of the respective bookmark.

***** insure document properties *****

```
Author=Maas Holding GmbH
CreationDate=D:20100722060748Z
Creator=AFP2web v4.0 [Built for Windows 2003/2008 32-bit on Jul 22 2010 at 11:36:30]
Keywords=AFP LPD PDF TIFF
Producer=Maas C PDF Library V3.1
Subject=AFP and TIFF Conversions (afp2web.com)
Title=AFP2web v4.0 [Built for Windows 2003/2008 32-bit on Jul 22 2010 at 11:36:30]
***** End document properties *****
```

***** insure document metadata *****

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ix="http://ns.adobe.com/ix/1.0/">
  <rdf:Description rdf:about="" xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
    <pdf:Producer>Maas C PDF Library V3.1</pdf:Producer>
    <pdf:Keywords>AFP LPD PDF TIFF</pdf:Keywords>
  </rdf:Description>
  <rdf:Description rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/">
    <xmp:CreateDate>2010-07-22T06:07:48Z</xmp:CreateDate>
    <xmp:CreatorTool>AFP2web v4.0 [Built for Windows 2003/2008 32-bit on Jul 22 2010 at
```

```

11:36:30]</xmp:CreatorTool>
  <xmp:Identifier>
    <rdf:Bag>
      <rdf:li>e1dbbd2f-59c2-480e-b03b-37e2f6dfe906</rdf:li>
    </rdf:Bag>
  </xmp:Identifier>
</rdf:Description>
<rdf:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:creator>
    <rdf:Seq>
      <rdf:li>Maas Holding GmbH</rdf:li>
    </rdf:Seq>
  </dc:creator>
  <dc:title>
    <rdf:Alt>
      <rdf:li xml:lang="x-default">AFP2web v4.0 [Built for Windows 2003/2008 32-bit on Jul 22 2010
at 11:36:30]</rdf:li>
    </rdf:Alt>
  </dc:title>
  <dc:description>AFP and TIFF Conversions (afp2web.com)</dc:description>
</rdf:Description>
<rdf:Description rdf:about="" xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">
  <pdfaid:part>1</pdfaid:part>
  <pdfaid:conformance>B</pdfaid:conformance>
</rdf:Description>
<rdf:Description rdf:about="" xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#">
  <xmpMM:DocumentID>uuid:e1dbbd2f-59c2-480e-b03b-37e2f6dfe906</xmpMM:DocumentID>
  <xmpMM:VersionID>1.0.0</xmpMM:VersionID>
  <xmpMM:RenditionClass>default</xmpMM:RenditionClass>
  <xmpMM:History>
    <rdf:Seq>
      <rdf:li rdf:parseType="Resource">
        <stEvt:action>created</stEvt:action>
        <stEvt:parameters>Maas C PDF Library V3.1</stEvt:parameters>
        <stEvt:softwareAgent>AFP2web v4.0 [Built for Windows 2003/2008 32-bit on Jul 22 2010 at
11:36:30]</stEvt:softwareAgent>
        <stEvt:when>2010-07-22T06:07:48Z</stEvt:when>
      </rdf:li>
    </rdf:Seq>
  </xmpMM:History>
</rdf:Description>
<rdf:Description rdf:about="" xmlns:a2w="http://www.oxseed.com/xmp/1.0/a2w/">
  <a2w:IndexList>
    <rdf:Seq>
      <rdf:li rdf:parseType="Resource">
        <a2w:Name>Insured</a2w:Name>
        <a2w:Value>Geoffrey R Stephens</a2w:Value>
      </rdf:li>
      <rdf:li rdf:parseType="Resource">
        <a2w:Name>Policy</a2w:Name>
        <a2w:Value>324-1443255-11</a2w:Value>
      </rdf:li>
      <rdf:li rdf:parseType="Resource">
        <a2w:Name>Contents</a2w:Name>
        <a2w:Value>Disability Income Policy</a2w:Value>
      </rdf:li>
      <rdf:li rdf:parseType="Resource">

```

```

        <a2w:Name>Contents</a2w:Name>
        <a2w:Value>Policy Schedule</a2w:Value>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <a2w:Name>Contents</a2w:Name>
        <a2w:Value>Policy Schedule</a2w:Value>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <a2w:Name>Contents</a2w:Name>
        <a2w:Value>Table of Benefits</a2w:Value>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <a2w:Name>Contents</a2w:Name>
        <a2w:Value>Additional Benefits</a2w:Value>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <a2w:Name>Contents</a2w:Name>
        <a2w:Value>Premium Summary</a2w:Value>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <a2w:Name>Contents</a2w:Name>
        <a2w:Value>Definitions</a2w:Value>
    </rdf:li>
</rdf:Seq>
</a2w:IndexList>
</rdf:Description>
</rdf:RDF>
<?xpacket end="r"?>
***** End document metadata *****

***** insure document bookmarks *****
Insured (1)
    Geoffrey R Stephens (1)
Policy (1)
    324-1443255-11 (1)
Contents (1)
    Disability Income Policy (1)
    Policy Schedule (2)
        Table of Benefits (2)
        Additional Benefits (2)
        Premium Summary (2)
    Definitions (3)
***** End document bookmarks *****

```

7.2.17 sortPageTextObjs.pm: Sorting Text Objects in their Print Sequence

Raw AFP print data does not necessarily group and present text objects in the order in which they are printed. This makes it difficult to read AFP data dumped into text files.

sortPageTextObjs.pm shows how to access all text objects of a page. It sorts all text objects according to their X/Y coordinate positions and then writes their text values to an output file (the dump file) in this order.

7.2.17.1 Understanding sortPageTextObjs.pm

In this example, we focus on the subroutines:

Subroutine	Description
sub afp2web()	<p>Main entry for AFP2web.</p> <p>The return code in this routine tells AFP2web to begin a new document with this page or not, to skip the current page, or to stop AFP2web processing.</p> <p>In this routine, our example collect the text of the page and then calls the subroutine buildLines() to sort and write the text to the dump file.</p>
sub buildLines()	Sorts the text by y-coordinate (line number) and then by x-coordinate (position within the line)
sub complex_arrays()	Specifies which keys to use for the sort algorithm.

The routine afp2web() is the main entry point for AFP2web. This routine is called for each document page.

The first action is to initialize the return code value, which will control AFP2web processing. In addition, the current page ID (number of the document in the spool file) is written to the dump file. We do not show this here.

In the following, we show how afp2web() collects all text objects of the current page and then calls custom subroutine buildLines to sort the text.

sortPageTextObjs.pm - afp2web() - Gathering text objects and processing the results:

```
#---- Add the Page Text Objects to an unsorted Object List
@unsortedObjList = ();
my $PtrTmp = @unsortedObjList;
#---- Define temp variables
my $svTextTmp = "";
my $svTextXPosTmp = 0;
my $svTextYPosTmp = 0;
#---- Fetch first Text Object
my $a2wTextTmp = $a2wPagePar->getFirstText();
#---- Loop thru all the Text Objects
while ( $a2wTextTmp != 0 ){
    $svTextTmp = $a2wTextTmp->getText();
    $svTextXPosTmp = $a2wTextTmp->getXPos();
    $svTextYPosTmp = $a2wTextTmp->getYPos();
    if ( $bLog == $TRUE ){
        print " @(" . $svTextXPosTmp . "," . $svTextYPosTmp . ")>" . $svTextTmp .
            "<\n";
    }
    #---- Add Object to unsorted Object List
    @unsortedObjList[$PtrTmp++] = ({XPOS => $svTextXPosTmp,
        YPOS => $svTextYPosTmp,
        TEXT => $svTextTmp
    });
    #---- Get the next Text Object
    $a2wTextTmp = $a2wPagePar->getNextText();
} # end-while
#---- Build/Dump Lines
buildLines();
```

The subroutine buildLines() first sorts the text and then prints the text in the sort order to the dump file. The Y-coordinate determines the line number. Whenever a new line occurs, the new line number is printed to the dump file.

sortPageTextObjs.pm - buildLines():

```
sub buildLines(){
    #---- Define temp variables
    my $LineTmp = "";
    my $LineCountTmp = 0;
    my $currYPosTmp = 0;
    my @sortedObjList = ();
    #---- Sort Array: first Key is YPOS, second Key is XPOS
    @sortedObjList = sort complex_arrays @unsortedObjList;
    #---- Build Lines based on sorted Array
    foreach (@sortedObjList){
        $YPosTmp = $_->{YPOS};
        $TextTmp = $_->{TEXT};
        if ( $currYPosTmp != $YPosTmp && $currYPosTmp != 0 ){
            #---- Add line
            printf fDumpFile ("%06d:", $LineCountTmp);
            print fDumpFile (" $LineTmp\n");
            $LineCountTmp++;
            $LineTmp = $TextTmp;
        }
        else{ # append Text to current line
            $LineTmp = $LineTmp . $TextTmp;
        }
        $currYPosTmp = $YPosTmp;
    }
}
```

The subroutine `complex_arrays()` delivers the first set of sort keys required for the sort command. In this case, the sort should be done first for the y-coordinates (the line number of the text objects) and then for the x-coordinates (the position of the text within the line).

sortPageTextObjs.pm - complex_arrays():

```
#-----
# Sorting Algorithm
#
#1st Key is YPOS, 2nd Key is XPOS
#-----
sub complex_arrays{
    $a->{YPOS} <=> $b->{YPOS} ||
    $a->{XPOS} <=> $b->{XPOS};
}
```

7.2.17.2 Running the Script Example

You use the following command in the demo batch file to start AFP2web with the Scripting Facility:

On Windows:

```
afp2web.exe -q -c -doc_cold -sp:sortPageTextObjs.pm samples\insure.afp
```

On Unix:

```
./afp2web -q -c -doc_cold -sp:sortPageTextObjs.pm samples/insure.afp
```

7.2.17.3 Results of sortPageTextObjs.pm

The resulting PDF document will have a start page, which begins like this:



Insurance
Specialists

Disability Income Policy

The Preferred Professional

Insurance Specialists Company
100 Main Street
Anycity, Anystate 99999-9999

Insurance Specialists Company will pay the benefits provided in this Policy for loss due to Injury or Sickness.
We have issued this Policy to You in consideration of the payment of the premium and the statements made in Your application. Your application is part of Your Policy.

Insured Geoffrey R Stephens

Policy Number 324-1443255-11

03-25-53

Date of Issue

NON-CANCELLABLE AND GUARANTEED CONTINUABLE TO AGE 65. NO CHANGE IN PREMIUM RATES.

You will find the dump file in the output subfolder (the default is the subfolder /pdf). The following example shows a few lines of the content you will find in the dump file. Each text line is preceded by the line number. Please note that additional line breaks were added when we inserted this example into our tutorial:

sortPageTextObjs.pm - contents of the dump file (extract):

```
===== Page 000001
=====
000000:Demo Version
000001:Specialists
000002:Insurance Specialists Company
000003:Disability Income Policy
000004:100 Main Street
000005:Anycity, Anystate 99999-9999
000006:Demo Version
000007:Insurance Specialists Company will pay the benefits provided in
this Policy for loss due to Injury or Sickness.
000008:We have issued this Policy to You in consideration of the payment
of the premium and the statements made in
000009:Your application. Your application is part of Your Policy.
000010:Demo VersionGeoffrey R Stephens
000011:Policy Number324-1443255-1103-25-53Demo Version
000012:Demo Version
000013:As long as the premium is paid on time, We cannot change Your Policy
or its premium rate until the first premium
000014:due date after Your 65th birthday.
000015:RENEWAL OPTIONS AFTER YOU REACH AGE 65. SUBJECT TO CHANGE IN PREMIUM
RATES.Demo Version
000016:65 to age 72, You may continue Your Policy for a Total Disability
benefit with a limited benefit period while You
```

000017:are actively and regularly employed full time. This option is explained in PART 5.
..... additional lines not shown here....

7.3 PDF/A Support in AFP2web

This appendix is a supplement to the AFP2web Version 4.x User Guide and Reference. It describes how to use AFP2web Version 4.x to produce PDF/A-compliant output.

7.3.1 AFP2web's Commitment to the PDF/A Standard

AFP2web produces PDF output, which conforms to the PDF/A-1b standard. The standard is defined in ISO 19005-1. Document management — Electronic document file format for longterm preservation — Part 1: Use of PDF 1.4 (PDF/A-1) Reference number ISO 19005-1:2005(E) and Corrigendum 1 (ISO 19005-1:2005, TECHNICAL CORRIGENDUM 1, Published on 2007-04-01, Corrected version of 2005-12-01).

PDF/A-compliant documents are self contained PDF files, which are used for long-term archival. The standard is based on PDF Version 1.4 and imposes restrictions on the PDF functionality.

Adobe Systems Limited (www.adobe.com), the originator of this technology, states, "[PDF/Acompliant files] are primarily used for archiving. Because long-term preservation is the goal, the document must contain only what is needed for opening and viewing throughout the intended life of the document. For example, PDF/A-compliant files can contain only text, raster images, and vector objects; they cannot contain encryption and scripts. In addition, all fonts must be embedded so the documents can be opened and viewed as created."

7.3.2 Important Notices

- AFP2web does not validate the output for compliance to PDF/A. It remains the responsibility of the user to set the AFP2web parameters correctly and to avoid features, which are not allowed for PDF/A, such as font referencing or transparency.
- XMP metadata embedded on PDF/A output will contain document indexes packed in using AFP2web's schema and namespace URL for AFP2web's schema will be as given below

<http://www.oxseed.com/xmp/1.0/a2w/>

Previously the namespace URL is used to be "<http://www.maas.de/xmp/1.0/xmpMHT/>". Now it is obsolete.

7.3.3 INI Parameters and Command Line Options

To produce output for PDF/A-1b, the following parameters are relevant:

INI Parameter	Command Line Option	Description
OutputFormat=PDF/A	-pdfa	Specifies the output format, which meets the PDF/A conformance level: PDF/A-1b.
Strict=on	-strict	When running AFP2web to produce PDF/A output, you must use the processing parameter Strict=on. Doing this will ensure that AFP2web will stop processing if any resources (fonts or AFP resources) are missing.
CMYKtoRGB=on		Ensures that all colors are converted to use only one color space. AFP2web converts all colors to RGB. Note: AFP2web uses ICC sRGB IEC61966-2-1 color profile as OutputIntent. Important: Do not use the command line option -nocmyktorgb, because this will overwrite the INI parameter.

7.3.4 Fonts

For PDF/A the fonts must be embedded in the PDF file.

AFP2web supplies a mapping table for font processing. In this file the section [CHARSET RENDERING] is used to define how to process an AFP character set. The processing mode can either use the original AFP font or a substitution font.

Configuration Setting	Description
mapping.def, section [CHARSET RENDERING], Rendering flag	Please set the rendering flag to either:
	0 (this is the default for AFP fonts)
	2 to embed substitution font. (For output to PDF/A, however, this flag MUST NOT be used - see the note below.)
	Important: <ol style="list-style-type: none"> Do not use the rendering flag 1 to reference fonts, because this setting is not allowed for PDF/A. AFP2web does not support rendering flag 2 to embed substitution font for PDF/A output.

7.3.5 Metadata in XMP Format

AFP2web generates metadata in XMP format automatically, embeds the metadata in PDF as a stream object, and references it using the "Metadata" entry in the "Catalog" dictionary.

AFP2web maps metadata as shown in the table below:

INI File Parameter	XMP metadata		Values
Field	Type		
Title	dc:title	Text	Title as specified in the INI parameter. Default: AFP2web Evaluation Version
Subject	dc:subject	Text	Subject as specified in the INI parameter. Default: Please register
Keywords	pdf:keywords	Text	Keywords as specified in the INI parameter. Default: AFP LPD PDF TIFF
---	dc:creator	Text	AFP2web v<version number> [Built for <OS name> <Machine architecture> on <Date> at <Time>]
---	xmp:CreatorTool	Text	AFP2web v<version number> [Built for <OS name> <Machine architecture> on <Date> at <Time>]
---	pdf:Producer	Text	Maas C PDF Library Vx.x
---	xmp:CreateDate	Text	Date and Time of PDF creation

AFP2web also generates metadata for outline fonts.

7.3.6 Features Not to Be Used for PDF/A

The following table lists the features, which are not compliant to PDF/A, and suggests how to avoid them when using AFP2web to create PDF/A-compliant documents:

Limitation	Solution
No hyperlinks to external files.	Hyperlink annotations are not possible. Do not use the Scripting Facility "addAnnotation" interface of "a2w::Page".
Scripting Facility, a2w::Page->addText	Do not use the Scripting Facility "addText" interface of "a2w::Page". It does not embed the font.

Encryption	Encryption is not allowed for PDF/A. Therefore do not set PDF Security options (this applies to the INI parameter PDFSecurity or the command line option -ps).
Transparency	Do not use transparency. When producing output for PDF/A, AFP2web will remove the transparency.
Watermark	Do not use AFP2web to insert text as watermark. Do not use the INI parameter Watermark nor the command line option -wm.

List of Figures

Figure 1 Scenario: Document Archival and Index Update	22
Figure 2 Scenario: On-the-Fly Conversion for PDF On-Demand.....	23
Figure 3 Scenario: AFP Viewer on Workstations.....	24
Figure 4 PDF Bookmarks for Index Elements	41
Figure 5 PDF with watermark	42
Figure 6 fonts example	48
Figure 7 AFP font, monospaced	52
Figure 8 Mapped to Gothic in PDF	52
Figure 9 The original AFP monospaced font	53
Figure 10 Mapped to Lucida Console in PDF	53
Figure 11 AFP original font monospaced	55
Figure 12 Automatically embedded Gothic.....	56
Figure 13 Courier Font Selected For Embedding.....	56
Figure 14 Scripting Facility - Document Splitting	58
Figure 15 Scripting Facility - Data Extraction and Indexing.....	59
Figure 16 Scripting Facility - Archival Flow	60
Figure 17 sf_api_ccrrgbb_image.....	251

List of Tables

Table 1 INI file.....	110
Table 2 Command line.....	110

Index

No index entries found.