Documentation

DocBridge Mill

Administration and Implementation Manual

May 2010

compart

Compart AG
Otto-Lilienthal-Str. 38
71034 Böblingen
Germany
Phone +49-7031-6205-0
www.compart.net

Compart AG
Otto-Lilienthal-Str. 38
71034 Böblingen
Germany
Phone: +49 7031 6205-0
Fax: +49 7031 6205-555

Homepage: http://www.compart.net
E-Mail: info@compart.net

4th Edition (May 2010)

Table of Contents

List of Figures

List of Tables

About this Manual

The manual describes the implementation, usage, and settings of Compart DocBridge Mill and is based on the DocBridge Mill version 201004 and higher.

Audience

This manual is written for guidance and instruction for those who are responsible for the installation, implementation, customization, and administration of the product. Assumed is the fundamental knowledge of a system administrator who is familiar with command line instructions and SGML respectively XML syntax. Furthermore, the person working with this manual should have a solid understanding of document structures and its objects.

Beside the descriptions of the handling with the product as well as with its profiles and functions this manual provides an introductory insight into the general functionality of the product that makes it easier for the reader to understand potential fields of application and the concept of the product.

Structure of this Manual

Chapter 1. DocBridge Mill Introduction on page 1 informs the user about the general functions of the product and its fields of application. It provides an insight into the features of the product.

Chapter 2. DocBridge Mill Installation on page 16 explains how to install the product.

Chapter 3. DocBridge Mill Command Line Parameters on page 19 describes how to call the product, which parameters are available, and how they can be used.

Chapter 4. JavaScript in DocBridge Mill 64 describes how to use JavaScript when working with DocBridge Mill's components cpmill.

Chapter 5. Customization of Profiles on page 143 describes the customization of the product profiles that are used to control several functions and processes.

Appendix A: Deprecated JavaScript Constructs on page 276 lists the JavaScript constructs that should not be used anymore working with the current software version.

Appendix B: Code Page Names on page 277 lists the code pages supported by DocBridge Mill.

Appendix C: Paper Formats on page 280 lists paper formats and related keywords.

Appendix D: Barcodes on page 284 lists the types that are supported by Compart software products.

Appendix E: Units of Measurements on page 285 lists the unit types and the appropriate abbreviations.

Appendix F: Monitored Messages on page 286 contains tables that list the monitored messages.

Other Important Sources of Information

Because we continue to develop the products, the reader should inquire for the newest information about it. The most current source is the Compart homepage http://www.compart.net.

Related Documentation

- *Compart Products Reference Guide for Messages and Codes*
- *DocBridge Delta User's Guide*
- *DOPE/compose Administration and Implementation Manual*
- *Compart Imposition Support User's Guide*

Trademarks

- LuraDocument JBIG2 and LuraWave JP2 © 2006 LuraTech Imaging GmbH

Summary of Changes

The following table summarizes changes made to *DocBridge Mill Administration and Implementation Manual*.

Table 1: Change Record

Revision No.	Date	Description	Chapter
24	Jan 2009	cpmcopy Command Line Parameters	Chap. 3.2
25	Jun 2009	MFF Filter Overview cpmill Command Line Parameters and Variables Built-in Functions Page cpmill Profile Elements MFF Filter Profiles Monitored Messages	Chap. 1.1 Chap. 3.3 Chap. 4.15 Chap. 4.2 Chap. 5.2 Chap. 5.3 App. F
26	Jul 2009	cpmcopy Command Line Parameters File Handling Page cpmill Profile Elements Code Pages Names	Chap. 3.2 Chap. 4.13 Chap. 4.2 Chap. 5.2 App. B
27	Aug 2009	cpmill Command Line Parameters and Variables General Information Items File Handling cpmill Profile Elements Monitored Messages Frequently Asked Questions – FAQ	Chap. 3.3 Chap. 4.1 Chap. 4.10 Chap. 4.13 Chap. 5.2 App. F App. G
28	Sep 2009	Items File Handling Frequently Asked Questions – FAQ	Chap. 4.10 Chap. 4.13 App. G

Table 1: Change Record

Revision No.	Date	Description	Chapter
29	Oct 2009	cpmill Command Line Parameters and Variables Page File Handling Monitored Messages	Chap. 3.3 Chap. 4.2 Chap. 4.13 App. F
30	Nov 2009	DocBridge Mill Program Modules CpCOLD Command Line Parameters and Variables cpmill Command Line Parameters and Variables General Information Monitored Messages	Chap. 1.3 Chap. 3.1 Chap. 3.3 Chap. 4.1 App. F
31	Jan 2010	File Handling Built-in Functions	Chap. 4.13 Chap. 4.15
32	Feb 2010	cpmcopy Command Line Parameters	Chap. 3.2
33	Mar 2010	cpmcopy Command Line Parameters	Chap. 3.2
34	Apr 2010	cpmcopy Command Line Parameters Page Items	Chap. 3.2 Chap. 4.2 Chap. 4.10
35	May 2010	Page File Handling Barcodes	Chap. 4.2 Chap. 4.13 App. D

At present, *DocBridge Mill Administration and Implementation Manual* is not part of a change service. If necessary, the user must ask Compart, if the copy at hand is still up-to-date.

1. DocBridge Mill Introduction

DocBridge Mill is a product of the Compart DocBridge family, which relates to the subject of *Document Processing*. With DocBridge Mill you can convert documents, document batches, or spools (print output) of different occurrences in many ways. Controlled by a configurable profile the batches or documents to be processed can be separated, distributed, classified, indexed, and converted into formats normally be used in document processing. Thus the documents can be displayed, printed, archived almost anywhere, or otherwise be analyzed and processed.

Like all products of the DocBridge family, also this product orientates itself to the greatest possible extent to standards. Beside designed modularly it works with a strong optimized and operating system independent code.

The following chapters describe at first the concept of the DocBridge products and give an overview about its functionality. The functions and DocBridge Mill's area of application are presented in detail.

1.1. Concept of the DocBridge Products

The Compart DocBridge products are components of a complete processing concept that unites the different document processing functions with their general common kernel. In this way, Compart focuses on the development on the base functionalities. The result is a highly optimized and platform independent code that sets standards for all areas of functionalities and performance.

1.1.1. Presentation Area (PA) and Mixed Format Filter (MFF)

The document processing functions described below consist mainly of input and output processing of documents or document batches. Therefore, it is possible to separate all traditional processing functions like scanning, fax processing, spool output processing, and e-mail processing, printing, and converting.

Table 2: Input and Output Functions of Document Processing

Processing Step	Input Processing	Output Processing	Target Application (Examples)
Scanning	Reading scanned document batch	Writing classified documents in the requested format as batch	Separating scanned batches in single documents, classifying and then saving them
Receiving faxes	Reading input faxes	Writing faxes after reading in the requested format	Forwarding and saving faxes
Sending faxes	Reading documents	Writing documents in fax format	Faxing out documents with a facsimile device
Importing spools	Reading spools (print output)	Writing classified documents in the requested format as batch	Separating imported spools in single documents, classifying and then saving them
Receiving e-mails	Reading e-mails	Writing e-mails after reading in the requested format	Archiving received e-mails
Sending e-mails	Reading documents to be sent	Writing documents to be sent in the requested format	Sending documents per e-mail
Printing	Reading spools	Writing documents after reading either directly in the printing format or in AFP format	Redirecting spool documents to printers and/or printing system with other format requirements (e.g. printing PDF documents with an AFP printer)
Saving application documents in a format not supported by the application	Reading documents by a printer driver	Writing documents in the requested format	Building Word documents as TIFF files with a printer driver to be able to archive them
Converting	Reading of the document batch to be converted	Writing document batches after converting	Joining documents or converting them in another format

Even though the processing functions vary, they all more or less follow the same logic:

- During input processing, a page, document batch or single document with a certain input format is read, and if applicable structured in batches or documents.
- During output processing, a document batch or a single document is written in the same format or any requested other output format.

The DocBridge products that execute the processing units listed in *Table 2: Input and Output Functions of Document Processing on page 2* use for all input and output functions the MFF (Mixed Format Filter) modules. Each MFF filter is specialized in processing of a certain format: e.g. PDF, AFP, or a raster image. These filters can read and write documents of one of these formats. The input as well as the output processing functions use of the same common layer - the presentation area (PA): The input processing functions of a MFF filter read the source format into the PA and the output processing functions write the target format out of the PA. Thus, the DocBridge products are able to combine the input processing functions of any MFF filter with the output processing functions of any other MFF filter. For example, the product DocBridge Application Renderer reads a TIFF formatted document with the MFF filter MFFTIF (MFF filter for TIFF format) and writes the document in PDF format with the MFF filter (MFFPDF for PDF format).

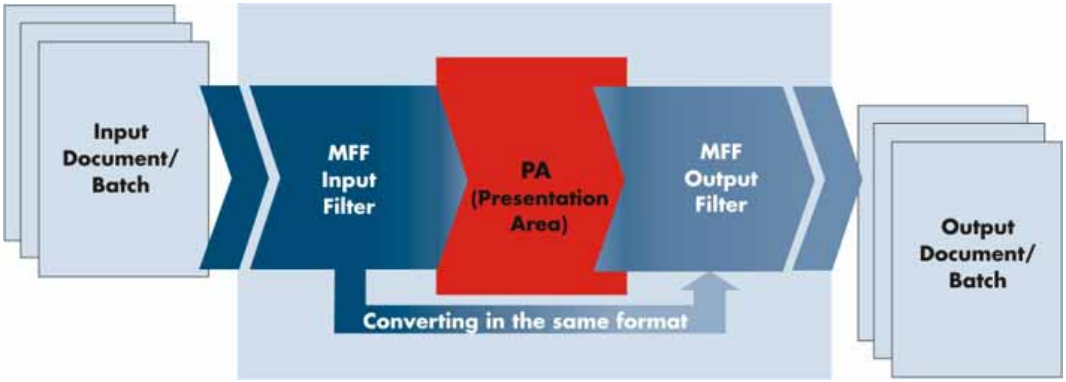


Figure 1: Input and Output Processing via MFF Filter and Presentation Area

The presentation area (PA) completely describes one page of a document. All objects of one page (text in Unicode, raster images, lines, rectangles etc.), also user defined code pages and special characters (in the areas of mixed object formats like AFP and PDF) are saved in the PA in a unique format, independent of their input and output.

- The input filters enable the reading of a single page of a document or the document batch in the PA. During this process, the format specific objects are converted to generic objects of the PA. The various object types (text, raster images, lines, rectangles etc.) remain unchanged.
- The output filters write the various objects out of the PA page by page in the requested output format. Therefore, at the end, the complete document and/or document batch will be present in the target format. Depending on the output format used, the respective object type will be transferred into the related target format. If the input format is a mixed object format (e.g. PDF) and the target format is a pure raster format (e.g. TIFF), the input object will be rasterized as PA raster image and converted in the requested raster image format (in the mentioned example with TIFF as output format the PDF text will be transferred in a TIFF raster image).

1.1.2. MFF Filter Overview

Due to the separation of the input and output functions, it is possible to develop an adequate MFF filter for each supported format. Depending on the format, they are use as input filter and as output filters.

In the following chapters, summarize the filters and their functions. If not otherwise specified, the filters are available as both output and input filter.

Input \ Output	AFP	PDF & PDF/A	PCL 5 & HPGL	PCL 6	PostScript	VPS	Linemode ASCII/EBCDIC	Data File	Metacode/ DJDE	IPDS	LIPDS	XPS	XML	HTML**	SVG	Raster Format ***	PC Printer
AFP	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
PDF & PDF/A	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
PCL 5 & HPGL	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
PCL 6	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
PostScript	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
PPML	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
VIPP*	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
VPS	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Linemode ASCII/EBCDIC	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Data File	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
LCDS/DJDE	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Metacode/DJDE	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
PRESCRIBE	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
XPS	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
XML	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
HTML/CSS	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
XSL-FO	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
SAP ALF + OTF	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
RTF	✔	✔	✔	✔	✔	✔			✔	✔	✔	✔	✔	✔	✔	✔	✔
SVG	✔	✔	✔	✔	✔	✔			✔	✔	✔	✔	✔	✔	✔	✔	✔
Raster Format***	✔	✔	✔	✔	✔	✔			✔	✔	✔	✔	✔	✔	✔	✔	✔
PC Documents	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

* With restrictions ** With format related restrictions *** Support of the raster formats: BMP, IOCA, JPEG, GIF, PCX, PNG, TGA, HD Photo, JPEG 2000 and TIFF

Figure 2: Compart Conversion Matrix

Note: For a complete list of MFF input and output filters, visit www.compart.net and search with keyword ‘conversion matrix’.

1.1.2.1. MFFAFP: AFP and MO:DCA

The MFF filter MFFAFP supports the slightly different IBM formats AFP (Advanced Function Printing) and MO:DCA (Mixed Object:Document Content Architecture). Because of the market significance of AFP in the large customer area, strong emphasis focuses on conversion fidelity and depth. The filter is designed as input and output filter and supports the following features:

- Inline and external resources as well as ACIF compatible resource libraries
- Raster fonts and related code pages
- Overlays, page segments and medium maps
- Pantone and true colors (e.g. JPEG images)
- TLE and NOP processing for archives and printing systems (e.g. POSI)

1.1.2.2. MFFGOF: SAPGOF

The GOF format used by SAP (GOF = Generic Output Format) includes the OTF format (Output Text Format) and the SAP list format. Since in this case only conversions of GOF documents into other formats are relevant, the MFF filter MFFGOF is only designed as input filter. Because of the market penetration of GOF in the large customer area, strong emphasis focuses on conversion fidelity and depth. The filter supports the following features:

- Intelligent font mapping
- Raster images

1.1.2.3. MFFIFF: Raster Images

The MFF filter MFFIFF supports as an input and output filter common raster image formats like BMP, GIF, IOCA, JPEG, PCX, TGA and TIFF. In particular, the TIFF generation is verified for almost all known archives. The filter supports the following features:

- High performance scaling during reading process
- Optimized scale to gray function
- Resource saving I/O functions

1.1.2.4. MFFIJP: IJPDS

The MFF filter MFFIJP is designed as output filter and supports the Kodak Versamark (Scitex) IJPDS format (Ink Jet Printer Data Stream). The filter supports the following features:

- ICC color profiles
- Spot color and full color mode
- High volume printout, approx. speed of 150 m/min. duplex printing and full color mode

1.1.2.5. MFFIPD: IPDS

The MFF filter MFFIPD is designed as output filter for the IPDS protocol (Intelligent Printer Data Stream) of IBM. The filter supports the following features:

- Printer communication via TPC/IP and SCSI
- Printer support major manufacturers' hardware: IBM, Nipson, Océ, and Xerox
- Print output resolution is independent of input data stream resolution

1.1.2.6. MFFLCD: LCDS

The MFF filter MFFLCD is designed as input filter for the LCDS/DJDE format of Xerox. The filter supports the following features:

- Use of original resources
- Form/image replacement with other file types, e.g. PDF
- Support of formatting commands (DJDE/JSL)
- Support of Xerox EBCDIC code page

1.1.2.7. MFFPCL: PCL

The MFF filter MFFPCL is designed as input filter and output filter and supports the format PCL (Printer Command Language). The filter supports the following features:

- Intelligent font mapping
- Macro processing
- Download fonts

1.1.2.8. MFFPDF: PDF

The MFF filter MFFPDF supports as input filter and output filter the PDF format (Portable Document Format) including PDF/A. Because of its market penetration, a main focus is on conversion fidelity and depth. The filter supports the following features:

- Optimized compression
- Web format (optimized)
- Password protection with encryption
- Font mapping versus font inclusion for size optimizing
- Harmonization of AFP functions like e.g. logos as fonts

1.1.2.9. MFFPOS: PostScript

The MFF filter MFFPOS supports as input filter and output filter the PostScript format. The PostScript format describes the appearance of text, graphics, and images on printed or displayed pages. The filter supports the following features:

- Embedding of type 1 and type 3 fonts
- Tray control
- Font size optimizing via referencing for multiple occurrence of graphical elements

1.1.2.10. MFFPRT: Printer Output

The MFF filter MFFPRT supports a document output to any of the supported formats directly to the printer. Therefore, it is only an output filter. The filter supports the following features:

- Available for Windows
- Tray control
- Printer margin (problems with PC printers unlike host printers)
- Automatic scaling and rotation

1.1.2.11. MFFSVG: SVG of W3C

The MFF filter MFFSVG is designed as input filter and output filter and supports the vector format SVG (Scalable Vector Graphics) defined by the W3C International Community.

1.1.2.12. MFFTXT: ASCII/EBCDIC Linemode Format

This MFF filter MFFTXT is designed as input filter and output filter and supports the conversion of ASCII and EBCDIC Linemode formats. The filter supports the following features:

- Fix and variable record length
- Free code page definition
- Processing of control channels - carriage control and font index byte

1.1.2.13. MFFXFF: XFF Format

The MFF filter MFFXFF supports the Compart specific format XFF (Extensible File Format), which is used as intermediary format, e.g. in DocBridge Pilot. It is primarily used, if the file size should be relatively small and rapidly accessible.

1.1.2.14. MFFXFO: XSL-FO

The MFF filter MFFXFO is designed as input filter and supports XSL-FO (Extensible Style Language Formatting Objects) as an input format. The filter supports the following features:

- Font Mapping
- Processing of documents converted or preformatted with any MFF filter
- Compart's own XSL-FO functional extensions

1.1.2.15. MFFXIF: Compart XML Format

The MFF filter MFFXIF is designed as input filter and output filter and supports the Compart specific XML format XIF (Extended Interchange format) and is used e.g. for requirements related to persistence or serialization.

1.1.2.16. MFFXRX: Metacode

The MFF filter MFFXRX is designed as input filter and output filter for the Metacode/DJDE format of Xerox. The filter supports the following features:

- Image replacement with other file types, e.g. PDF
- Reading and writing of offline files
- Support of Unicode mapping

1.1.3. MFF Manager

The MFF Manager is an integrated part of all DocBridge products. It manages and identifies the respective formats and handles the fine tuned configuration of the related filter, e.g. entries for font mapping and customer code pages.

1.2. Functions of the DocBridge Products

The DocBridge products are based on the described above PA concept and the separated MFF filter. Part of each of these products is the MFF Manager and a batch-processing module, which manages the input and output document batches.

The following Compart products for the general document processing functions are available:

DocBridge Mill Program for separating, collecting, classifying, indexing and format converting of documents and document batches or spool output of different sources, e.g. for storage, archiving, printing, or analyzing and processing.

DocBridge Notes Simple and fast conversion program for all Lotus Notes documents including attachments for long-term archiving.

DocBridge Pilot Program for print pool, production and postage optimization, bundling preparation and multi-channel output. It is a complete solution for configuring and combining print data streams deriving from various input resources.

DocBridge View Viewer for documents or document batches. It displays in one window all document formats supported by a MFF filter.

DocBridge Application Renderer
A Compart printer driver for DocBridge products converting PC documents (e.g. Microsoft Word .DOC files) to application independent formats (e.g. TIFF, PDF, AFP etc.).

DocBridge Mill Toolkit
Development toolkit for applications which convert and process documents of an input format on-the-fly or in batch mode into a desired output format.

1.3. DocBridge Mill Program Modules

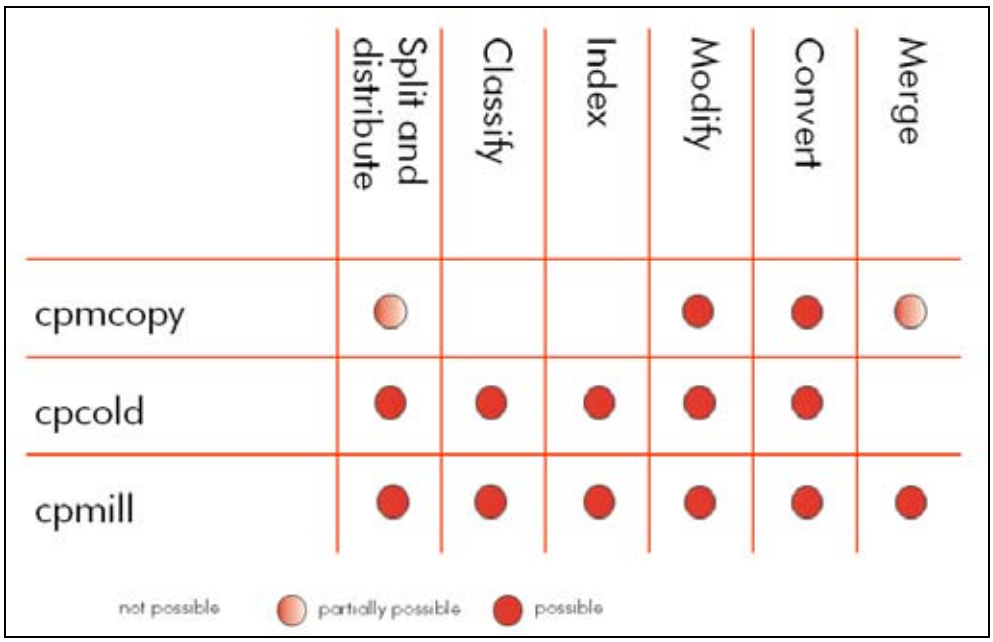


Figure 3: DocBridge Mill Program Modules Overview

1.3.1. CpCOLD

Computer **O**utput on **L**aser **D**isc (COLD) is a term that derives from the early days of digital archiving. CpCOLD is capable of separating large spool files into several documents. CpCOLD is a reliable module characterized by its flexibility and its productive efficiency. The functional range focuses on the controllability of generated output files.

Overview of CpCOLD functionalities:

- Conversion
- Modification
- Indexing
- Classification
- Splitting and distribution

For more information, see *chapter 3.1. CpCOLD Command Line Parameters and Variables on page 20*.

1.3.2. cpmcopy

cpmcopy is the command line versions of DocBridge Mill. It focuses mainly on format conversion without any other further processing. Nevertheless, more complex processing steps can be executed, like “stamping” watermarks, extracting document pages or merging documents.

Overview of cpmcopy functionalities:

- Merging
- Conversion
- Modification
- Splitting and distribution

For more information, see *chapter 3.2. cpmcopy Command Line Parameters on page 26*.

1.3.3. cpmill

cpmill is a file based module of DocBridge Mill that can copy, modify and convert documents. The module’s focus is the processing flexibility of the input data. Application specific programs can be integrated in a flexible and structured way.

Overview of cpmill functionalities:

- Merging
- Conversion
- Modification
- Indexing
- Classification
- Splitting and distribution

For more information, see *chapter 3.2. cpmcopy Command Line Parameters on page 26*.

1.4. Functions of DocBridge Mill

Because of its document processing functionality, this member of the DocBridge family was named *Mill*: Like in a mill, the incoming material *Digitalized Input Documents* is parted during a fragmentation process into its internal objects and is converted depending on filter profiles and other settings into a requested outgoing material *Digital Output Documents*.

The following chapters describe which form of digitalized input documents is processed by DocBridge Mill, which processing functions are available in detail and which form of outgoing material, the digitalized output documents, could be obtained.

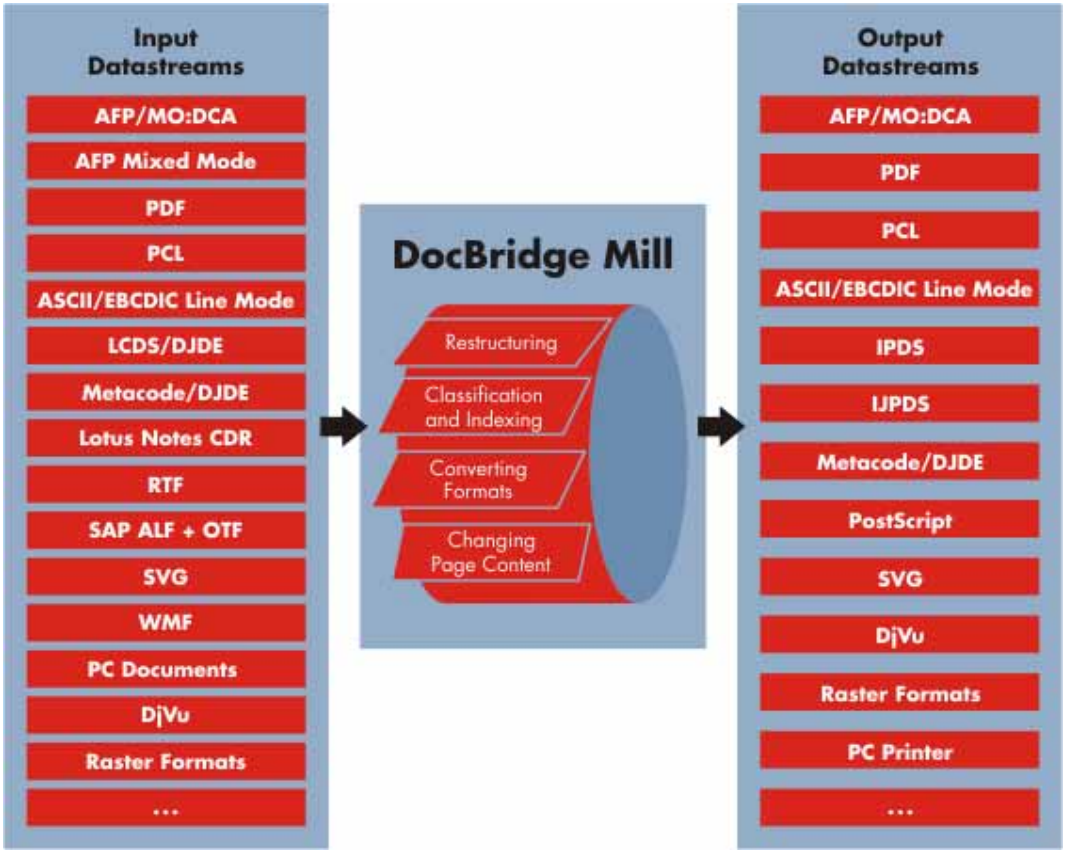


Figure 4: Functional Overview of DocBridge Mill

1.4.1. Supported Document Batch Types

DocBridge Mill can process as input as well as output the following document batch types:

- Single documents
- Datastream oriented batches

In the case that several documents bundled in a document batch have to be processed, the documents refer to each other. Page groups or TLEs can be specified as reference for document batches that are built from single documents in a folder or as subsequent pages of a single file either in an external batch file or inside a document data stream by additional structure information like in case of an AFP datastream.

1.4.1.1. Single Documents

The simplest way of document handling by DocBridge Mill is the processing of single documents, e.g. to convert documents. In this case, no specifications for document batch types are required.

1.4.1.2. Datastream Oriented Document Batch

For datasstream oriented document batches the structure information is contained in the document datasstream itself, i.e. in this case several documents are sequentially joined together in one file. The joints between the documents, i.e. between the last page of a document and the first page of the following document, are identified by internal structure information.

1.4.2. Processing Functions and Applications

DocBridge Mill offers for the processing of document batches (incl. single documents) the following jointly usable document processing functions:

- Reorganizing documents
- Classifying and indexing
- Converting formats
- Modifying page information

The following chapters describe applications for these document processing functions.

1.4.2.1. Reorganizing Documents

To reorganize documents the following functions are available:

- Separating spools in single documents
- Distributing documents according to predefined criteria (e.g. according to zip code areas)
- Merging documents or several batches to one batch
- Filtering documents or pages according to predefined criteria (e.g. according to payment forms)

1.4.2.2. Classifying and Indexing

To classify the incoming documents e.g. for the transfer to an archive (that means to assign it to an index class or an archive) and to provide them with indexes, the following attributes can be analyzed:

- Attributes contained in the data stream like NOPs or TLEs (in the case of AFP)
- General attributes like page numbers (e.g. page groups in AFP) or overlay names
- Text strings that can be gripped at determinable area
- Terms that match predefined search items
- Values or text strings in raster image documents that can be extracted via barcode recognition

With these functions, DocBridge Mill is a fully functional COLD product for archiving spool outputs of the supported formats – but goes far beyond these primary functions.

1.4.2.3. Converting Format

DocBridge Mill can convert document batches of the following formats into another one of these formats:

- Mixed Object formats: AFP, ASCII-/EBCDIC line mode (only input format), MO:DCA, PCL, PDF, Print Output (only output format), SAP GOF (OTF and ALF) (only input format) a. o.
- Raster formats: BMP, GIF, IOCA, JPEG, PCX, PNG, TGA, TIFF a. o. with the following additional functions: rasterizing the pages with a Scale-To-Gray function, and reducing color to black/white images.

A prerequisite for the requested conversion is the licensing of the necessary input and output format filter. More details about the conversion fidelity and restrictions of the supported input and output filters are described in *chapter 1.1.1. Presentation Area (PA) and Mixed Format Filter (MFF) on page 2.*

1.4.2.4. Page Modification

With DocBridge Mill, you can change or correct the following page information on the pages of the document batches to be processed:

- Recognizing the main page orientation and automatic rotation of misaligned pages
- Changing the page size and displacing the page content
- Masking text strings
- Adding text strings and OMR marks

1.4.2.5. Analog and Binary Copying

In the context of terms like input and output processing, MFF filters, and Presentation Area (PA), analog and binary copying must be explained:

- Analog Copying: It is always processed through the PA. The conversion into any available format is possible. The processing is based on input and output filters (which may not be different). Each document page is read separately to the PA and written separately out of the PA. The above-mentioned modifications can be performed.
- Binary Copying: The copy process is performed without a conversion to another format. Document pages are not read into the PA. Only format specific structures are used and copied to generate a new page.

1.4.3. Program Flow Functions

To control, rework, and trouble shooting the following functions are available:

- Logging appropriate for auditing of all program operations
- Optionally deleting or moving (copying) of the processed documents
- Starting redo functions (e.g. activating printing jobs)
- Moving erroneous documents in an error folder
- Trigger functions in case of errors (e.g. SMS transmission via external program)
- Extensive trace control

2. DocBridge Mill Installation

The DocBridge Mill application does not require a specific installation procedure due to its availability for many operation systems. This means that during a Windows installation no registry entries are generated. You merely copy the shipped files to a specific directory. DocBridge Mill must have access to the necessary resources (fonts, code pages, etc.). Therefore, you must specify resource paths in the profile files of the MFF filters.

A program run **without** profile files functions only in a limited way, i.e. font files may not be localized. The profile files for the input filter MFFXFO and the appropriate output format, for example MFFAFP, must be stored in the current program directory. Detailed information about profile files, see *chapter 5.3. MFF Filter Profiles on page 223*.

For a basic DocBridge Mill installation, you copy the following files into a directory:

- Program file (cpmill.exe)
- Valid license file (cpmill.lic)
- Fonts for document processing. If you do not copy any fonts in the directory, the TrueType font Arial (arial.ttf) is used as default.

Note: You can specify an environment variable. If specified, the application finds the license file in the specified path. If no environment variable is specified, the license file must be stored in the program directory.

Example: DocBridge Mill environment variable

```
set CPLICENSEPATH=c:\compart\license
```

Example: DocBridge Mill directory structure

Compart recommends to use the directory structure displayed in *Figure* 5 to arrange the resource files, programs, input and files etc. in separate folders.

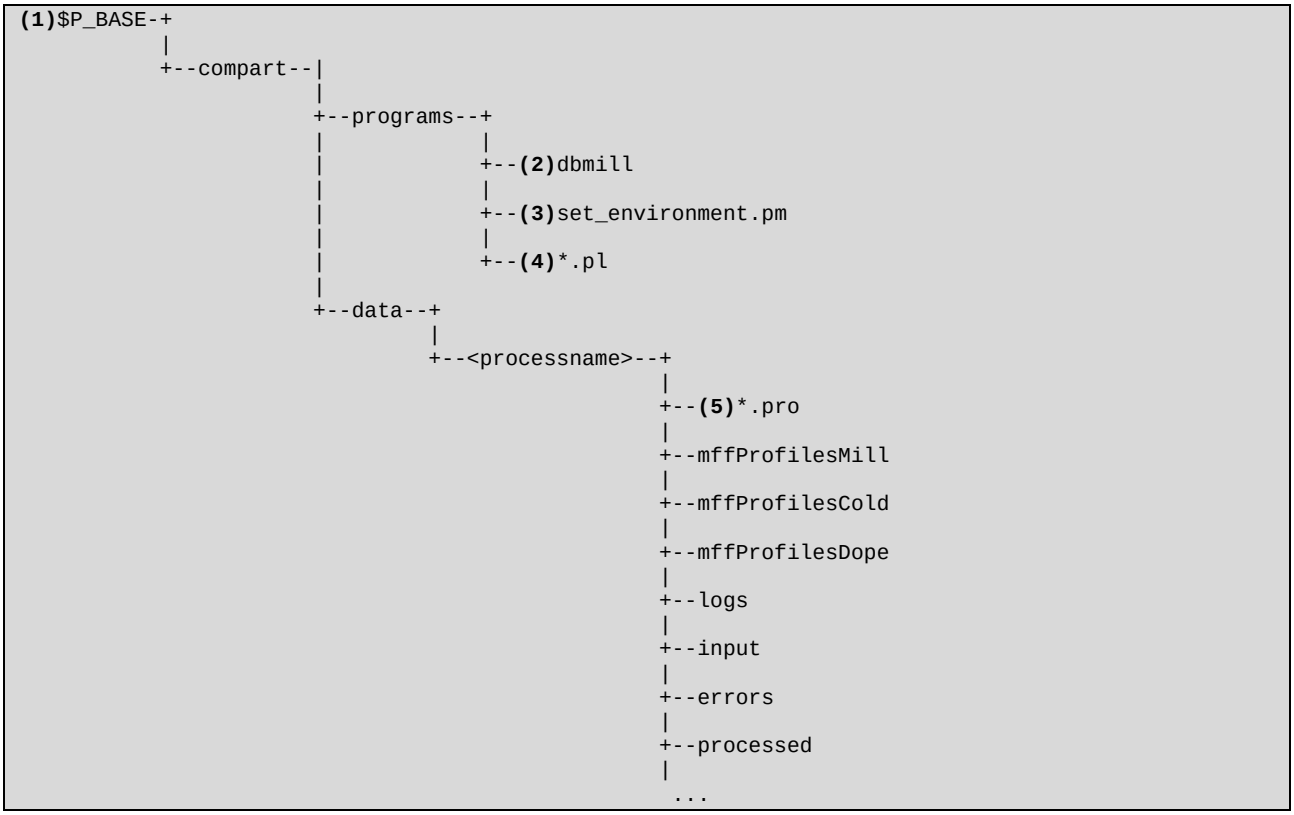


Figure 5: DocBridge Mill Directory Structure

Notes:

- **(1)**: Directory $P_BASE is a user-defined root directory
- **(2)**: Directory contains amongst others the executable programs, e.g. DocBridge Mill
- **(3)**: Perl module that can be called internally by the Perl scripts. Normally no modifications are necessary. The process runs independently of the position of $P_BASE in the structure. If necessary, it can be modified to meet the requirements.
- **(4)**: Perl start-up script that starts the process. If necessary, the application profile must be stored here.
- **(5)**: cpmill or CpCOLD application profile

2.1. Supported Operating Systems

Because all DocBridge products and filters are written in a code that is based on an operating system independent library, a wide range of operating systems can be supported:

- Microsoft Windows 2000, XP, and Server 2003
- Mac OS X
- x86 Linux
- FreeBSD
- AIX
- Sun Solaris
- HP-UX
- z/OS
- zLinux
- z/OS UNIX System Services

3. DocBridge Mill Command Line Parameters

DocBridge Mill consists of three program parts:

- The document management processor **CpCOLD** that covers the areas of reorganizing, classifying, and indexing of the documents,
- **cpmcopy** processing includes converting, modifying page information, separation, and distribution of documents.
- **cpmill** processing includes converting, modifying page information and indexing of documents.

The following chapters describe program parts and its parameters in detail.

3.1. CpCOLD Command Line Parameters and Variables

CpCOLD is developed to process any document format for which an input filter exists within the Compart MFF architecture. Processing means separating the input files in single documents according to selectable criteria (i.e. also no separation), generating index files for subsequent archiving systems, inserting of index or control information directly into the output files for subsequent printing systems, and the simultaneous conversion of the input files in any format supported by an output filter inside the Compart MFF architecture. If the input files are ‚converted' in the same format as the input format, it is possible to copy any single page identically (binary copy).

During the processing single pages can be suppressed according to almost any criteria. That is e.g. useful in a print spool when empty pages are inserted (for duplex printing) which do not have to be displayed in an archiving system.

3.1.1. Call and Parameters

CpCOLD can be called in the following form:

```
cpcold [-parameter] profile
```

Syntax Notes

- Parameters can be coded in lowercase or uppercase (e.g. `-D` or `-d`)
- Running in UNIX systems parameters should be prefaced by `'-'` and in Windows operating systems by `'-'` or `'/'`.

The specification of the profile is mandatory; the specification of the parameters are optional. The controlling of the CpCOLD processing is mainly done by the profile, i.e. calling CpCOLD without parameters is the normal case.

Table 3: CpCOLD Command Line Parameters

Parameter	Description
`-Dname=value`	Defined variables are valid for the whole operating time of CpCOLD. The definition is done in the way: `name=value`. No blanks are allowed inside of `name` or `value` and before or after the equal sign.
`-Ename=value`	Definition of profile entity for MFF profiles (can be specified multiply)
`-Hworkdir`	Home (working directory)
`-Iinputfile`	File is used as spool file (overrides input parsing)
`-Lvalue`	Protocol level specified in the profile can be overwritten. Protocol level: `0`: Errors `1`: Errors and warnings (default) `2`: Errors, warnings, and information `3`: Errors, warnings, information and trace information Default is 1. **Note:** The screen output of CpCOLD is controlled by `stdout` according level 2 and cannot be suppressed by parameter. If the screen output is unwanted, `stdout` should be redirected to a file or to `nul`.
`-M`	Available memory can be tested - no processing!
`-PdtdPath`	Path for DTD file. Default is current directory.

Table 3: CpCOLD Command Line Parameters	
Parameter	**Description**
`-Rlogfilename`	Name of the default log file can be changed. CpCOLD normally generates a log file with the current date as name. The location of the log file is specified in the profile. `-R` overwrites profile entry.
`-Snum`	Specified number of pages of the input data stream can be skipped. **Note:** Profile developer should use this parameter only.
`-Ttracedir`	Directory for trace output
`-Vname=value`	Variable definition for MFF profiles (may be used more than once)
`-Xname=value`	Definition of external profile entity for MFF profiles (may be used more than once)

3.1.2. Principle of the CpCOLD Processing

CpCOLD is looking for spools to be processed in one or more specified folders. This can be controlled by definable masks. As specified in the profile the folder will be scanned one or more times. The search interval (offset between 2 searches) can be specified for any folder.

Each page will be scanned by means of a list of specified strings (one or more words) (Eye catcher) and with any hit the content of the corresponding text item will be saved in a variable (`<searchlist>`).

Thereafter the page will be scanned for text items, which are located at a specified position. The value (content) of the text located in this area is also copied into a variable.

For example, with AFP input files the names of all overlays used in this page, the name of the page group, and its consecutive number are saved in corresponding variables. Subsequently all filter rules defined in the profile are analyzed and the page will be suppressed if applicable.

If a page passes the filter, CpCOLD checks for a *docstart* condition. The *docstart* condition can be specified in the profile similar to the filter condition almost without restrictions. The start of a new document effects the evaluation of a document class according to this page and the specified rules. Any number of different classes for a spool can be specified.

One document class defines a list of indexes. Its values are extracted from the corresponding first page. These index lists are written in a specific format to the export file. Currently, the following formats are available:

- Fixed Record File with configurable record layout
- Free definable attribute file
- XML file
- Easy Archive import file
- FileNET import file
- ISIS Web Archive Format
- IXOS import file

Any page which passes the filter and which does not activate a *docstart* condition, is appended as a following page to the current document. This is done in the same (multipage) or a separate (single-page) document file.

If a spool is processed error free, CpCOLD copies it in a so-called archive folder, if this is specified in the profile. In an error occurred, the spool is copied into an error folder, if this is specified in the profile. Thereafter CpCOLD deletes the spool in the original folder. The spool is also deleted in the original folder, if no archive or error folder is specified.

3.1.3. CpCOLD Variables

The special control of the processing is done by rules (formulas), which can record the processing status via variables. Variables can be specified by the user in the profile and via the parameter `-D`.

In addition, CpCOLD defines several system variables, which can be used without predetermination (inside of formulas).

The definitions of variables in the profile or with the call, which have the same name as system-defined variables, are not allowed. In this case, the result is not determined and should only be used by an experienced user.

3.1.3.1. System Defined CpCOLD Variables

CpCOLD uses the system-defined variables listed in the table below.

Table 4: System Defined CpCOLD Variables

Variable	Description
$ARCDIR	Folder where successfully processed spools are written
$CLASS	Current name of the class in process
$COPYGROUP	Copy group name of current page
$CR	Carriage return; hex `'0x0d'`
$CURRLINE	Content of the current line (only in LINET Mode)
$CURRLINENR	Current line number (only in LINET Mode)
$DOCDIR	Folder where document(s) are written
$DOCEXTENSION	File extension for document file
$DOCFILE	Current name of the document file
$DOCNR	Number of current document (1-based)
$DOCNRSTR	Number of current document as string (1-based)
$DOCPAGE	Page number of currently processed page of the document (1-based)
$DOCRESDIR	Folder for CpCOLD's external resources
$DOCRESFILE	Resource file (in `$DOCRESDIR`) used by CpCOLD for processing
$ERRDIR	Folder for erroneously processed spools
$EXPDIR	Folder for generated export file
$EXPFILE	Current name of export file
$EXTRACTRESFILE	External resource extracted from document
$FN3BUNDLENR	Bundle number of export type FILENET3
$FN3HEADER	Header for export type FILENET3
$FN3TOCFILE	Content file of export type FILENET3
$FN3TOCINDEXNAME	Index name for content file for export type FILENET3
$GROUPNR	Current sequence of copy group (1-based)
$GROUPPAGE	Current page number of copy group (1-based)
$GROUPPAGES	Total page number of copy group (1-based)
$HASCOLOR	Colored items in document page
$HASGRAY	Grayscale items in document page

Table 4: System Defined CpCOLD Variables	
Variable	**Description**
$IMMDONE	IMM (Invoke Medium Map) Structured Field before current page
$IXOS_CMDFILE	Command file name for archive format type IXOS
$IXOS_COMMANDS	Command file content for archive format type IXOS
$IXOS_LOGFILE	Log file name for archive format type IXOS
$JOURNALDIR	Journal file folder
$LOGDIR	Log file folder
$LOGLEVEL	Currently specified log level. CpCOLD offers 4 levels: 0: Error 1: Warning 2: Process flow information 3: Trace (debug) information
$METADOCCOUNT	Archive format type IXOS: Document count for meta documents
$NEWLINE	Carriage return and line feed
$NEWOVERLAY	Set to value 1, if overlay of current page has changed in relation to previous page (otherwise: 0)
$NEWPAGEGROUP	Set to value 1, if page group of current page has changed in relation to previous page (otherwise: 0)
$ORIENTATION	Page orientation
$OVERLAY	Name of overlay in currently processed page
$PAGEANNOTATION	Page annotation
$PAGECOMMENT	Page comment
$PAGEEMPTY	1, if current page is empty, otherwise 0
$PAGEGROUP	Name of page group of currently processed page
$PAGEGROUPINDEX	Name of page group index of currently processed page (0-based)
$PAGESWRITTEN	Number of written pages
$PAGETLE	Varpool for page TLEs
$PATHSEP	Path separator
$POS_capturename.X	Variable name: prefix $POS + *capturename* + X for x value
$POS_capturename.Y	Variable name: prefix $POS + *capturename* + Y for y value
$PROFILE	Name of current profile
$QUOTATIONMARK	Character "
$RESOLUTION	Raster resolution in dpi
$RESPAGE_DPI.RX	Contains resolution in x-direction (set only by capture function)
$RESPAGE_DPI.RY	Contains resolution in y-direction (set only by capture function)
$RETCODE	Return code
$SIMPLEXDUPLEX	Value for simplex/duplex printing
$SINGLEQUOTE	Character '
$SPOOLAUTHOR*	Author
$SPOOLCREATIONDATE*	Creation date
$SPOOLDIR	Folder for spool files
$SPOOLFILE	Current name of spool file
$SPOOLKEYWORD*	Keyword
$SPOOLMASK	Mask to identify files in spool folder
$SPOOLMODDATE*	Modification date
$SPOOLNR	Number of current spool
$SPOOLPAGE	Page number of currently processed page of spool (1-based)
$SPOOLPAGECOUNT	Page count of current spool file (1-based)
$SPOOLPRODUCER*	Producer
$SPOOLRESFILE	External resource file provided together with the spool
$SPOOLSIZE	File size of current spool in byte

Table 4: System Defined CpCOLD Variables	
Variable	**Description**
$SPOOLSUBJECT*	Subject
$SPOOLTITLE*	Title
$TLE	Varpool for TLEs
$TRAY	Name of input tray
NAME	TLE name
VALUE	TLE value
* Variables can only be used for the PDF output format.	

3.1.3.2. Implicit Defined CpCOLD Variables

Some document formats allow the definition of indexes or analogue constructs. By specification of a corresponding attribute in the profile of CpCOLD it is possible, to adopt these definitions as variables in the context of the profile, i.e. by references. The table below lists the attributes.

Table 5: Implicit Defined CpCOLD Variables	
Variable	**Description**
USETLE	This attribute is only valid for AFP format and has the effect that any TLE with name and value is available as variable in the profile. The corresponding variables are determined before `<varlist>` and can be referenced there.
USECOMMENT	This attribute is valid for AFP and SAP formats (OTF and LIST). For AFP documents this is the content of NOP tags, for SAP documents this is the content of header lines, which starts with ‚*'. The specified content is written in the system variable `$PAGECOMMENT`. `$PAGECOMMENT` is an array variable and does not provide a name for the corresponding value of the variable. Referencing is done by the index in the array (0-based). The corresponding variable is specified before `<varlist>` and can be referenced there.
USEANNOTATION	This attribute is only valid for PDF format. The resulting content is written in the system variable `$PAGEANNOTATION`. `$PAGEANNOTATION` is an array variable and does not provide a name for the corresponding value of the variable. The corresponding variable is determined before `<varlist>` and can be referenced there.

3.1.3.3. Profile Defined CpCOLD Variables

In the profile variables can be defined by the construct `name="varname"` and be referenced elsewhere. Presently variables can be defined inside the tags `<searchlist>`, `<capturelist>` and `<varlist>`.

The variables are determined in the following sequence:

After reading a page, the implicit defined variables (see above) are determined. Thereafter the variables of `<searchlist>` and immediately following the variables of the `<capturelist>` are determined.

The variables of the `<varlist>` are calculated after `<searchlist>` and `<capturelist>`. That means that all variables determined in `<searchlist>` and `<capturelist>` can directly be referenced in `<varlist>`. Furthermore, any variable, which is defined in `<var>`, can reference its direct predecessor.

3.1.3.4. Call Defined CpCOLD Variables

Variables can be specified inside the program call by the parameter `-D`. These variables are evaluated during the program start and are valid for the whole running time of CpCOLD. A profile defined variable with the same name overwrites the value of the call defined variable.

3.2. cpmcopy Command Line Parameters

Normally cpmcopy is called as follows:

```
cpmcopy -infile inputfile1 [inputfile2...] -outfile output file
```

or

```
cpmcopy -infile inputfilemask1 [inputfilemask2...] -outfile -type output type
```

The first command has the effect that the `inputfiles` are copied into the file `outputfile`. Thereby the type of the output file is derived from the file extension. This does not work with raster (image) formats.

In the second case all files `inputfilemask1` are copied in files of type `outputtype`, where the file name of the output file is the same as the file name of the input file, with the exception that the file extension is derived from the output type.

The second call has the advantage that also wildcards can be used for the source files.

The parameter and files, which are specified after `-infile`, are always jointly regarded. That means that all parameters till the following `-infile` or `-outfile` are applied to all files specified there. In addition, many of these input groups can be specified:

```
cpmcopy -infile inputfilemask1 [-infile inputfilemask2...]
        -outfile -type output type
```

Notes:

- Some parameters can be specified for input as well as for output processing. But input parameters are processed first. Thus, the results of the two statements below are different.

```
cpmcopy -infile -dx 5mm -outfile –rotate 90
cpmcopy -infile rotate -outfile –dx 5mm
```

- The parameters can be abbreviated, but they must be unique. Compart recommends **not** to abbreviate parameters in a productive environment. Reason: Parameters added with new versions might cause uniqueness problems.

When processing starts, cpmcopy needs access to the available resources (fonts, code pages, etc.). The resources must be specified in the profile files of the MFF filters.

The global, input, and output parameters are listed in the table below. The input parameters are coded after the global parameter `-infile` and the output parameters after the global parameter `-outfile`.

Table 6: cpmcopy – Global Command Line Parameters	
addhiddentext	lognofile
bigjob	logtime
codepage	maxthreads
converttoimage	msg.add
copies	msg.remove
copycomments	modulepath
cs.abort	outfile
cs.autostart	profiledir
cs.client	profileentity.external
cs.clientlogdir	profileentity.external
cs.clientlogfile	profilevar
cs.port	quiet
cs.retries	reduceimagecolors
cs.server	relaxed
cs.shutdown	remove
cs.start	render
flipimage	renderonlymonochrome
fontentries	repeatloops
ignoreerrors	sighandling.disable
infile	stdin
invertimage	stdout
licfile	stdoutdirect
logappend	strict
cs.retries	suppresscopycomments
logdate	tracefile
logdir	tracelevel
logfile	uselocale
loglevel	verbose

Table 7: cpmcopy – Input Command Line Parameters	
bwc	germanorthography.new
bwd	gamma
bwf	gentoc
bwj	imagedepth
bws	imagetype
bwt	jobname
cdr.attachment.errorhandlingmode	jsl
cdr.attachment.processmode	macrodir
cdr.hideabmps	mmap
charsetpath	mergetext
concatenatetext	mmd.pagedef
convertattachments	modifyoverlays
copyinfo	obfuscatetext
copysecurity	page
disablefdp	pcl.fullprintablearea
dx	pwi
dy	rastx
embedfilespath	rasty
enablefdp	repeat
fo.autopagegroup	reslibin
fo.fmprocessingmodeswap	reverseduplex
fo.halfleadingfactor	rotate
fo.resourcepath.images	suppresscopyinfo
fdp.pass.through	systemfontpath
fdp.viewmode	tray
formdef	truetypepath
formdefpath	type
gendocumentstructure	

Table 8: cpmcopy – Output Command Line Parameters	
addpagesforduplex	pagesize
afp.coloroutput	pagewidth
afp.generateoverlays	pcl.fullprintablearea
afp.writenods	pdf.accessibility.notallowed
author	pdf.assemble.notallowed
autoformdef	pdf.copy.notallowed
blackwhiteadjust	pdf.digitalcopy.notallowed
charsetpath	pdf.fill.notallowed
codepage	pdf.nochange
coloroutput	pdf.pagescalingfactor
combinefonts	pdf.print.notallowed
combineimages	pdf.textnotes.nochange
completeduplex	pdf.writeuncompressedstreams
converttoicc	ppml
cx	ppml.flushpagecount
cxmax	printtofile
cy	prt.scalemode
cymax	pwo
deldup	pwu
dir	rastx
documentcopies	rasty
dx	reducex
dy	reducey
embedfonts	replacepatterns
findpatterns	reslibin
formdef	revision
gamma	rgbgrayascmyk
genthumbnails	rgbgrayasgray
grayascmyk	rotate
iff.extractimages	rx
ifnotsetbyinput	ry
imagedepth	scaletogray
imageresforce	sd
imageresx	selfcontained
imageresy	separateoverlays
imagetype	shiftrect
jobname	shiftrect
keylength	splitdelta
keywords	splitmask
linearized	splitmaskoffset
macrodir	splitnamedcolorimages
mapcolortoname	stamp
mergefonts	subject
mergetext	suppressemtpypages
modifyoverlays	title
nobinarycopy	tray
nobinarystamp	truetypepath

Table 8: cpmcopy – Output Command Line Parameters	
pageheight	type
pagescalingfactor	watermark
obfuscatetext	

3.2.1. cpmcopy Parameter Description

The parameters are listed alphabetically.

addhiddentext	
Parameter	`-addhiddentext`
Description	The parameter `addhiddentext` is always used in combination with parameter `converttoimage`. When processing is completed by parameter `converttoimage`, text is treated as hidden text. Text is made searchable.
Example	`cpmcopy -converttoimage -addhiddentext -i file.jpg -o -type pcl`

addpagesforduplex	
Parameter	`-addpagesforduplex`
Description	When the parameter is specified, documents with an odd number of pages are provided with an additional empty end page. If you want to generate a back page for each front page, you can use the parameter `completeduplex`.
Example	`cpmcopy -i file.pdf -o -type afp -addpagesforduplex`

afp.coloroutput	
Parameter	`-afp.coloroutput`
Description	AFP specific parameter. Colored content is presented with color in the AFP output file.
Example	`cpmcopy -i file.pdf -o -type afp -afp.coloroutput`

afp.generateoverlays	
Parameter	`-afp.generateoverlays`
Description	AFP specific parameter. If you specify `afp.generateoverlays`, AFP overlays (resource objects) are generated. Additionally, you can specify `selfcontained` to embed the resources into the output file.
Example	`cpmcopy -i file.pdf -o -type afp -afp.generateoverlays -selfcontained`

afp.writenods	
Parameter	`-afp.writenods`
Description	AFP specific parameter. No DS byte is written into AFP output file.
Example	`cpmcopy -i file.pdf -o -type afp -afp.writenods`

author	
Parameter	`-author <value>`
Description	PDF specific parameter. The author information for the PDF metadata can be specified.
Example	The author is displayed in the Document Properties window. `cpmcopy -i file.afp -o file.pdf -author John Doe` 

autoformdef	
Parameter	`-autoformdef`
Description	The parameter activates automatic form definition generation for AFP files. Form definitions can be generated with the following information: tray, number of copies, simplex/duplex, etc.
Example	`cpmcopy -i file.pdf -o file.afp -autoformdef`

bigjob	
Parameter	`-bigjob`
Description	Files are processed sequentially. When program starts, the existence of the input files is not checked.
Example	`cpmcopy -bigjob -i *.afp -o -type pdf`

blackwhiteadjust	
Parameter	`-blackwhiteadjust`
Description	The parameter calculates the gray value. The first specified value is the limiting value for black, the second one separated by a comma is the limiting value for white. If the gray value is greater than the limiting value for black, the element color is black. If the gray value is less than the limiting value for white, the element color is white. The first or the second parameter can be omitted, e.g.: ,3.
Example	`cpmcopy -i file.pdf -o file.xrx -blackwhiteadjust 80,20`

bwc	
Parameter	`-bwc <value>`
Description	Clustered dithering with a value for the matrix size. Valid values are 2 to 8. If no value is specified, the default value is 3.
Example	`cpmcopy -i file.jpg -bwc -o -type iff.gif`

bwd	
Parameter	`-bwd <value>`
Description	Dispersed dithering with a value for the matrix size. If no value is specified, the default value is 4.
Example	`cpmcopy -i file.jpg -bwd -o -type iff.gif`

bwf	
Parameter	`-bwf value`
Description	Floyd-Steinberg dithering. The algorithm checks a small neighborhood of four pixels surrounding the central pixel, and distributes the error in a defined distribution key.
Example	`cpmcopy -i file.jpg -bwf -o -type iff.gif`

bwj	
Parameter	`-bwj`
Description	Jarvis-Judice-Ninke dithering. It uses a larger pixel neighborhood to optimize the error distribution.
Example	`cpmcopy -i file.jpg -bwj -o -type iff.gif`

bws	
Parameter	`-bws`
Description	Stucki dithering. Like Jarvis-Judice-Ninke it uses a larger pixel neighborhood than Floyd-Steinberg to optimize the error distribution.
Example	`cpmcopy -i file.jpg -bws -o -type iff.gif`

bwt	
Parameter	`-bwt <value>`
Description	Threshold dithering with specification of a threshold percent value (0 - 100). If no value is specified, the default value is 50.
Example	`cpmcopy -i file.jpg -bwt -o -type iff.gif`

cdr.attachment.errorhandlingmode	
Parameter	`-cdr.attachment.errorhandlingmode <value>`
Description	IBM Lotus Notes CDR specific parameter. Error handling mode for attachments of CD record files. Options are: ▪ `abort`: conversion is aborted ▪ `ignore-if-not-exist`: conversion is executed without attachment
Example	`cpmcopy -i file.cdr -cdr.attachment.errorhandlingmode ignore-if-not-exist -o file.pdf`

cdr.attachment.processmode	
Parameter	`-cdr.attachment.processmode <value>`
Description	IBM Lotus Notes CDR specific parameter. Processing mode for attachments of CD record files. Options are: ▪ `nop`: attachment is not converted ▪ `inplace`: attachment within text ▪ `append`: attachment after text ▪ `only`: only attachment is converted
Example	`cpmcopy -i file.cdr -cdr.attachment.processmode inplace -o file.pdf`

cdr.hideabmps	
Parameter	`-cdr.hideabmps`
Description	IBM Lotus Notes CDR specific parameter. The parameter hides attachments icons of CD-Record files.
Example	`cpmcopy -i file.cdr -cdr.hideabmps -o file.pdf`

charsetpath	
Parameter	`-charsetpath <value>`
Description	The parameter specifies the directory for the fonts.
Example	`cpmcopy -charsetpath fonts -i file.afp -o file.pdf`

codepage	
Parameter	`-codepage <value>`
Description	The parameter specifies program-internal the code page. Currently, the Compart internal code page number can be used only. For a list of Compart code pages, see *Appendix B: Code Page Names on page 277*.

Example	Example: Windows code page 1252:
	`cpmcopy -codepage 1252 -i *.afp -o -type pdf`
	Example: German EBCDIC code page 273 (OS/390 or z/OS only):
	`cpmcopy -codepage 273 -i *.afp -o -type pdf`

coloroutput	
Parameter	`-coloroutput`
Description	PCL specific parameter. Colored content is presented with color in the PCL output file.
Example	`cpmcopy -i file.jpg -o file.pcl -coloroutput`

combinefonts	
Parameter	`-combinefonts`
Description	You specified the parameter to convert fonts used in the input file to bit-mapped fonts.
Example	`cpmcopy -i file.afp -o file.pdf -combinefonts`

combineimages	
Parameter	`-combineimages`
Description	You specified the parameter to generate an output file to avoid rounding errors. The file contains images that are positioned close together.
Example	`cpmcopy -i input.xif -o output.xif -combineimages`

completeduplex	
Parameter	`-completeduplex`
Description	You specified the parameter for duplex printing. Each front page is provided with a back page. If a document shall have an even number of pages when printed duplex, you can specify `addpagesfor-duplex`.
Example	`cpmcopy -i file.afp -o file.pdf -completeduplex`

concatenatetext	
Parameter	`-concatenatetext`
Description	You specify the parameter to concatenate text elements, provided that it can be done without any losses.
Example	`cpmcopy -file.afp -concatenatetext -o file.pdf`

convertattachments	
Parameter	`-convertattachments`
Description	You specify the parameter for PDF input files (PDF 1.6 or higher) only. The attachments are converted and appended to the output file.
Example	`cpmcopy -i file.pdf -convertattachments -o file.afp`

converttoicc	
Parameter	`-converttoicc <file>`
Description	All colors are processed with the specified ICC profile to harmonize colors. Depending on the output format you have to specify that a colored output is generated, see `afp.coloroutput`. **Note:** The functionality is not related to any specific MFF filter, i.e. specified paths for resources set in a profile file do not have any impact.
Example	`cpmcopy -i file.jpg -o file.pdf -converttoicc AdobeRGB1998.icc`

converttoimage	
Parameter	`-converttoimage`
Description	The parameter converts pages into images, if this is not done by default. Note: Vector fonts may not be rendered in any cases.
Example	`cpmcopy -converttoimage -i file.jpg -o -type iff.bmp`

copies	
Parameter	`-copies <number>`
Description	Specific AFP and PCL parameter for input files. For each page of the input file the specified number of copies is generated in the output file, see also parameter `autoformdef`.
Example	

copycomments	
Parameter	`-copycomments`
Description	You specify the parameter to copy comments that are part of the input file. **Note:** It is not guaranteed that the output file accepts the same number of comments as specified in the input file. For example, in TIFF files only one comment is allowed.
Example	`cpmcopy -copycomments -i file.tif -o file.afp`

copyinfo	
Parameter	`-copyinfo`
Description	PDF specific parameter. During binary copying of PDF files metadata like title, author, etc. are copied to the output file.
Example	`cpmcopy -i filea.pdf -copyinfo -o fileb.pdf`

copysecurity	
Parameter	`-copysecurity`
Description	PDF specific parameter. You specify the parameter to copy the security settings.
Example	`cpmcopy -i filea.pdf -copysecurity -o fileb.pdf`

cs.abort	
Parameter	`-cs.abort`
Description	When client-server processing is used, you can specify the parameter to terminate the server processing. All running jobs are terminated, and the cpmcopy server processes are terminated. An error is signaled to the cpmcopy client process that is connected.
Example	`cpmcopy -cs.server -cs.abort -i file.pdf -o file.txt`

cs.autostart	
Parameter	`-cs.autostart`
Description	If client-server processing is used, you can specify the parameter at the client side to start the server processing.
Example	`cpmcopy -cs.client -cs.autostart -i file.pdf -o file.txt`

cs.client	
Parameter	`-cs.client`
Description	If client-server processing is used, the parameter indicates that it is a client process. **Note:** The server and client process must run on the same system. By default, the client starts a communication with the server using hostname 127.0.0.1 and Port 7001.
Example	`cpmcopy -cs.client -cs.port 7001 -i *.pdf -o -type txt`

cs.clientlogdir	
Parameter	`-cs.clientlogdir <path>`
Description	You specify the parameters to write additional information to the file `mcopy.log` about the files defined by the parameters.
Example	`cpmcopy -cs.client -cs.clientlogdir log -cs.clientlogfile clientlog.log -i file.txt -o file.pdf`

cs.clientlogfile	
Parameter	`-cs.clientlogfile <file>`
Description	You specify the parameters to write additional information to the file `mcopy.log` about the files defined by the parameters.
Example	`cpmcopy -cs.client -cs.clientlogdir log -cs.clientlogfile clientlog.log -i file.txt -o file. pdf`

cs.port	
Parameter	`-cs.port <port>`
Description	If client-server processing is used, you use the parameter to specify the server port number to establish the communication with the server. By default, the client starts a communication with the server using hostname 127.0.0.1 and Port 7001.
Example	`cpmcopy -cs.client -cs.port 7001 -i *.pdf -o -type txt`

cs.retries	
Parameter	`-cs.retries <xxx>`
Description	If client-server processing is used, you can specify the parameter to start more than one (default) attempt to establish the communication with the server.
Example	`cpmcopy -cs.client -cs.port 7001 cs.retries 3 -i *.pdf -o -type txt`

cs.server	
Parameter	`-cs.server`
Description	If client-server processing is used, you can specify the parameter to start cpmcopy at the server side. When started, the program waits for processing requests sent by the client.
Example	`cpmcopy -cs.server -cs.start`

cs.shutdown	
Parameter	`-cs.shutdown`
Description	If client-server processing is used, you can specify the parameter to notify the server to stop the processing after all ongoing processes are completed.
Example	`cpmcopy -cs.client -cs.shutdown`

cs.start	
Parameter	`-cs.start`
Description	If client-server processing is used, you specify the parameter to start the server.
Example	`cpmcopy -cs.server -cs.start`

cx	
Parameter	`-cx <value>`
Description	Output image size in x-direction specified as pixel value. You can use the parameter to generate thumbnails and its size and resolution.
Example	`cpmcopy -i file.jpg -o file.gif -cx 45 -cxmax 50 -cy 30 -cymax 35`

cxmax	
Parameter	`-cxmax <value>`
Description	Maximum output image size in x-direction specified as pixel value. You can use the parameter to generate thumbnails and its size and resolution.
Example	`cpmcopy -i file.jpg -o file.gif -cx 45 -cxmax 50 -cy 30 -cymax 35`

cy	
Parameter	`-cy <value>`
Description	Output image size in y-direction specified as pixel value. You can use the parameter to generate thumbnails and its size and resolution.
Example	`cpmcopy -i file.jpg -o file.gif -cx 45 -cxmax 50 -cy 30 -cymax 35`

cymax	
Parameter	`-cymax <value>`
Description	Maximum output image size in y-direction specified as pixel value. You can use the parameter to generate thumbnails and its size and resolution.
Example	`cpmcopy -i file.jpg -o file.gif -cx 45 -cxmax 50 -cy 30 -cymax 35`

deldup	
Parameter	`-deldup`
Description	You specify the parameter to identify duplicate page elements and to remove them.
Example	`cpmcopy -i file.afp -o -type pdf -deldup`

dir	
Parameter	`-dir <dir>`
Description	If you do not specify an output file name for the conversion process, the file name will be composed of the input file name and the output file type. By default, the output file is stored in the directory of the input file. To change the directory you can use a relative (current directory) or an absolute path.

Example	Relative path for output directory:
	`cpmcopy -i *.afp -o -type pdf -dir pdfout`
	Absolute path for output directory:
	`cpmcopy -i *.afp -o -type pdf -dir C:\temp\pdfout`

disablefdp	
Parameter	`-disablefdp`
Description	The parameter deactivates the form definition processing. If the parameter `enablefdp` is specified, the parameter `disablefdp` will be ignored.
Example	`cpmcopy -i file.afp -disablefdp -o -type afp`

documentcopies	
Parameter	`-documentcopies`
Description	PCL specific parameter. The PCL datastream contains one copy of the document, but the specified number of copies will be printed provided that the printer supports the multi-copy function.
Example	`cpmcopy -i file.pdf -o file.pcl -documentcopies 3`

dx	
Parameter	`-dx <length>`
Description	You can use the parameter to shift page contents in x-direction.
Example	`cpmcopy -i -filein.pdf -dx 2cm -dy 2cm -o -fileout.pdf`

dy	
Parameter	`-dy <length>`
Description	You can use the parameter to shift page contents in y-direction.
Example	`cpmcopy -i -filein.pdf -dx 2cm -dy 2cm -o -fileout.pdf`

embedfilespath	
Parameter	`-embedfilespath <dir>`
Description	You can use the parameter to embed a directory path to relocated resource files.
Example	

embedfonts	
Parameter	`-embedfonts`
Description	PDF specific parameter. You can use the parameter to embed fonts into the output file (PDF) according to the specifications of the profile file. By default, fonts are not embedded.
Example	`cpmcopy -i file.afp -o file.pdf -embedfonts`

enablefdp	
Parameter	`-enablefdp`
Description	The parameter activates the form definition processing with the Formdef Processor, see also parameter `disablefdp`.
Example	`cpmcopy -i file.afp -enablefdp -o -type afp`

fdp.pass.through	
Parameter	`-fdp.pass.through`
Description	The parameter activates the form definition processing with the Formdef Processor using the passthrough mode.
Example	`cpmcopy -i file.afp -fdp.pass.through -o -type afp`

fdp.viewmode	
Parameter	`-fdp.viewmode`
Description	When the Formdef Processor is activated, see parameter `enablefdp`, medium map names can be identified in document pages.
Example	

findpatterns	
Parameter	`-findpatterns`
Description	You use the parameter to identify patterns in an image. If a pattern is identified, the image is converted to a rectangle with patterns. The parameter can be used in combination with the parameter `replacepattern` to optimize visual results.
Example	`cpmcopy -i input.afp -o gray.ps -findpatterns -replacepattern`

flipimage	
Parameter	`-flipimage`
Description	You use the parameter to flip images: vertically (`v`), horizontally (`h`), or vertically and horizontally (`vh`).
Example	`cpmcopy -i file.jpg -o flippedimage.png -flipimage v -nobinarycopy`

fo.autopagegroup	
Parameter	`-fo.autopagegroup`
Description	XSL-FO input and AFP output specific parameter. When you use the parameter, a page group is generated for each page-sequence.
Example	`cpmcopy -i file.fo -fo.autopagegroup -o file.afp`

fo.fmprocessingmodeswap	
Parameter	`-fo.fmprocessingmodeswap`
Description	XSL-FO input parameter. The parameter activates the formatter's swap mode. It results in a performance improvement and a reduction of memory usage. This is especially true for files that contain nested tables that run over many pages.
Example	`cpmcopy -i file.fo -fo.fmprocessingmodeswap -o file.pdf`

fo.halfleadingfactor	
Parameter	`-fo.halfleadingfactor`
Description	XSL-FO input parameter. XSL-FO input specific parameter. The value specified with the parameter is a multiple of the font size. The following formula applies: line height = font size x half leading factor. Example: If a piece of text is 12pt high and the line-height value is 14pt, 2pts of extra space should be added: 1pt above and 1pt below the characters. The parameter can be used when adjusting the layout for a document that is generated with XSL-FO formatters other than the Compart formatter. **Note:** Leading is the vertical spacing between lines of type. The leading value divided in half is the half-leading. This value will be applied above and below.
Example	`cpmcopy -i file.fo -fo.halfleadingfactor 1.1 -o file.pdf`

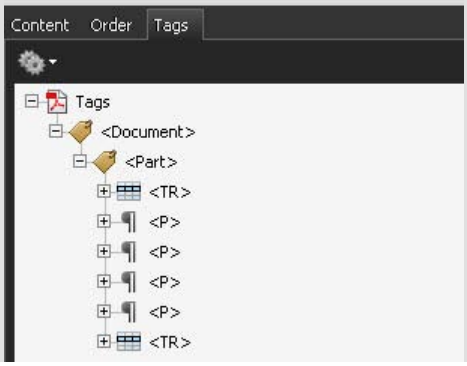
fo.resourcepath.images	
Parameter	`-fo.resourcepath.images`
Description	XSL-FO input specific parameter. You use the parameter to specify paths for images that are references in XSL-FO files. The image paths in the XSL-FO file are used as relative paths. When you specify for example `fo.resourcepath.images images`, and you use the image names without any further path specifications, the images are expected to be stored in the subdirectory `images` of the program directory.
Example	`cpmcopy -i file.fo -fo.resourcepath.images images –o file.pdf`

fontentries	
Parameter	`-fontentries <fontentriesfile>`
Description	You use the parameter to write the font information of the input files to a file with a user-defined name.
Example	`cpmcopy -fontentries fonts.xml -i *.afp -o -type pdf`

formdef	
Parameter	`-formdef <file>`
Description	The parameter specifies a form definition. If specified for an input file, the form definition overwrites the definition of `defaultformdef` in a profile file, e.g. `mffafp.pro`. The parameter `enablefdp` is activated automatically. If you specify a form definition for an output file, it can be written to the output file itself, see also the parameter `selfcontained`.
Example	`cpmcopy -i file.afp -formdef F1A10111 -o file.pdf`

formdefpath	
Parameter	`-formdefpath <xxx>`
Description	The parameter specifies a directory path where form definitions are stored.
Example	`cpmcopy -i file.afp -formdef F1A10111 -formdefpath formdefs -o file.pdf`

gamma	
Parameter	`-gamma`
Description	The parameter specifies values for gamma correction and controls thereby the brightness of an image. Valid values are: 1.5, 2.2, 0.5, etc.
Example	`cpmcopy -i input.pcl -o gray.pdf -gamma 1.5`

gendocumentstructure	
Parameter	`-gendocumentstructure`
Description	XSL-FO input and PDF output specifc parameter to generate Tagged PDF files. When you specify the parameter, the attributes `role` of the XSL-FO file are read and written as tags in the PDF file. Additionally, you must specify the element `<autotagging value="TRUE"/>` in the profile file mffpdf.pro.
Example	`cpmcopy -i documentwithroles.fo -gendocumentstructure –o taggedpdf.pdf` The tags are displayed in the Tag Navigation window of your PDF Viewer. 

genthumbnails	
Parameter	`-genthumbnails`
Description	PDF specific parameter. The parameter generates thumbnails in the output file.
Example	`cpmcopy -i file.afp -o -file.pdf -genthumbnails`

gentoc	
Parameter	`-gentoc`
Description	Specific parameter for AFP and PDF output. The parameter generates bookmarks out of TLEs specified in the input file to be used in the output file.
Example	`cpmcopy -i file.afp -gentoc -o file.pdf`

germanorthography.new	
Parameter	`-germanorthography.new`
Description	XSL-FO input specific parameter. When you specify the parameter, the hyphenation process is based on the new German orthography.
Example	`cpmcopy -i file.fo -germanorthography.new -o file.pdf`

grayascmyk	
Parameter	`-grayascmyk`
Description	Colors defined as grayscale values are converted to a CMYK color where the black channel is specified appropriately.
Example	`cpmcopy -i file.afp -o file.pdf -grayascmyk`

iff.extractimages	
Parameter	`-iff.extractimages`
Description	Specific parameter for IFF output. Extract images to a multi-page file. The output type must be a multi-page image format, e.g. `iff.tif`.
Example	`cpmcopy -i file.pdf -o extractedimages.tif -iff.extractimages`

ifnotsetbyinput	
Parameter	`-ifnotsetbyinput <parameter>` (version 2009009)
Description	You must always specify the parameter in combination with other settings. It is used if the input file does not contain the very settings itself. Currently, the parameter is used together with the parameter `sd`. The parameter `sd` is only specified in this combination, if the input file does not contain any simplex or duplex settings. If you use PCL as an output format, the following settings are required in the MFF profile file `mffpcl.pro`: ▪ `<writesimplexduplex value="asinput"/>` ▪ `<duplexcontrol value="undefined"/>`
Example	`cpmcopy -i *.pdf -o -type pcl -ifnotsetbyinput -sd simplex`

ignoreerrors	
Parameter	`-ignoreerrors`
Description	You use the parameter to complete process run in spite of errors.
Example	`cpmcopy -ignoreerrors -i *.afp -o -type pos`

imagedepth	
Parameter	`-imagedepth <xxx>`
Description	You can use the parameter to specify the color depth in bits: ▪ `1`: Monochrome, `8`: Grayscale, Palette, `24`: RGB, `32`: CMYK
Example	`cpmcopy -i file.jpg -o file.png -imagedepth 24`

imageresforce	
Parameter	`-imageresforce`
Description	You can use the parameter to enforce the specified resolution. It includes an upscaling, which cannot be done just by using the parameters `imageresx` and `imageresy`.
Example	`cpmcopy -i file.jpg -o file.png -imageresx 700 -imageresy 700 -imageresforce`

imageresx	
Parameter	`-imageresx`
Description	You can use the parameter to specify the image resolution in x-direction. By default, the resolution value of the input file is used. Upscaling is not supported. For upscaling you have to use the parameter `imageresforce`.
Example	`cpmcopy -i file.jpg -o file.png -imageresx 50 -imageresy 50`

imageresy	
Parameter	`-imageresy`
Description	You can use the parameter to specify the image resolution in y-direction. By default, the resolution value of the input file is used. Upscaling is not supported. For upscaling you have to use the parameter `imageresforce`.
Example	`cpmcopy -i file.jpg -o file.png -imageresx 50 -imageresy 50`

imagetype	
Parameter	`-imagetype <type>`
Description	The parameter specifies the image type, e.g.: rgb, cmyk, grayscale, palette
Example	`cpmcopy -i file.jpg -o file.png -imagetype rgb`

infile	
Parameter	`-infile <file>`
Description	The parameter `infile` required for running conversion processes with cpmcopy. One or more files can be specified as input files as follows: ▪ file name list: `cpmcopy -i filea.pcl fileb.pcl -o -type afp` ▪ regular expression: `cpmcopy -i *.pcl -o -type afp` ▪ input file directory: `FOR %C IN (afpin\*.afp) DO cpmcopy -infile %C –outfile -type PDF pdfout\%~nC.pdf – loglevel v` In this case all AFP files in the directory `afpin` are converted to PDF files in the directory `pdfout`. **Note:** When cpmcopy is called out of a Windows batch file, the percent sign must be coded twice. It is possible to specify several `-infile` groups: `cpmcopy -i filea.pdf -i fileb.pdf -o -type pcl` Thus, different input parameters can be specified for input files.

Example	`cpmcopy -i *.pdf -o singleoutputfile.afp`
	To prevent storage problems with an input file list you can write the file list into a file. Then, cpmcopy can process this file.
	▪ Generation and processing an input file list: Unix
	`find dir -name "*.pdf" > inputfiles.txt`
	`cpmcopy -i @inputfiles.txt -o singleoutputfile.afp rm inputfiles.txt`
	▪ Generation and processing an input file list: Windows
	`dir /b *.pdf > inputfiles.txt cpmcopy -i @inputfiles.txt -o singleoutputfile.afp del inputfiles.txt`

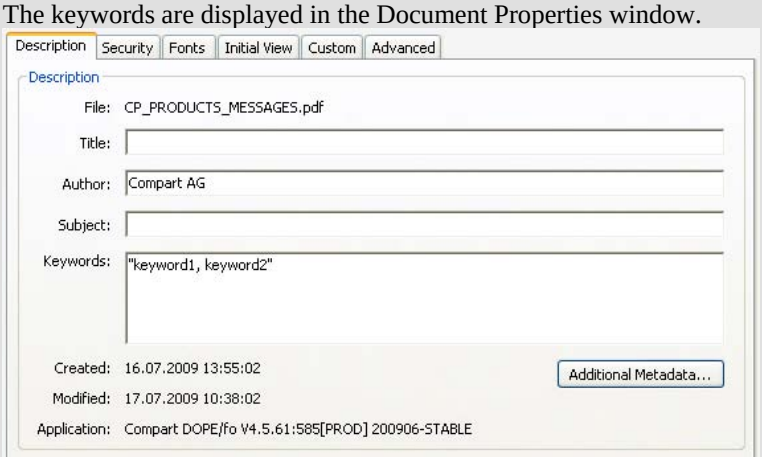
invertimage	
Parameter	`-invertimage <xxx>`
Description	You can use the parameter to generate inverted images of the input images.
Example	`cpmcopy -i file.jpg -o invertedfile.png -invertimage -nobinarycopy`

jobname	
Parameter	`-jobname <xxx>`
Description	Xerox specific parameter. The parameter specifies the name of the print job (PRT). The parameter overwrites the attribute value in the profile files `mfflcd.pro` or `mffxrx.pro`.
Example	

jsl	
Parameter	`-jsl <jslname>`
Description	Xerox specific parameter. The parameter specifies the name of the JSL file. The parameter overwrites the attribute value in the profile files `mfflcd.pro` or `mffxrx.pro`.
Example	

keepduplex	
Parameter	`-keepduplex`
Description	The parameter controls that the duplex settings for pages of the input file are available for the output file, i.e. it is made sure that front page remains front page and back page remains back page.
Example	`cpmcopy -i file.pdf -keepduplex -o file.afp`

keylength	
Parameter	`-keylength <number>`
Description	PDF specific parameter. The parameter specifies the key length (in bits) for the encryption of PDF files. Valid values are 40 and 128. The parameter can be specified in combination with `pdf.assemble.notallowed` and `pdf.accessibility.notallowed` to set the encryption.
Example	`cpmcopy -i file.afp -o file.pdf` `cpmcopy -i file.afp -o file.pdf -pdf.accessibility.notallowed -keylength 128`

keywords	
Parameter	`-keywords <xxx>`
Description	PDF specific parameter. You can use the parameter to add keywords as metadata to the output file.
Example	`cpmcopy -i file.afp -o file.pdf -keywords "keyword1, keyword2"` The keywords are displayed in the Document Properties window. 

licfile	
Parameter	`-licfile <file>`
Description	You can use the parameter to specify a user-defined name for the license file. By default, one of the license files, mcopy.lic, cpmill.lic, or cpsdk.lic, is used in the profile directory. The profile directory is identical with the program directory, unless you specify another directory with the parameter `modulepath`.
Example	`cpmcopy -licfile cpmilla.lic -i *.pdf -o -type pcl`

linearized	
Parameter	`-linearized`
Description	PDF specific parameter. The parameter generates a 'linearized', i.e. fast web view enabled file. Using the browser plug-in of a PDF viewer the first pages of a PDF file can be displayed before the complete document is loaded from a web server. The default is "Not Linearized".
Example	The example copies the first two pages of all AFP files are copied to a fast web view enabled PDF file: `cpmcopy -i *.afp -page ..2 -o webout.pdf -linearized`

logappend	
Parameter	`-logappend`
Description	You specify the parameter to add log messages to an existing log file. Already existing log files are not overwritten. The default is that a new log file is written and it overwrites an existing log file.
Example	`cpmcopy -logappend -i *.afp -o -type pdf`

logdate	
Parameter	`-logdate`

Description	By default, log messages are written without any date and time stamp. You specify the parameter to add a date stamp to each log message. Additionally, you can use the parameter `logtime` to add a time stamp. Log message with date stamp:
	`2009-06-05 APP1000I Compart DocBridge Mill/cpmcopy - Mixed File Format Copy`
	`2009-06-05 APP1001I Release: 200904`

Example	`cpmcopy -logfile mylogfile.log -logdate -i *.afp -o -type pdf`

logdir	
Parameter	`-logdir <path>`
Description	By default, cpmcopy writes the log file into the program directory. **Note:** When you specify a log directory with the parameter, the log file is stored in the log directory and the trace file in the root directory after the first cpmcopy run. After all further cpmcopy runs, when you have specified the parameter, the log file, and the trace file are stored in the log directory.
Example	`cpmcopy -logdir log -i *.afp -o -type pdf`

logfile	
Parameter	`-logfile <file>`
Description	By default, log messages are written in a file with the name `mcopy.log`. You use the parameter to specify a user-defined log file name.
Example	`cpmcopy -logfile mylogfile.log -i *.afp -o -type pdf`

loglevel	
Parameter	`-loglevel <level>`
Description	You use the parameter to specify a severity level for messages. Additionally, all messages that are more restrictive are issued. Following severity levels are available: `F` (Fatal), `E` (Error), `W` (Warning), `I` (Info), `V` (Verbose), and `D` (Debug)
Example	`cpmcopy -loglevel i -i *.afp -o -type pdf`

lognofile	
Parameter	`-lognofile`
Description	By default, cpmcopy generates a log file. You specify the parameter with the result that log file is written.
Example	`cpmcopy -lognofile -i *.afp -o -type pdf`

logtime	
Parameter	`-logtime`

Description	By default, log messages are written without any date and time stamp. You specify the parameter to add a time stamp to each log message. Additionally, you can use the parameter `logdate` to add a date stamp. Log message with a time stamp:
	`15:40:47 APP1000I Compart DocBridge Mill/cpmcopy - Mixed File Format Copy`
	`15:40:47 APP1001I Release: 200904`
	Log message with date and time stamp:
	`2009-06-05 15:39:15 APP1000I Compart DocBridge Mill/cpmcopy – Mixed...`
	`2009-06-05 15:39:15 APP1001I Release: 200904`

Example	`cpmcopy -logfile mylogfile.log -logdate -logtime -i *.afp -o -type pdf`

macrodir	
Parameter	`-macrodir <path>`
Description	The parameter generates the directory where macros are stored. By default, the current directory is used.
Example	

mapcolortoname	
Parameter	`-mapcolortoname`
Description	The parameter specifies color definitions that can be mapped to a name. The three values can be specified separated by commas. 1. color angle 2. treshold angle 3. color name to be mapped to the color **Note:** Currently, only palette images are supported.
Example	`cpmcopy -i file.pdf -o -mapcolortoname 0,20,red -mapcolortoname 240,20,blue -type xif`

maxthreads	
Parameter	`-maxthreads <value>`
Description	Using the client/server mode the parameter value specifies the number of processes running parallel that is executed by the server process. The maximum performance rate depends on several determining factors. **Note:** Compart does not give any overall statement about the value of the parameter `maxthreads`. Generally, the value corresponds to the number of available CPU cores. You can do tests with lifelike data to get the setting results. Alternatively, you can reduce the available CPU performance for the server process to get a sufficient performance rate for the process running parallel.
Example	`cpmcopy -maxthreads 100 -i *.afp -o -type pdf`

mergefonts	
Parameter	`-mergefonts`
Description	You specify the parameter to reduce the number fonts used in a page merging overlapping font subsets into a font.
Example	`cpmcopy -i file.afp -o file.pdf -mergefonts`

mergetext	
Parameter	`-mergetext`
Description	Text items are merged based on defined conditions, The values are as follows: `mergetext [t|h][c][w|l][s][m][p]`. None of the arguments is required alone, but the string may not be empty. If one of the values existing of the pairs `t`,`h` and `w`,`l` is placed in the same string after specifying the other one, the last specified value will be overwritten. `t`: Merge in items order `h`: Merge in character reading order. It is default; if nothing else (t or h) is given. `c`: Preserve text color. If not specified, color converted to monochrome. `w`: Merge according word boundaries. Formatting results are improved. `l`: Make text items as long as possible. It is default; if nothing else (w or l) is given. `s`: Preserve character spacing. Formatting results are improved. `m`: Reduce leading and trailing blanks `p`: Insert space between words
Example	`cpmcopy -i file.afp -o file.pdf -mergetext tcwm`

mmap	
Parameter	`-mmap`
Description	You use the parameter to open input files as memory mapped files. It has the great advantage that you get a better reading data performance. The parameter must be specified for each group of input files (FileInGroup). **Note:** When you use this access method be sure that your system provides enough memory. Compart recommends to test this parameter in a non-production environment.
Example	`cpmcopy -i input.pdf -mmap -o output.pdf`

mmd.pagedef	
Parameter	`-mmd.pagedef <file>`
Description	AFP specific parameter. Page definition file for Mix Mode data (MMD)

modifyoverlays	
Parameter	`-modifyoverlays`
Description	Page modification that normally has an impact on pages only can be applied to overlays recursively.
Example	`cpmcopy -i input.afp -remove t -modifyoverlays -o output_without_text.pdf`

modulepath	
Parameter	`-modulepath`
Description	You can use the parameter to specify a user-defined directory for MFF profile files, if you do not want to manage them within the program directory.
Example	`cpmcopy -modulepath profiles -i file.afp -o file.pdf`

msg.add	
Parameter	`-msg.add <xxxnnnn>`

Description	You can use the parameter to extent the list of log messages that cause a program termination. For example, if you do not want a file to be converted if a font replacement is required, you can add the message number PDI2005 to the list, see also *Compart Products Reference Guide for Messages and Codes*. The reason for the program termination is issued.
	`PDI2005W Font 'Courier New' (weight 'MEDIUM' style 'UPRIGHT' replaced with 'Default font')`
	`MCF1237F Error 'PDI2005' occurred in logfile`

Example	`cpmcopy -msg.add PDI2005 -i file.afp -o file.txt`

msg.remove	
Parameter	`-msg.add <xxxnnnn>`
Description	You can use the parameter to remove a message number from the list of log messages that caused a program termination.
Example	`cpmcopy -msg.remove AFP1010 -i file.afp -o file.txt`

nobinarycopy	
Parameter	`-nobinarycopy`
Description	You specify the parameter to make sure that not a binary copy is generated.
Example	`cpmcopy -i file.pdf -o filecopy.pdf -nobinarycopy -mergetext tcwm`

nobinarystamp	
Parameter	`-nobinarystamp`
Description	If you specify the parameter, a stamp or watermark is not binary copied to be part of the output file. The stamp or watermark is read and written as normal.
Example	`cpmcopy -i filein.pdf -nobinarystamp -o fileout.pdf`

obfuscatetext	
Parameter	`-obfuscatetext <method>`
Description	You specify the parameter to make a document text illegible. The layout remains as it is. The following algorithms are available: `rot13`: ROT13 encryption `random`: randomly changing of characters `random.with.digits`: changing characters and digits
Example	`cpmcopy -i file.pdf -o obfuscatedfile.pdf -obfuscatetext random.with.digits`

outfile	
Parameter	`-outfile <file>`
Description	The parameter `outfile` is required for running conversion processes with cpmcopy. On the one hand, an output file name can be specified directly for the output file. The output file target type will be identified by the file extension. On the other hand the output type can be specified by the parameter `type`. With the parameter `splitmask` more than one output file can be written. It is possible to output to `stdout`.

Example	▪ output file name: `cpmcopy -i filea.afp -o filea.pdf`
	▪ output file type to generate an output file of this file type for each input file:
	`cpmcopy -i filea.afp -i fileb.afp -o -type pdf`
	▪ file mask to output more than one output file:
	`cpmcopy -i filea.afp -o -splitmask %01d.png`
	▪ output directory: `FOR %C IN (afpin\*.afp) DO cpmcopy -infile %C –outfile -type PDF pdfout\%~nC.pdf – loglevel v`
	All AFP files in directory `afpin` are converted to PDF files in directory `pdfout`.
	Note: When cpmcopy is called out of a Windows batch file, the percent sign must be coded twice.

page	
Parameter	`-page <m..n>`
Description	Page interval of the input files to be processed. You can code: `-page x` for one page `-page x..` to process all pages from page `x` upward `-page ..y` to process all pages till page `y` `-page x..y` to process the page interval from page `x` till page `y`. As an alternative to the last page you can specify `-page last`. If the number of the first page is greater than the second one, the page sequence of the interval can be reversed. Default: 1..∞
Example	`cpmcopy -i *.afp -page ..2 -o startpagecollection.pdf`

pageheight	
Parameter	`-pageheight <yyy>`
Description	The parameter specifies the page height for output file, e.g.29cm or 11in. To define a relative page size you can specify percentage values. Then, the page will be scaled by these values. The parameters `pageheight` and `pagewidth` can be combined with the parameter `pagescalingfactor` to scale the page size and the page content.
Example	`cpmcopy -i file.jpg -o file.pdf -pagewidth 21cm -pageheight 29cm` `cpmcopy -i file.jpg -o file.pdf -pagewidth 95.0% -pageheight 85.0%cm`

pagescalingfactor	
Parameter	`-pagescalingfactor`
Description	The parameter sets a scaling factor to scale page contents, e.g. 1.25, 0.5. For example, you can scale the page contents to fit into the logical page. You cannot use the parameter to change the page size. To do that you use the parameter `pagesize`.
Example	`cpmcopy -i fileA4.pdf -o fileA5.pdf -pagescalingfactor 0.7 -pagesize A5 -nobinarycopy`

pagesize	
Parameter	`-pagesize <papersize>`
Description	The parameter specifies the paper format for the output file, e.g. A4 or letter. No scaling process is executed. For a list of paper sizes, see *Appendix C: Paper Formats on page 280*.
Example	`cpmcopy -i fileA4.pdf -o fileLetter.pdf -nobinarycopy -pagesize Letter`

pagewidth	
Parameter	`-pagewidth <xxx>`
Description	The parameter specifies the page width for output file, e.g. 21cm or 9in. To define a relative page size you can specify percentage values. Then, the page will be scaled by these values. The parameters `pageheight` and `pagewidth` can be combined with the parameter `pagescalingfactor` to scale the page size and the page content.
Example	`cpmcopy -i file.jpg -o file.pdf -pagewidth 21cm -pageheight 29cm` `cpmcopy -i file.jpg -o file.pdf -pagewidth 95.0% -pageheight 85.0%cm`

pcl.fullprintablearea	
Parameter	`-pcl.fullprintablearea`
Description	PCL specific parameter. You specify the parameter a printer datastream when the physical page size matches the logical page size (not according to PCL specifications).
Example	`cpmcopy -i file.pdf -o file.pcl -pcl.fullprintablearea`

pdf.accessibility.notallowed	
Parameter	`-pdf.accessibility.notallowed`
Description	PDF specific parameter. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Content Copying*, *Content Copying for Accessibility*, and *Page Extraction*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.accessibility.notallowed`

pdf.assemble.notallowed	
Parameter	`-pdf.assemble.notallowed`
Description	PDF specific parameter. The assembling of several documents is not allowed. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Document Assembly*, *Page Extraction*, *Commenting*, *Signing*, and *Creation of Template Pages*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.assemble.notallowed -keylength 40`

pdf.copy.notallowed	
Parameter	`-pdf.copy.notallowed`
Description	PDF specific parameter. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Document Assembly*, *Content Copying*, *Content Copying for Accessibility*, *Page Extraction*, *Commenting*, *Signing*, and *Creation of Template Pages*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.copy.notallowed`

pdf.digitalcopy.notallowed	
Parameter	`-pdf.digitalcopy.notallowed`
Description	PDF specific parameter. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Printing*, *Document Assembly*, *Page Extraction*, *Commenting*, *Signing*, and *Creation of Template Pages*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.digitalcopy.notallowed`

pdf.fill.notallowed	
Parameter	`-pdf.fill.notallowed`
Description	PDF specific parameter. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Document Assembly*, *Page Extraction*, *Commenting*, *Signing*, and *Creation of Template Pages*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.fill.notallowed`

pdf.nochange	
Parameter	`-pdf.nochange`
Description	PDF specific parameter. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Document Assembly*, *Page Extraction*, *Commenting*, *Signing*, and *Creation of Template Pages*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.nochange`

pdf.pagescalingfactor	
Parameter	`-pdf.pagescalingfactor <value>`
Description	PDF specific parameter. You specify the parameter to scale logical page of output file, e.g.: 0.5 = 50% of original size, 1 = original size (default), 2 = double size
Example	`cpmcopy -i file.pdf -o scaledfile.pdf -pdf.pagescalingfactor 0.7`

pdf.print.notallowed	
Parameter	`-pdf.print.notallowed`
Description	PDF specific parameter. The assembling of several documents is not allowed. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Printing*, *Document Assembly*, *Page Extraction*, *Commenting*, *Signing*, and *Creation of Template Pages*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.print.notallowed`

pdf.textnotes.nochange	
Parameter	`-pdf.textnotes.nochange`
Description	PDF specific parameter. If you specify the parameter, the following PDF security attributes are set to *Not Allow*: *Document Assembly*, *Page Extraction*, *Commenting*, *Signing*, and *Creation of Template Pages*. Additionally, you can specify the parameter `keylength` to define the security level.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.textnotes.nochange`

pdf.writeuncompressedstreams	
Parameter	`-pdf.writeuncompressedstreams`
Description	PDF specific parameter. You specify the parameter to write PDF datastream uncompressed.
Example	`cpmcopy -i file.afp -o file.pdf -pdf.writeuncompressedstreams`

ppml	
Parameter	`-ppml <file>`
Description	You specify the parameter to define a Personalized Print Markup Language (PPML) file name for imposition.
Example	`cpmcopy -i file.afp -o file.pdf -ppml UtilsPPML.ppml`

ppml.flushpagecount	
Parameter	`-ppml.flushpagecount <xxx>`
Description	The parameter value specifies the number of pages that are collected in the PPML module and then written (imposition). The parameter value should be directly connected to the page contents that is described in the PPML file. In general, the value matches the number of logical pages (original pages) that are positioned on a sheet by the PPML module. If you omit the parameter, all generated pages are stored and written in one step.
Example	`cpmcopy -i file.afp -o file.pdf -ppml UtilsPPML.ppml -ppml.flushpagecount 3`

printtofile	
Parameter	`-printtofile`
Description	PRT specific parameter. The parameter writes the print datastream to a file.
Example	`cpmcopy -i file.afp -o file.prt -printtofile`

profiledir	
Parameter	`-profiledir <dir>`
Description	You use the parameter to specify a directory different to the program directory of the MFF profile files. Then, the profile files are searched in the specified directory.
Example	`cpmcopy -profiledir profiles -i *.pdf -o -type pcl`

profileentity.external	
Parameter	`-profileentity.external name=value`
Description	You use the parameter `profileentity.internal` and `profileentity.external` to specify an entity instead of entering settings in the profile file. The entity is replaced by the appropriate data when the program is called. When `profileentity.external` is used, the entity is replaced by data of an external file.

Example	First, you replace the value in the profile file by an entity that you want to replace by data in an external file, e.g. the name of a code page in the profile file `mfftxt.pro`. Then, you create an XML file with the XML snippet that replaces the entity, e.g. `<codepage iana = "IBM850"/>`. Then, you can reference XML file.
	`cpmcopy -profiledir profiles -profileentity.external codepage=entity1.xml -ifile.txt -o file.pdf`
	It is possible to replace several entities, e.g.:
	`cpmcopy -profiledir profiles -profileentity.external codepage=entity1.xml -profileentity.external version=entity2.xml -i file.txt -o file.pdf`

profileentity.external	
Parameter	`-profileentity.internal name=value`
Description	You use the parameter `profileentity.internal` and `profileentity.external` to specify an entity instead of entering settings in the profile file. The entity is replaced by the appropriate data when the program is called. When `profileentity.internal` is used, the entity is replaced by data of an external file.

Example	First, you replace the value in the profile file by an entity that you want to replace by data in an external file, e.g. the name of a code page in the profile file `mfftxt.pro`
	`<format>`
	`  <codepage iana = "" />`
	`</format>`
	In the command line you can replace the entity:
	`cpmcopy -profiledir profiles -profileentity.internal code=IBM850 -i file.txt -ofile.pdf`
	It is possible to replace several entities, e.g.:
	`cpmcopy -profiledir profiles -profileentity.internal code=IBM850 -profileentity.internal version=1.4 -i file.txt -o file.pdf`

profilevar	
Parameter	`-profilevar name=value`
Description	You use the parameter to define variables for the MFF profiles files. The name within a MFF profile file can be referenced.

Example	`cpmcopy -infile ENT394296.txt -type txt –profilevar asacode=true -outfile -type pdf`
	MFFAFP profile file extract:
	`<mffafp>`
	`<cp:if test="!VAREXISTS(asacode)">`
	`<cp:then>`
	`<cp:var name="asacode" type="string">'false'</cp:var>`
	`</cp:then>`
	`</cp:if>`
	`<cp:if test="equalsIgnoreCase(asacode,'false')">`
	`<cp:then>`
	`<fontlist>`
	`<font family="Courier New" serifstyle="SERIF" spacing="MONOSPACED" id="1" size="10" />`
	`</fontlist>`
	`</cp:then>`
	`<cp:else>`
	`<fontlist>`
	`<font family="Courier New" serifstyle="SERIF" spacing="MONOSPACED">`
	`<face weight="MEDIUM" id="1" size="10" pitch="15000"/>`
	`<face weight="BOLD" id="3" size="10" pitch="15000"/>`
	`</font>`
	`</fontlist>`
	`</cp:else>`
	`</cp:if>`
	`</mffafp>`
	Note: Currently, `<cp:...>` statements are documented in *DOPE/compose Administration and Implementation Manual, see chapter 7. Sequence Control.*

prt.scalemode	
Parameter	`-prt.scalemode <value>`
Description	PRT specific parameter. You specify the parameter to print directly to the default print (PCL or PostScript). The page content is scaled to the logical page size.
Example	`cpmcopy -i file.jpg -o file.prt -prt.scalemode page`

pwi	
Parameter	`-pwi <value>`
Description	PDF specific parameter. You specify the parameter to remove the password protection of input files.
Example	`cpmcopy -i readprotectedfile.pdf -pwi secret -o file.pdf`

pwo	
Parameter	`-pwo <xxx>`
Description	PDF specific parameter. You specify a password that the user must enter to change an output file (owner password).
Example	`cpmcopy -i file.afp -o changeprotectedfile.pdf -pwo secret`

pwu	
Parameter	`-pwu <xxx>`
Description	PDF specific parameter. You specify a password that the user must enter to open and read an output file (user password).
Example	`cpmcopy -i file.afp -o readprotectedfile.pdf -pwu secret`

quiet	
Parameter	`-quiet`
Description	If you use the parameter, no log messages are issued in the command line, see also the parameter `verbose`.
Example	`cpmcopy -quiet -i file.afp -o -type pdf`

rastx	
Parameter	`-rastx`
Description	AFP specific parameter. The parameter specifies the raster resolution in x-direction. During the dithering process, the raster resolution is synchronized with the resolution of AFP output file.

Example	Two different ways to code the resolution:
	`cpmcopy -i file.jpg -rastx 600 -rasty 600 -bwf -o file.afp`
	`cpmcopy -i file.jpg -bwf -o file.afp -rastx 600 -rasty 600`

rasty	
Parameter	`-rasty`
Description	AFP specific parameter. The parameter specifies the raster resolution in y-direction. During the dithering process, the raster resolution is synchronized with the resolution of AFP output file.

Example	Two different ways to code the resolution:
	`cpmcopy -i file.jpg -rastx 600 -rasty 600 -bwf -o file.afp`
	`cpmcopy -i file.jpg -bwf -o file.afp -rastx 600 -rasty 600`

reduceimagecolors	
Parameter	`-reducex <value>`
Description	You specify the parameter to convert RGB images to palette images in order to use the parameter `-mapcolortoname` in a subsequent processing. You specify the number of colors that the resulting image should contain.
Example	`cpmcopy  -i file.jpg -o -type xif –reduceimagecolors 20`

reducex	
Parameter	`-reducex <value>`
Description	You specify the parameter to define a reduced page size in x-direction.

Example	For the output file, a logical and physical page size is specified.
	`cpmcopy -v -logdir . -i subbu_test.jpg -o -type pcl -converttoimage`
	`-reducex 210mm -reducey 297mm -pagesize A4`

reducey	
Parameter	`-reducey <value>`
Description	You specify the parameter to define a reduced page size in y-direction
Example	`cpmcopy -v -logdir . -i subbu_test.jpg -o -type pcl -converttoimage` `-reducex 210mm -reducey 297mm -pagesize A4`

relaxed	
Parameter	`-relaxed`
Description	Relaxed mode: Fault tolerant control of the conversion process. Potential conversion problems do not abort the conversion process. You can use it for test conversion runs, see also parameter `–strict`. Default: normal mode.
Example	`cpmcopy -relaxed -i file.pdf -o file.afp`

remove	
Parameter	`-remove <itemtype>`
Description	You can use the parameter to remove selected page elements during the conversion process. Following elements can be specified: `a`: arc `b`: barcode `c`: comment `e`: external `f`: field `h`: clipping path `i`: images `l`: lines `n`: link `r`: rec(tangle)s `o`: annotation `p`: path `s`: shading `t`: text Additionally, the following conditions can be specified: `0`: all elements with clipping `1`: all elements without clipping `2`: all filled paths/arcs `3`: all paths/arcs with contours `4`: all fonts If no element types and conditions are specified, all elements are copied.
Example	`cpmcopy -remove abc0 -i file.afp -o file.pdf`

render	
Parameter	`-render <itemtype>`
Description	You can use the parameter to render all or selected page elements: `a`: arc, `p`: path, `b`: barcode, `r`: rec(tangle)s, `e`: overlays, `s`: shading, `i`: images, `t`: text, `l`: lines
Example	`cpmcopy -render abt5 -i file.afp -o file.pdf`

renderonlymonochrome	
Parameter	`-renderonlymonochrome`
Description	You specify the parameter to render all monochrome page elements.
Example	`cpmcopy -renderonlymonochrome abt5 -i file.afp -o file.pdf`

repeat	
Parameter	`-repeat <value>`
Description	You specify the parameter to repeat the conversion process n times to analyze the performance. Thereafter, in a next process steps the result file in generated. Do not use the parameter in a production environment.
Example	`cpmcopy -i file.afp -repeat 3 -o file.pdf`

repeatloops	
Parameter	`-repeatloops <value>`
Description	You specify the parameter to repeat the conversion process n times to analyze the performance. The result file is increased gradually. Do not use the parameter in a production environment.
Example	`cpmcopy -repeatloops -i file.afp -o file.pdf`

replacepatterns	
Parameter	`-replacepatterns`
Description	You specify the parameter to replace patterns by calculated grayscale values. The grayscale value is applied to all color channels.
Example	`cpmcopy -i input.afp -o gray.ps -findpatterns -replacepattern`

reslibin	
Parameter	`-reslibin <xxx>`
Description	You specify the parameter to define a resource library for AFP resources.
Example	`cpmcopy -i file.afp -reslibin RES -o file.pdf`

reverseduplex	
Parameter	`-reverseduplex`
Description	If you specify the parameter the duplex information is reversed, i.e. back page becomes front page and vice versa.
Example	`cpmcopy -i file.afp -reverseduplex -o reverseduplexfile.afp -nobinarycopy`

revision	
Parameter	`-revision <n>`
Description	TIFF specific parameter. You use the parameter to specify the TIFF version. Valid values: 5 or 6.
Example	`cpmcopy -i file.png -o file.tif -type iff.tif -revision 5`

rgbgrayascmyk	
Parameter	`-rgbgrayascmyk`
Description	If you use the parameter, RGB colors representing a gray value (red channel = green channel = blue channel) are converted to a CMYK color with a black channel set appropriately.
Example	`cpmcopy -i rgbgray.afp -o cmyk.pdf -rgbgrayascmyk`

rgbgrayasgray	
Parameter	`-rgbgrayasgray`
Description	If you use the parameter, RGB colors representing a gray value (red channel = green channel = blue channel) are converted to a gray value appropriately.
Example	`cpmcopy -i rgbgray.afp -o gray.pdf -rgbgrayasgray`

rotate	
Parameter	`-rotate <n>`
Description	You specify the parameter to rotate to main text alignment and to use automatic rotation of misaligned pages. The following values are available: `0`, `90`, `180`, `270`, `landscape`, `portrait`, `auto`, `autoignorespaces`, `autocalculateitems`.
Example	`cpmcopy -i file.tif -o rotatedfile.pdf -rotate 270`

rx	
Parameter	`-rx <value>`
Description	Image and AFP specific parameter. You specify the resolution of the output file in x-direction in dpi (dots per inch). You can create thumbnails and specify their size and resolution.
Example	`cpmcopy -i file.tif -o file.png -rx 600 -ry 600`

ry	
Parameter	`-ry <value>`
Description	Image and AFP specific parameter. You specify the resolution of the output file in y-direction in dpi (dots per inch). You can create thumbnails and specify their size and resolution.
Example	`cpmcopy -i file.tif -o file.png -rx 600 -ry 600`

scaletogray	
Parameter	`-scaletogray`
Description	Image specific parameter. The parameter converts an output page into 16 grayscales, not black/white. The font display quality is improved.
Example	`cpmcopy -i file.tif -o file.png -scaletogray`

sd	
Parameter	`-sd <xxx>`
Description	Simplex/Duplex information. Default: Input file settings. Valid values are: `front`: duplex printing – front page `back`: duplex printing – back page `simplex`: single-sided printing `shortfront`: Short edge binding – front `shortback`: Short edge binding – back The parameter `ifnotsetbyinput` specifies that the defined simplex/duplex settings are activated, if the input file does not contain any simplex/duplex information.
Example	`cpmcopy -i file.afp -o duplexfile.pcl -sd front`

selfcontained	
Parameter	`-selfcontained`
Description	Write self-contained AFP files. Required (inline) resources are included into output file.
Example	`cpmcopy -i file.pdf -o -type afp -afp.generateoverlays -selfcontained`

separateoverlays	
Parameter	`-separateoverlays`
Description	Object in an overlay (currently limited to images) that is part of more than one document page or in several input documents is available only once in data stream. For additional occurrences, image references are embedded in overlays.
Example	`cpmcopy -i file.pdf -o file.afp -separateoverlays`

shiftinrect	
Parameter	`-shiftinrect`
Description	In combination with the parameters `–dx` and `–dy` items that are completely within the defined rectangle are shifted.
Example	`cpmcopy -i file.jpg -o shiftfile.pdf -dx 2cm -dy 3cm` `-shiftinrect5cm,4cm,2cm,3cm`

shiftrect	
Parameter	`-shiftrect`
Description	In combination with the parameters `–dx` and `–dy` all items that intersect the area defined by `shiftrect` parameter are shifted.
Example	`cpmcopy -i file.jpg -o shiftfile.pdf -dx 2cm -dy 3cm` `-shiftrect 5cm,4cm,2cm,3cm`

sighandling.disable	
Parameter	`-sighandling.disable`
Description	You specify the parameter to deactivate the signal handling (SEGV, BREAK etc.).
Example	`cpmcopy -sighandling.disable -i *.afp -o -type pdf`

splitdelta	
Parameter	`-splitdelta <n>`
Description	You specify the parameter with a page number value. A new output file is created when the specified number of pages is processed. The file mask for output file names must be specified with the parameter `splitmask`.
Example	`cpmcopy -i file.pdf -o -type xff -splitmask "test%d.xff" -splitdelta 10`

splitmask	
Parameter	`-splitmask <mask>`
Description	You can use the parameter to convert an input file into several output files, i.e. one file per page is default. If you need to specify a different number of pages, you use the parameter `splitdelta`. The file mask creates the output file name. For example, if you specify `%08`, a consecutive 8-digit number with leading zeros is generated. If you specify `test%d`, the file name consists of the prefix "test" and a consecutive number. With the parameter `splitmaskoffset`, you can define the starting number of the output files.
Example	`cpmcopy -i filea.afp -o -splitmask %01d.png` **Note:** When cpmcopy is called out of a Windows batch file, the percent sign must be coded twice.

splitmaskoffset	
Parameter	`-splitmaskoffset <value>`
Description	You use the parameter to specify the file number that starts the consecutive numbering of output files.
Example	In the example, the starting file name is `test3.xff`. `cpmcopy -i file.pdf -o -type xff -splitmask "test%d.xff" -splitdelta 10 -splitmaskoffset 3`

splitnamedcolorimages	
Parameter	`-splitnamedcolorimages <value>`
Description	Palette images are split into grayscale images sorted by color names and stored internally for further processing, for example by an output filter. When you specify the optional parameter value `replace`, the input images are replaced by grayscale images with color names. Otherwise, the input images are still available and not replaced by grayscale images. The example converts the image with the parameters `reduceimagecolors` and `mapcolortoname` into a palette image. The input image is replaced.
Example	`cpmcopy -i test.jpg -o test.xif -reduceimagecolors 256 -mapcolortoname 0x81AF29,25,green -splitnamedcolorimages replace`

stamp	
Parameter	`-stamp <file>`
Description	You use the parameter to add stamp to the document page. Multi-page stamp files are supported. To add a watermark you use the parameter `watermark`.
Example	`cpmcopy -i file.pdf -o stampedfile.pdf -stamp stamp.jpg`

stdin	
Parameter	`-stdin`
Description	If you specify the parameter, the input datastream is read from `stdin`.
Example	`cpmcopy -i -stdin -o -stdout -type iff.bmp`

stdout	
Parameter	`-stdout`
Description	You specify the parameter to write the output datastream to `stdout`.
Example	`cpmcopy -i -stdin -o -stdout -type iff.bmp`

stdoutdirect	
Parameter	`-stdoutdirect`
Description	If you specify the parameter, the output datastream is written without buffering to `stdout`. The parameter cannot be specified for all filters, e.g. not for MFFPOS, MFFVPS, MFFXFF.
Example	`cpmcopy -i 1.afp -o -type xif stdoutdirect`

strict	
Parameter	`-strict`
Description	Strict mode: Non-fault tolerant control of the conversion process. Potential conversion problem may abort the conversion process, see also parameter `–relaxed`. Default: Normal mode.
Example	`cpmcopy -strict -i file.pdf -o file.afp`

subject	
Parameter	`-subject <xxx>`
Description	PDF specific parameter. The subject information for the PDF metadata can be specified.
Example	`cpmcopy -i file.afp -o file.pdf -subject "My subject"` The subject is displayed in the Document Properties window. 

suppresscopycomments	
Parameter	`-suppresscopycomments`
Description	You specify the parameter to suppress comments that are part of the input file.
Example	`cpmcopy -i file.afp -suppresscopycomments -o filewithoutcomments.afp`

suppresscopyinfo	
Parameter	`-suppresscopyinfo`
Description	PDF specific parameter. If you specify the parameter input file metadata (title, author, etc.) are suppressed.
Example	`cpmcopy -i file.pdf -suppresscopyinfo -o nocopyinfo.pdf`

suppressemtpypages	
Parameter	`-suppressemptypages`
Description	You specify the parameter to suppress empty pages in the output file.
Example	`cpmcopy -i duplexempty.afp -o emptyremoved.pdf -suppressemtpypages`

systemfontpath	
Parameter	`-systemfontpath <dir>`
Description	You specify a directory path to the location where system fonts are stored.
Example	`cpmcopy -i file.pdf -systemfontpath C:/Fonts -o file.afp`

title	
Parameter	`-title <value>`
Description	PDF specific parameter. The title information for the PDF metadata can be specified.
Example	`cpmcopy -i file.afp -o file.pdf -subject "my title"` The title is displayed in the Document Properties window. 

tracefile	
Parameter	`-tracefile <filename>`
Description	The default name for the trace file is `cpmcopy.trc`. You specify the parameter to use a user-defined trace file name. **Note:** When you specify a log directory with the parameter `-logdir`, the log file is stored in the log directory and the trace file in the root directory after the first cpmcopy run. After all further cpmcopy runs, when you have specified the parameter `-logdir`, the log file, and the trace file are stored in the log directory.
Example	`cpmcopy -tracefile mytracefile.trc -i *.afp -o -type pdf`

tracelevel	
Parameter	`-tracelevel <n>`
Description	The parameter specified the trace level (0 - 9). By default, the trace level is specified by the environmental variable `CPTRACE`.
Example	`cpmcopy -tracelevel 2 -i *.afp -o -type pdf`

tray	
Parameter	`-tray <xxx>`
Description	AFP, PCL, and PRT specific parameter. The parameter specified the printer tray name. The information is written into the output file.

Example	`cpmcopy -i file.pdf -o file.afp -tray TRAY2`
	Working with PCL files it is required to map the tray name to a device ID in the profile file `mffpcl.pro`.
	`<inputtray deviceid='1' name="Tray2">`

truetypepath	
Parameter	`-truetypepath <path>`
Description	The parameter specifies a directory path to location where TrueType fonts are stored.
Example	`cpmcopy -i file.afp -truetypepath C:\Fonts -o file.pdf`

type	
Parameter	`-type xxx[.yyy]`
Description	If you specify the parameter `type`, an input or output file will be processed with the specified format type. For input files, it is recommended to specify the type for rarely used or unknown file types as for files that are processed with a differing profile file. If necessary, a subtype can specified.

Example	The example conversion process runs with a PDF profile.
	`cpmcopy -i file.tst -type pdf -o file.xif`
	Instead of specifying an output file just the target format is defined with `type`. For each input file an output file in generated with the appropriate format type.
	`cpmcopy -i *.afp -o -type pdf`
	▪ AFP, subtypes: PSG, OVL
	▪ IFF, subtypes: BMP, GIF, HDP, IOC, JB2, JP2, JPG, PCX, PNG, TGA, TIF, XRX
	▪ GOF, subtype: OTF
	▪ PCL
	▪ PDF
	▪ POS
	▪ PRT, for output file printer name is specified as subtype:
	`-outfile - type "prt.\\server\printer"`
	▪ RTF (input only)
	▪ SVG
	▪ TXT
	▪ VIP (input only)
	▪ XIF
	▪ XFF
	▪ XHM
	▪ XRX, subtypes: LCD (input), MET (output)

uselocale	
Parameter	`-uselocale <value>`
Description	You use the parameter to specify the local system code page used by the program. The default value is C (Posix). The setting controls which code page is used when the operating system interprets strings. This impacts file name, console output, etc., see also the parameter `setlocale`.

Example	Strings under Linux `uselocale=de_DE-UTF8`
	`cpmcopy -uselocale de_DE-UTF8 -i *.afp -o -type pdf`
	Windows with German code page `uselocale=de_DE`
	`cpmcopy -uselocale de_DE -i *.afp -o -type pdf`

verbose	
Parameter	`-verbose`
Description	If you specify the parameter detailed information is issued in the command line, see also the parameter `quiet`.
Example	`cpmcopy -verbose -i file.afp -o -type pdf`

watermark	
Parameter	`-watermark`
Description	You use the parameter to add a watermark to a file.
Example	`cpmcopy -i file.pdf -o filewithwatermark.pdf -watermark draft.jpg`

3.3. cpmill Command Line Parameters and Variables

The tables below list parameter characters including its type and description in alphabetical order.

-a	
Parameter	`-a`
Parameter Type	Boolean
Description	Audible signal on some processing events

-A	
Parameter	`-A`
Parameter Type	Boolean
Description	No new log file is generated. Data is stored into an existing log file.

-b	
Parameter	`-b`
Parameter Type	String
Description	Directory with Compart binaries

-c	
Parameter	`-c`
Parameter Type	Boolean
Description	No console I/O. No output to stderr/stdout (except if requested explicitly in some other context, i.e. java flush buffers)

-d	
Parameter	`-d`
Parameter Type	Boolean
Description	Use the parameter, if cpmill runs as service, daemon, in detached context. It will trigger some other switches, no console log etc.

-D	
Parameter	`-D`
Parameter Type	NAME=VALUE assignment
Description	Define variables for later processing

-e	
Parameter	`-e`
Parameter Type	Boolean
Description	Emulate processing. Do not really do it.

-E	
Parameter	`-E`
Parameter Type	NAME=VALUE assignment
Description	Define so-called external entities, `cpmill -Emyentity=myfile.pro`. Any occurrence of the entity `&myentity;` in a cpmill profile will then be replaced with the content of `myfile.pro`.

-f	
Parameter	`-f`
Parameter Type	Boolean
Description	Enable smart file operations (delete/retry)

-g	
Parameter	`-g`
Parameter Type	Boolean
Description	Debug, turns on even more traces

-H	
Parameter	`-H`
Parameter Type	Boolean
Description	Enable the system's signal handler. By default, cpmill will install its own signal handler in order to catch cases of an abnormal program ending. This will prevent the operating system's signal handler from kicking in and, for example on Windows, display dialog boxes that may halt batch processing. When cpmill's signal handler kicks in, you'll see the following message: `DMI1900F SEGMENTATION VIOLATION`

-j	
Parameter	`-j`
Parameter Type	Boolean
Description	Specify Java context, slightly different application behavior regarding stdio/stderr, etc.

-J	
Parameter	`-J`
Parameter Type	Integer number
Description	Memory for JavaScript runtime in MB (default: 8 MB)

-k	
Parameter	`-k`
Parameter Type	Boolean
Description	For test purposed only. Input files are not deleted after processing, and if necessary they are processed several times (driver related). Temporary files and incomplete output files remain available. The parameter is equivalent to the attribute `keepinput="yes"` of the element `<globals>` of the cpmill profile.

-l	
Parameter	`-l`
Parameter Type	String
Description	Specify log file name

-L	
Parameter	`-L`
Parameter Type	String
Description	Specify license file name. Default: cpmill.lic

-m	
Parameter	`-m`
Parameter Type	Long integer number
Description	Set maximum number of file handles (WIN32 and Unix only)

-O	
Parameter	`-O`
Parameter Type	String
Description	Specify computed OEM license key; matched against public key portion as defined in license file

-p	
Parameter	`-p`
Parameter Type	Boolean
Description	Disable semaphore protection. By default, it is not possible to run more than one instance of cpmill on the same machine. This is not a technical restriction. Nevertheless, it is a protection against accidental invocation of two cpmill instances on the same set of files. Provided that you keep the cpmill instances strictly separated, i.e. including all input, output, and temporary(!) directories - including those hidden in filter profiles files - you can use the command line option `-p` to override the semaphore and run two (or more) cpmill instances. **Important:** Double-check your configuration and profiles first. Use the parameter at your own risk!

-P	
Parameter	`-P`
Parameter Type	NAME=VALUE assignment
Description	Define MFF profile variables

-R	
Parameter	`-R`
Parameter Type	Boolean
Description	Relaxed mode. cpmill starts in Normal mode (while all previous versions started in Relaxed mode). For error messages related to a processing in Strict and Normal mode, see *Appendix F: Monitored Messages on page 286*. If neither `-R` nor `-S` are specified, then cpmill will start in Normal mode. Specifying both options will cause cpmill to abort with an error message: `Conflicting warning modes selected. Use either 'strict' or 'relaxed'`

-s	
Parameter	`-s`
Parameter Type	Boolean
Description	Run only once

-S	
Parameter	`-S`
Parameter Type	Boolean
Description	Strict mode. cpmill starts in Normal mode (while all previous versions started in Relaxed mode). For error messages related to a processing in Strict and Normal mode, see *Appendix F: Monitored Messages on page 286*. If neither `-R` nor `-S` are specified, then cpmill will start in Normal mode. Specifying both options will cause cpmill to abort with an error message: `Conflicting warning modes selected. Use either 'strict' or 'relaxed'`

-t	
Parameter	`-t`
Parameter Type	Long integer number
Description	Internal trace level 0-9

-T	
Parameter	`-T`
Parameter Type	String
Description	Specify trace file name

-v	
Parameter	`-v`
Parameter Type	Boolean
Description	Verbose, turn on some more traces

-u	
Parameter	-u
Parameter Type	String
Description	Only specified unit key from profile will be executed (even if disabled!). All non-matching unit keys will be disabled.

-X	
Parameter	-X
Parameter Type	String
Description	Entity definition for MFF profiles. The name can be referenced in a MFF profile file.

-0	
Parameter	-0
Parameter Type	Boolean
Description	Set trace level to 0. It overrides any environment setting.

-?	
Parameter	-?
Parameter Type	Boolean
Description	Help. Explanatory text

Note: If you specify Boolean values in cpmill profiles, use the values `true` and `false`, in cpmill command line use *-n* or *-n+* for true and *-n-* for false.

3.3.1. cpmill Example Command Line Call

The standard cpmill command line call example:

```
cpmill –uUNITNAME cpmill.pro
```

3.3.2. Variables

The special control of the processing is done by rules (formulas), which can record the processing status via variables. Variables can be specified by the user in the profile and via the `–D` parameter.

In addition, cpmill can work with several defined system variables, which can be used without pre-determination (inside of formulas).

3.3.3. System Defined Variables

cpmill uses the system defined variables listed in the table below.

Note: In the DocBridge Mill JavaScript environment the variable prefix **$** is replaced by **sys.**, for example `sys.page` instead of `$PAGE`.

Table 9: cpmill System Defined Variables

Variable	Description
$COUNTER_0000	Global internal counter
$CURRENT_PAGE_IN_SD_BACK	Current input page is front page (0) or back side (1) (backside only if duplex): long edge
$CURRENT_PAGE_IN_SD_DUPLEX	Current input page is simplex (0) or duplex (1): long edge
$CURRENT_PAGE_IN_SE_BACK	Current input page is front page (0) or back side (1) (backside only if duplex): short edge
$CURRENT_PAGE_IN_SE_DUPLEX	Current input page is simplex (0) or duplex (1): short edge
$DATE_DDMMYY	Current date, format DD.MM.YY
$DATE_YYMMDD	Current date, format YY.MM.DD
$DOCUMENT_EXT_MAIN	File extension of currently polled input file
$DOCUMENT_NAME_MAIN	Current name (file name body) of polled input file
$DOCUMENTNUMBER	Current input document sequence number within output as 1-based number
$**FILEEXTENSION**	Extension of the output file
$**FILETYPE**	Output file type
$INPUTPAGE	Current page within current document 1-based number
$PAGE	Current page within current document 1-based number
$PREV_DOCUMENT_EXT_MAIN	File extension of previous polled input file
$PREV_DOCUMENT_NAME_MAIN	Previous name (file name body) of polled input file
$ROOT_DIR	Base/root directory of current unit
$SDATE_YYMMDD	Current date, YYMMDD
$SPOOLPAGE	Current page number within input spool 1-based number
$STIME_HHMM	Current time, HHMM
$STIME_HHMMSS	Current time, HHMMSS
$TIME_HHMMSS	Current time, format HH.MM.SS
$UNIT_KEY	Key of current unit
$USERSTRING_00	Application dependant user string 00, i.e. DocBridge Pilot: eStamping
$USERSTRING_01	Application dependant user string 01
$USERSTRING_02	Application dependant user string 02

4. JavaScript in DocBridge Mill

JavaScript support in cpmill component of DocBridge Mill provides a fully JavaScript 1.7 compliant interpreter (based on the Mozilla project's SpiderMonkey JavaScript engine) that has been extended to support a variety of Compart specific objects.

4.1. General Information

4.1.1. Global and Local Variables in JavaScript

Working with variables in JavaScript is different from other programming languages. Note the different ways of declaring variables in the chapter below. Keep in mind that all code samples are just samples and must be modified before use.

4.1.1.1. Global and Local Variables in JavaScript

The scope of a variable refers to the variable's visibility within a program. Variables, which are accessible only to a restricted portion of a program, are said to have local scope. Variables that are accessible from anywhere in the program, however, are said to have global scope.

In JavaScript, variables fall into two major categories: **global** and **local**. Global variables are accessible from anywhere in the program and retain their values until the document is unloaded. Local variables, on the other hand, are created during function calls for temporary use and are accessible only within the functions that create them. This is done by preceding the first instance of the variable's name with the keyword `var`.

Global variables that are changed accidentally are often a source of errors. Compart recommends to use local variables, if possible, and declare all variables with `var`. For you and others the program readability will be improved. The program code will be less error-prone and easier to maintain.

Example: Local Variable

```
var a = 10;

disp_a();

function disp_a()
   {
   var a = 20;
   alert("Value of 'a' inside the function " + a);
   }

alert("Value of 'a' outside the function " + a);
```

Example: Global Variable

```
var a = 10;

disp_a();

function disp_a()
   {
   a = 20;
   alert("Value of 'a' inside the function " + a);
   }

alert("Value of 'a' outside the function " + a);
```

4.1.2. JavaScript in cpmill

The chapter describes the use of the Compart Document Components (Docponents), and, specifically, their use from within the JavaScript sections (event handlers) in a cpmill profile. The table below lists the Docponents classes with its descriptions. For deprecated JavaScript constructs, see *Appendix A: Deprecated JavaScript Constructs on page 276*.

Table 10: Docponents Classes

Class Name	Description
Annotation	Docponent class: Annotation item. It provides an annotation item, see *chapter 4.10.1.8. Annotation on page 105*.
Barcode	Docponent Barcode class to create barcodes, see *chapter 4.7. Barcode on page 93*.
Bookmark	Docponent Bookmark class to create bookmarks, see *chapter 4.9. Bookmark and TOC on page 98*.
Color	Docponent class: Color item. It provides a color item, see *chapter 4.10.1.2. Color on page 102*.
Comment	Docponent class: Comment item. It provides a comment item, see *chapter 4.5.1. Comment) on page 90*.
Docponent	This is the generic Docponent parent class from which all the other Docponent classes are derived. The methods described here can be applied to all Docponent objects, i.e. even those that have been created through the more specific constructors.
ExternalItem	Docponent class: External Item. It provides an external item, see *chapter 4.10.1.4. ExternalItem on page 103*.
FileInputBinary	Docponent helper class: Binary File Access. It is a helper class that provides binary file access, see *chapter 4.13.8. FileInputBinary on page 128*.
FileInputDocument	Docponent helper class: Input File. It is a helper class that provides an abstraction of files, see *chapter 4.13.3. FileInputDocument on page 114 on page 113*.
FileInputText	Docponent helper class: Text File Access. It is a helper class that provides text-only file access, see *chapter 4.13.6. FileInputText on page 127*.
FileOutputBinary	Docponent helper class: Binary File Access. It is a helper class that provides binary file access, see *chapter 4.13.9. FileOutputBinary on page 129*.
FileOutputDocument	Docponent helper class: Output File. It is a helper class that provides an abstraction of files so that documents, sheets, and pages can be written to a file, see *chapter 4.13.4. FileOutputDocument on page 116.*
FileOutputText	Docponent helper class: Text File Access. It is a helper class that provides text-only file access, see *chapter 4.13.7. FileOutputText on page 128.*
Font	Docponent class: Font item. It provides a font object to be used with text items, see *chapter 4.10.1.1. Font) on page 101*.
Group	Docponent class to handle a document's group information, see *chapter 4.6. Group on page 92*.
Length	Docponent helper class: Length. It provides an abstract length class, so we can handle lengths of 2.31in etc, see *chapter 4.11. Length on page 108*.
Line	Docponent class: Line Item. It provides a line item, see *chapter 4.10.1.7. Line on page 105.*
Omr	Docponent class to create OMR marks, see *chapter 4.8. OMR on page 96*.
Page	Docponent class that describes pages, see *chapter 4.2. Page on page 72*.
Rect	Docponent class: Rect item. It provides a rectangle item, see *chapter 4.10.1.6. Rect on page 105*.
Text	Docponent class: Text item. It provides a text item that can be put on pages, see *chapter 4.10.1.3. Text) on page 103*.
TOC	Docponent TOC class to handle bookmarks, see *chapter 4.9. Bookmark and TOC on page 98*.

4.1.2.1. Events

The JavaScript driven processing in cpmill is based on five events:

- `beginprocess`
- `documentstart`
- `documentpageread`
- `documentend`
- `endprocess`

As the names imply, these events are sent when cpmill begins processing, when a document starts, when a page has been read, when a document ends, and when the processing ends, respectively.

You can implement event handlers for any of the above events. Event handlers are all optional.

```
<eventhandler type="documentstart">
    <script type="text/javascript">

        // JavaScript code goes here

    </script>
</eventhandler>
```

The following events are available additionally to the above listed:

- `openinputdocument`
- `openoutputdocument`
- `inputdocumentopened`
- `outputdocumentopened`

They give the user more control over the parameters with which cpmill's main input and output files are opened and created, see also *chapter 4.13.5.4. Integration with cpmill on page 126*.

- `begin`
- `end`

These events are only fired once per unit, at the very beginning and very end of the unit processing, respectively. As such, they are executed before `beginprocess` and `endprocess`.

- `successinfo`
- `errorinfo`

These events are only fired when a `successaction` or `erroraction` is triggered. They do not allow the script to change the action. They will only inform the script so that some additional action can be triggered.

4.1.2.2. Examples

A Minimal Event Handler

cpmill sends a `documentpageread` event for every page it reads during processing. In order for those pages to actually appear in the output document(s), you have to implement the following minimal event handler:

```
<eventhandler type="documentpageread">
    <script type="text/javascript">

        output.push(page);

    </script>
</eventhandler>
```

Here, `page` is an object representing the current page (the one that cpmill just read for this event) and `output` is cpmill's output queue.

Note: Only pages that are pushed into the output queue will be written!

Skipping Pages

The global `sys` variable provides access to predefined values (e.g. the number of the current page) as well as all those values coming from a `*.csv` file (named after the column name).

If you want to write out, for example, only every second page, you can do the following:

```
<eventhandler type="documentpageread">
    <script type="text/javascript">

        if(sys.PAGE % 2 == 1)
            {
            output.push(page);
            }

    </script>
</eventhandler>
```

See also *chapter A Minimal Event Handler on page 67* and remember that only those pages that are pushed into the output queue are written. All other pages simply "vanish". Here, it only writes out the odd-numbered pages from the input document(s), i.e. pages 1, 3, 5, ...

Note: For those who are familiar with the "classic" cpmill profiles: All the variables, that were preceded by a `$` character there, are available in JavaScript as well by simply replacing the character with `sys`.

Delayed Writing

You do not have to push a page into the output queue in the `documentpageread` event handler. You can do that at a later stage, e.g. in the `documentend` or `endprocess` event handler. However, you have to store the page somewhere in the meantime, e.g. in a JavaScript array:

```
<eventhandler type="beginprocess">
    <script type="text/javascript">

        stack = new Array();

    </script>
</eventhandler>

<eventhandler type="documentpageread">
    <script type="text/javascript">

        stack.push(page);

    </script>
</eventhandler>

<eventhandler type="endprocess">
    <script type="text/javascript">

        for(var i = 0; i &lt; stack.length; i++)
            {
            output.push(stack[i]);
            }

    </script>
</eventhandler>
```

The above example stores the page in a JavaScript array called `stack` in the `documentpageread` event handler and only writes its contents out at the end of the processing, i.e. in the `endprocess` event handler. You can also do this at the document level (i.e. in the `documentend` event handler) if needed - or even in the `documentpageread` event handler, based on certain criteria.

Note: JavaScript arrays are very flexible and can be used as a stack (as in the above example) as well as like normal arrays (as in other programming languages).

4.1.2.3. Event Calling Order

The events are called in the following order:

- `beginprocess`: once per cpmill run
- `openoutputdocument`: once for every output file, before the file is opened
- `openinputdocument`: once for every input file, before the file is opened
- `inputdocumentopened`: once for every input file, after the file was opened
- `outputdocumentopened`: once for every output file, after the file was opened
- `documentstart`: once per document
- `documentpageread`: once for every page
- `documentend`: once per document
- `endprocess`: once per cpmill run

Note: Some events are called more than once.

Calling Order Examples

For a cpmill driver 121 with one input file that contains two pages, the calling order would be like this:

```
 1. beginprocess
 2. openoutputdocument
 3. openinputdocument
 4. inputdocumentopened
 5. outputdocumentopened
 6. documentstart
 7. documentpageread
 8. documentpageread
 9. documentend
10. endprocess
```

For a cpmill driver 121 with two input files and two pages each, the calling order would be like this:

```
 1. beginprocess
 2. openoutputdocument
 3. openinputdocument
 4. inputdocumentopened
 5. outputdocumentopened
 6. documentstart
 7. documentpageread
 8. documentpageread
 9. documentend
10. openoutputdocument
11. openinputdocument
12. inputdocumentopened
13. outputdocumentopened
14. documentstart
15. documentpageread
16. documentpageread
17. documentend
18. endprocess
```

Note: The events `beginprocess` and `endprocess` are currently called for every input file.

For a cpmill driver N21 with two input files with two pages each, the calling order will be like this:

```
 1. beginprocess
 2. openoutputdocument
 3. openinputdocument
 4. inputdocumentopened
 5. outputdocumentopened
 6. documentstart
 7. documentpageread
 8. documentpageread
 9. documentend
10. openinputdocument
11. inputdocumentopened
12. documentstart
13. documentpageread
14. documentpageread
16. documentend
17. endprocess
```

The calling order is mostly identical to the example for the cpmill driver 121 with more than one input file, with the exception that the cpmill driver N21 only has one output file.

For a CLU (cluster or CSV) driver with one CSV file containing two lines, referencing documents with two pages each, the calling order will be like this:

```
 1. beginprocess
 2. openoutputdocument
 3. openinputdocument
 4. inputdocumentopened
 5. outputdocumentopened
 6. documentstart
 7. documentpageread
 8. documentpageread
 9. documentend
10. openinputdocument
11. inputdocumentopened
12. documentstart
13. documentpageread
14. documentpageread
15. documentend
16. endprocess
```

In this example, the calling order is identical to that of the cpmill driver N21 above. With more than one CSV file, you would see additional calls to `openoutputdocument` and `outputdocumentopened`, though.

4.1.2.4. Event Handlers and Objects

The overview below lists the objects that are available for the event handlers. **Note:** Not all objects are available for all event handlers.

- `begin`: output object
- `beginprocess`: output object, context object
- `openoutputdocument`: output object, OutputDocumentAttributes object, file name
- `openinputdocument`: output object, InputDocumentAttributes object, file name
- `inputdocumentopened`: output object, InputDocumentAttributes object, file name
- `outputdocumentopened`: output object, OutputDocumentAttributes object, file name
- `documentstart`: output object, context object
- `documentpageread`: output object, context object
- `documentend`: output object, context object
- `endprocess`: output object, context object

4.1.3. Advanced Processing Mode Control

With JavaScript for cpmill, the processing mode can be further modified, see also *Appendix F: Monitored Messages on page 286* and the parameter `general.warningmode` in *chapter 5.2.1. cpmill Profile Element <parameter> on page 211*. To set the advanced processing mode control, you the global object `Frame`.

```
docponentFrame = context.getFrame();
```

You can add messages that inform about error that terminate the process to that very list. In the example below, PDI2010 is the ID of the message.

```
docponentFrame.unregisterBreakMessage("PDI2010");
```

You can remove messages that inform about error that terminate the process from that very list. In the example below, RMG2001 is the ID of the message.

```
docponentFrame.registerBreakMessage("RMG2001");
```

You can issue the list of message that inform about termination errors.

```
iNumOfBreakMessages = docponentFrame.getNumberOfBreakMessages();
for (ibc = 0; ibc < iNumOfBreakMessages ; ibc++)
    {
    log(" abort log message " + ibc + ": " + docponentFrame.getBreakMessage(ibc));
    }
```

4.2. Page

4.2.1. Page Object

In addition to the page object, representing the current page that is passed to the event handler `documentpageread` you can create new pages at any time:

```
p = new Page();
p2 = new Page(new Length("8in"), new Length("11in"));
```

Operations related to the `page` object are:

- Page rotation: `setRotation`, `setRotationMode`, `setInRotationMode`
- Index: `getIndex`, `setIndex`
- Group: `getGroup`, `setGroup`
- Comments: `getComments`, `setComments`
- Trays: `setPaperInTray`, `setPaperOutTray`
- Capturing: `captureText`, `captureBarcode`

4.2.1.1. Adding Objects to a Page

It is possible to add page items (DocponentItem object) and page objects to a page. There are two different methods to add objects: `put` and `place`. The method `place` offers an additional parameter for the z-order.

The methods `put` and `place` make a copy of the object and inserts the copied object on the page.

```
page.put(pageAdditional, new Length("1in"), new Length("1in"));
// or
page.place(pageAdditional, new Length("1in"), new Length("1in"));
```

When the object on the page needs to be modified, it has to be retrieved from the page with the method `select`.

Additionally, the method `place` allows putting objects “below” others. A typical use may be to “watermark” a page with another page containing an image (watermark or stamp).

```
page.place(imagePage, new Length("25mm"), new Length("2in"), Constants.Docponent.Z_ORDER.First);
```

Note: This is not identical to placing the `page` on the `imagePage`, because the `imagePage` is here just treated as one object on the `page`.

4.2.1.2. Page Size

Changing the Page Size

To change the size of an existing page, use `setPageSize`:

```
p.setPageSize(new Length("8in"), new Length("11in"));
```

Note: This does **not** scale the page. It only cuts it off or expands its size.

Checking the Page Size

If you need to know the size of a page, you can get the page's respective width and height as a Length object, see *chapter 4.11. Length on page 108*.

```
width = page.getWidth();
height = page.getHeight();
```

There are also convenience functions to check if a page is has landscape or portrait orientation:

```
if(page.isLandscape())
    {
    // do something
    }

if(page.isPortrait())
    {
    // do something else
    }
```

Note: For a *square* page size, both `isLandscape` and `isPortrait` would return false.

4.2.1.3. Scaling

ScaleToImage

Note: For most MFF filters, scaling currently works by converting the page content to an image. It is done implicitly when using this function:

```
page.scaleToImage(new Length("148mm"), new Length("210mm"));
```

The second parameter (height) is optional and can be left off. Scaling will then be done in proportion to the page width:

```
page.scaleToImage(new Length("148mm"));
```

Note: When scaling is used in combination with page rotation, see *chapter 4.2.4. Page Rotation on page 85*, you will have to swap the width and height values if the page is rotated by 90 or 270 degrees! In such a case, it is recommended to specify both value, i.e. width and height.

Since this operation involves converting the page to an image, you can also (optionally) specify value for the image resolution, see *section convertToImage on page 76*:

```
page.scaleToImage(new Length("148mm"), new Length("210mm"), 240, 240);
```

Note: The resolution value for the y-direction is optional.

SetScaleFactor

Setting a scale factor for a page will resize **all** of the items on the page by that factor:

```
page.setScaleFactor(0.9);
// or equivalent:
page.setScaleFactor(0.9, false);
```

When the page size should be kept as it is and only the content should be scaled, the following can be used:

```
page.setScaleFactor(0.9, true);
```

Note: This method currently only works for PDF. It should not be used with other methods that change the page size.

4.2.1.4. Cropping

Parts of a page can be returned as a new `Page` object using the function `crop`:

```
rect = page.crop(new Length("0"), new Length("1cm"),
                 new Length("4in"), new Length("4in"));
```

The four values represent the x-coordinate and y-coordinate as well as the width and height (in that order) of the rectangle.

Note: `crop` leaves the specified rectangle unchanged. It returns a copy of all the objects within that rectangle.

4.2.1.5. Shifting Page Contents

There are two methods for shifting the contents of a page:

- `setInShift` to shift the contents right after the page has been read from the file
- `setShift` to shift the contents after all page modifications (e.g. adding OMRs, barcodes, etc.) have been applied

```
p.setInShift(new Length("10mm"), new Length("0mm"));

o = new Omr();
o.setMetrics(new Length("7mm"), new Length("28tw"), new Length("24pt"));
[...]
p.put(o, new Length("2mm"), new Length("1in"));
```

The example above shifts the page contents 10mm to the right to make room for the OMR.

The shifting methods can be used to emulate functionalities of the attributes `pageoffsetx` and `pageoffsety` of CpCOLD's element `<coldunit>`.

4.2.1.6. Shifting Rectangle Contents

To shift the content of a specified rectangle, use `setShiftRect`:

```
rx = new Length("115mm");
ry = new Length("40mm");
rw = new Length("80mm");
rh = new Length("2415tw");
shiftx = new Length("-50mm");
shifty = new Length("25mm");
page.setShiftRect(shiftx, shifty, rx, ry, rw, rh);
```

The first two parameters specify the shift offset, as with `setShift` and `setInShift` in *chapter 4.2.1.5. Shifting Page Contents on page 74*.

An additional (optional) parameter allows you to select which items should be shifted: Only those that fall entirely into the given rectangle or all those that are at least partially within the given rectangle, which is the default behavior.

```
page.setShiftRect(shiftx, shifty, rx, ry, rw, rh, Constants.Page.Shift.CompleteIn);
```

4.2.1.7. Replacing Variables

When the page represents for example an XIF file, you may want to replace the variables it contains. To do that, you need a varpool that contains the variables (and their values) that you want to replace. You can either create a copy of the `sys` varpool or your own `Vars` (varpool) object, see *chapter 4.12. Vars (Varpool) on page 111*:

```
f = new FileInputDocument(contect.getFileMgr(), "stamp.xif");
p = f.readPage(1);

v = new Vars(sys);
v.myvariable = 'myvalue';

p.replaceVars(v);
```

Note: The `Page` object (`p` in the above example) will be “used up” by the `replaceVars` call, i.e. you cannot use it again for another `replaceVars` with different variable. However, you can clone the page object first:

```
f = new FileInputDocument(contect.getFileMgr(), "stamp.xif");
p = f.readPage(1);

clone = p.clone(); // clone the template

v = new Vars(sys);

v.myvariable = 'myvalue';
p.replaceVars(v); // replace in original

v.myvariable = 'anothervalue';
clone.replaceVars(v); // replace in cloned object
```

4.2.1.8. Transparency

To make objects transparent on a page, you can use the method `setTransparency`:

```
page.setTransparency(true, Constants.Page.Transparency.Text);
```

Constants you can use (should be self-explanatory): `Text`, `Line`, `Rectangle`, `Image`, `External`, `Comment`, `Path`, `Clippath`, `Annotation`, `Arc`, `Barcode`, `Field`, `Link`, `All`.

These constants can be OR’ed together, e.g. to make all text and barcode objects transparent:

```
trans = Constants.Page.Transparency.Text | Constants.Page.Transparency.Barcode;
page.setTransparency(true, trans);
```

4.2.1.9. Merging Images

To merge adjacent images into a single large image you can use the method `setCombineImages`.

```
page.setCombineImages(true);
```

4.2.1.10. Converting to Image

In some cases it is required to convert the page contents to an image and only wrapping that page in the output format (so that, for example, every page in the resulting PDF is in fact an image). You can do that on a per-page basis using `convertToImage`.

convertToImage

```
page.convertToImage(true);
```

It is also possible to specify the resolution of the image (default resolution 300 dpi):

```
page.convertToImage(true, 240); // == 240 x 240 dpi
page.convertToImage(true, 300, 600);
```

Specifying only one value is equivalent to specifying the same value for x resolution and y resolution.

Note: Using different values for the x and y resolution can have unintended side effects, as not all output filters can handle this properly (or at all).

convertItemsToImage

You use the method `convertsItemsToImage` to convert only certain item types to an image.

```
page.convertItemsToImage(Constants.Page.ItemType.Text);
```

The example above converts all text items on the page to one image. It leaves all other items untouched. The `Constants` can be OR'ed together to convert items of several types at once.

Like `convertToImage`, this method has optional parameters to set the image resolution:

```
page.convertItemsToImage(Constants.Page.ItemType.Text, 300);
page.convertItemsToImage(Constants.Page.ItemType.Text, 300, 600);
```

Another optional parameter converts all the items to separate images:

```
page.convertItemsToImage(Constants.Page.ItemType.Text, 300, 300, true);
```

To convert several image types to images, you have to OR their respective values:

```
page.convertItemsToImage(Constants.Page.ItemType.Text | Constants.Page.ItemType.Line, 300);
```

There's also a special flag to indicate that only rotated items of the given type should be converted to images.

```
page.convertItemsToImage(Constants.Page.ItemType.Text | Constants.Page.ItemType.Rotated);
```

This will only convert rotated text items to images while leaving non-rotated items unchanged.

Note: `convertItemsToImage()` overrides `convertToImage()`.

setImageConversionType

The method `setImageConversionType` detects the required image depth automatically and color and grayscale images will be generated. To override this automatic behavior, you code the following:

```
page.setImageConversionType(Constants.Page.ImageType.Grayscale);
```

Available image types are `Monochrome`, `Grayscale`, `RGB`, and `CMYK`.

This override applies to the following functions:

- `convertToImage`
- `convertItemsToImage`
- `scaleToImage`

CYMK images will be converted to RGB unless you explicitly set the image conversion type to CMYK.

To query the current state of the conversion, you use the method `getImageConversionType()`. It may return `Default`, if no specific image type has been set yet.

setKeepTextAsHiddenText

It is possible to keep text as invisible text on a page even when the page is rendered to an image. The method `setKeepTextAsHiddenText` (`true`/`false` (Standard)) turns this behavior on or off.

To query the current state of visibility, you use the method `getKeepTextAsHiddenText()`.

4.2.1.11. Dithering and Threshold

Available types for `DitherTypes` are:

- `Dispersed`
- `Clustered`
- `FloydSteinberg`
- `Stucki`
- `JarvisJudiceNinke`

The second parameter is optional and specifies the size of the dither matrix (not needed for `Floyd-Steinberg`).

Note: The dither methods `Stucki` and `JarvisJudiceNinke` have been available in all recent releases.

```
page.setDither(Constants.Page.DitherType.Dispersed, 0);
```

In some (rare) cases, image dithering may not provide usable results. Setting a `threshold` value is a simple means of converting an image to monochrome. All pixels below that level will be rendered as white, while all pixels above that level will be rendered as black. The threshold value is defined in percent:

```
page.setThreshold(60);
```

4.2.1.12. Checking for Blank Areas

With the method `isBlank`, you can check if a rectangular area is blank, i.e. does not contain any visible (!) items.

```
if(page.isBlank(x, y, w, h)) ...
```

All four parameters are optional. **Note:** When you call the method `isBlank()` without any parameters, it will check the entire page. However, this will use a lot of memory (and time) and should only be used sparingly.

Internally, this function will convert the specified part of the page to an image and check if there are any non-white pixels in that area. If you need more control over the conversion to an image, you may want to do it yourself, using the method `convertToImage()` and the dither methods first before calling the method `isBlank`, see also *chapter 4.2.1.11. Dithering and Threshold on page 77*.

The method `isBlank` will use a maximum resolution of 300 dpi for the image (i.e. only if the page's resolution is higher than 300 dpi) to speed up the function.

In case you need a higher resolution, you can now specify the resolution as two (optional) parameters. You only need to specify both in the rare case that the x and y resolution differ.

```
if(page.isBlank(x, y, w, h, 7200)) ...
```

Detailed information about parameters of length objects, see *chapter 4.11. Length on page 108*.

4.2.1.13. Setting Thresholds for Black and White

With the method `setBlackWhiteAdjust`, you can force, that everything on the page that is darker than a supplied grayscale value is set to black and everything that is lighter than a supplied grayscale value is set to white.

```
page.setBlackWhiteAdjust(<black>, <white>);
```

Possible values are: `0 <= <white>-1 <= <black> <= 100`.

4.2.1.14. Optimizing the Output

Replacing Patterns

The method `setReplacePatterns` corresponds to the cpmcopy parameter `-replacepatterns`. It replaces fill patterns with a color to reduce the file size in some cases, e.g. when input and output file have different resolutions and the fill patterns would have to be rendered to images otherwise.

```
page.setReplacePatterns(true);
```

To query the current state of replacing patterns you use the method `getReplacePatterns()`.

Merging Fonts

The method `setMergeFonts` corresponds to the cpmcopy parameter `-mergefonts`. It tries to minimize the amount of fonts used in a page by merging overlapping subsets into one font.

```
page.setMergeFonts(true);
```

To query the current state of merging fonts you use the method `getMergeFonts()`.

Merging Text

The method `setMergeText` attempts to merge separate text items together. This is useful to reduce the total number of items on a page and can also help with further page manipulations, see also *chapter 4.14.3. Selecting Items on page 134*.

```
page.setMergeText(Constants.Page.MergeTextPreset.MergeLines);
```

The method `setMergeText` takes one parameter, which consists of a series of flags that can be ORed together.

```
Constants.Page.MergeText.PreserveItemOrder
Constants.Page.MergeText.PreserveColor
Constants.Page.MergeText.SplitWords
Constants.Page.MergeText.PreserveCharacterSpacing
```

So, if you want to preserve the text color but want to split words (as opposed to merging all the items in a line), look at the example below.

```
page.setMergeText(Constants.Page.MergeText.PreserveColor | Constants.Page.MergeText.SplitWords);
```

Additionally, cpmill provides some predefined sets of flags.

```
Constants.Page.MergeTextPreset.MergeLines
Constants.Page.MergeTextPreset.PreserveAlignment
Constants.Page.MergeTextPreset.PreserveAlignmentWeak
Constants.Page.MergeTextPreset.PreserveVisualAspect
Constants.Page.MergeTextPreset.PreserveVisualAspectWeak
```

Note: The ...Weak sets include the `SplitWords` flag.

The usage of these options depends on your input file and what you actually want to achieve. With formats that tend to split words into separate text items for single characters or groups of characters, you may want to use `SplitWords` to merge the words back together (but still keep separate words). For additional processing of the document, applying `MergeLines` first may make things easier.

The method `getMergeText` returns the current status.

Merging Rectangles

The method `setMergeRects` is the equivalent of the cpmcopy parameter `-deldup`. Its main purpose is to reduce the number of rectangles on a page by trying to merge adjacent and overlapping rectangles. For some types of documents, this can result in a smaller amount of items on the page, so that the resulting documents are smaller (in terms of memory consumption).

```
page.setMergeRects(true);
```

The method `getMergeRects` returns the current status.

4.2.1.15. Simplex/Duplex Handling

To explicitly set the simplex and duplex mode for pages, use one of the following:

```
page.setSimplexDuplex(Constants.Page.SimplexDuplex.DuplexFront);
page.setSimplexDuplex(Constants.Page.SimplexDuplex.DuplexBack);
page.setSimplexDuplex(Constants.Page.SimplexDuplex.DuplexFrontShortEdge);
page.setSimplexDuplex(Constants.Page.SimplexDuplex.DuplexBackShortEdge);
page.setSimplexDuplex(Constants.Page.SimplexDuplex.Simplex);
page.setSimplexDuplex(Constants.Page.SimplexDuplex.Nothing);
```

Where `Nothing` is the default, i.e. simplex/duplex handling is not done in a special way (and cpmill's `"general.duplex"` setting will apply, if set).

The following constants but can be replaced with their numerical value:

- `page.setSimplexDuplex(0)`: no simplex/duplex handling
- `page.setSimplexDuplex(1)`: simplex mode
- `page.setSimplexDuplex(2)`: duplex mode, front page
- `page.setSimplexDuplex(3)`: duplex mode, back page
- `page.setSimplexDuplex(4)`: duplex mode, front page, short-edge binding
- `page.setSimplexDuplex(5)`: duplex mode, back page, short-edge binding

The method `getSimplexDuplex()`returns the current mode.

4.2.1.16. Jogging

Some printers support an option to stack sheets in an output tray with an offset from previously stacked sheets (jog). So, sheets belonging together can be identified more easily.

The method `toggleJog()` can used for the first page where the job shall start.

```
page.toggleJog();
```

In contrast to the deprecated method `setJog()` the jog status is activated until the next call of `toggleJog()`. Thus, you do not have to specify the call for each page but for new stacks of sheets only.

The method `getJogged` returns the current status of an input page: true for an new jog and false, if the job status did not change.

```
if (page.getJogged()) ...
```

You can use the method `clearJog()` to "unjog" a jogged input page.

4.2.1.17. Obfuscating Text

If you are dealing with sensitive customer documents, you may want to make some text unreadable while keeping the page layout intact. Three algorithms are provided to make the text unreadable: ROT13 encryption (`rot13`), randomly changing of letters (`random`) and changing letters and digits (`random.with.digits`).

Be aware of the fact that these are very weak encryption methods, aimed mainly at preventing accidental disclosure of a document's content. For example, the rot13 algorithm can be reverted by simply applying it twice. Also, `rot13` and `random` do not obfuscate digits. Furthermore, obfuscation does not work for documents where the text is stored in a custom encoding.

As with page rotation, see *chapter 4.2.4. Page Rotation on page 85*, there are two methods that apply text obfuscation, either on input, i.e. when reading the page, or on output, i.e. when writing the page.

```
page.setInTextObfuscation("rot13");
page.setTextObfuscation("rot13");
```

4.2.1.18. Removing Items

EraseRect

To remove items from within a rectangle on the page, you use the method `eraseRect`.

```
x = new Length("20mm");
y = new Length("125mm");
w = new Length("60mm");
h = new Length("35mm");
page.eraseRect(x, y, w, h);
```

By default, only items that are placed entirely inside the given rectangle are removed. You can control this behavior with a fifth parameter.

```
page.eraseRect(x, y, w, h, Constants.Page.RemoveMode.AllInside);
page.eraseRect(x, y, w, h, Constants.Page.RemoveMode.AllInsideNoBorder);
page.eraseRect(x, y, w, h, Constants.Page.RemoveMode.AllIntersect);
page.eraseRect(x, y, w, h, Constants.Page.RemoveMode.AllIntersectRaiseError);
```

- `AllInside`: default value used when the fifth parameter is omitted.
- `AllInsideNoBorder`: items will not be removed that share one or more edges with the rectangle.
- `AllIntersect`: all items are removed that intersect with the rectangle, i.e. even those items that are only partially within the rectangle.
- `AllIntersectRaiseError`: an exception will be raised when there are any items that are not fully within the given rectangle. This is useful to ensure that the given coordinates are correct and to not accidentally remove contents from a page.

RemoveItems

With the method `removeItems`, you can remove **all** items of one or more specified types from a page.

```
page.removeItems(Constants.Page.ItemType.Text);
```

As can be seen from the example below, item types can be ORed in case you want to remove more than one item type.

```
page.removeItems(Constants.Page.ItemType.Text | Constants.Page.ItemType.Image);
```

4.2.1.19. Overlays

Two methods are available that deal with external items, such as overlays, on the page:

- `readOverlays`
- `hasOverlays`

```
newpage = page.readOverlays();
```

This returns a new `Page` object where any references to overlays and other external items have been resolved. The original page remains unchanged. It is useful to perform operations on a page that should take overlays into account, e.g. data capturing, see also *chapter 4.14.1. Capturing Text on page 130*.

The method `readOverlays` will throw an exception when the page does not contain any overlays. You can use the method `hasOverlays` to check the page for overlay existence first.

```
if(page.hasOverlays()) ...
```

4.2.1.20. Resolution

Note: The default resolution will be 7200 dpi.

SetInitialResolution

You can change the resolution of a newly created, empty page with the method `setInitialResolution`.

```
page = new Page();
page.setInitialResolution(600);
```

You can specify different resolutions for x-coordinate and y-coordinate but they may not work with every MFF filter.

```
page.setInitialResolution(300, 600);
```

SetResolution

To set the output resolution, i.e. instruct the filter to use a specific resolution for the page, you use the method `setResolution`.

```
page.setResolution(1200, 1200);
```

The second parameter is optional.

GetResolution

The methods can be used to query the resolution separately for x-coordinate and y-coordinate.

```
xres = page.getXResolution();
yres = page.getYResolution();
```

SetFontResolution (PCL only)

To set the font resolution for a PCL output file, you use the method `setFontResolution`.

```
resolutionPreferred = new Resolution(300, 300, "inch");
..
page.setFontResolution(Resolution resolutionPreferred, Resolution resolutionMin,
                       Resolution resolutionMax);
```

The last two parameters are optional. The PCL standard allows 600 and 300 dpi for font resolution.

SetRasterResolution (PCL only)

To set the raster resolution for a PCL output file, you use the method `setRasterResolution`.

```
resolutionPreferred = new Resolution(300, 300, "inch");
..
page.setRasterResolution(Resolution resolutionPreferred, Resolution resolutionMin,
                         Resolution resolutionMax);
```

The last two parameters are optional. The PCL standard allows 600, 300, 200, 150, 100 and 75 dpi for raster resolution.

4.2.2. Applying ICC Profiles

To apply an ICC profile to an individual page, you can code the following:

```
page.applyIccProfile("AdobeRGB1998.icc");
```

If you want to apply the ICC profile to an entire file, you can use the method described in *chapter 4.13.5.2. OutputDocumentAttributes on page 121, which* is more efficient.

To check whether an ICC profile is being applied to a page, you can code the following:

```
if(page.appliesIccProfile()) ...
```

You can also get the file name of the ICC profile by using

```
filename = page.getAppliedIccProfile();
```

Note: This method will return an empty string if no ICC profile has been set to be applied yet. ICC profiles to be inherited from `OutputDocumentAttributes` will not be returned here since the inheritance will only happen when the page is actually written to a file.

4.2.3. Format-Specific Operations

4.2.3.1. File Offset (AFP only)

The method `getFileOffset()` returns the physical offset of a page within the file (in bytes relative to the start of the file).

```
offset = page.getFileOffset();
```

Currently, the method supports AFP files only. It will throw an exception when used with an unsupported format or when used on a page that was not read from a file.

Note: JavaScript does not have a native 64-bit integer type. Instead, numerical values in JavaScript are stored as type `Number`, which is either a 32-bit integer or a double (floating-point value).

Due to technical reasons, the file offset is effectively limited to a 53-bit integer, which is roughly equivalent to 9000 Terabyte and therefore should not really be a problem for the near future.

4.2.3.2. Page Name (AFP only)

The method `getPageName()`returns the AFP page name.

```
pageName = page.getPageName();
```

Currently, the method supports AFP files only. It will return an empty string when used with an unsupported format or when used on a page that was not read from a file.

4.2.3.3. PJL Commands (Printer Job Language)

PJL commands are used by PCL, PXL (= PCL 6), PSB, and PostScript. While Compart's software will usually create the necessary PJL commands automatically, there are situations where customers need to inject specific or proprietory PJL commands into the document stream. In that case, you do the following:

```
page.addPjlCommand("@PJL SET QTY=42");
```

The method `addPjlCommand` can be called more than once to inject multiple PJL commands. The provided string is used verbatim and should always start with `@PJL`.

Notes:

- Injecting PJL commands is for advanced users only. You should know what you're doing! You should use it only if the functionality is not currently supported by Compart software.
- The use of PJL commands may interfere with the PJL and PCL commands created by Compart's software and may change the output result.
- There is no guarantee regarding where in the document stream these PJL commands will be injected. They may appear before or after PJL commands created by Compart and this position may change without notice.

4.2.4. Page Rotation

```
page.setRotation(90);
```

The page rotates by 90 degrees.

When you want to make sure that a page has a certain orientation, e.g. portrait mode, you do the following:

```
page.setRotationMode(Constants.Page.RotationMode.Portrait);
```

The difference between the method `setRotation` and `setRotationMode` is that when using `setRotation`, the page is **always** rotated by the specified value, while `setRotationMode` only rotates pages that do not already have the specified orientation.

In some cases, you need to rotate the page **before** any other modifications (putting other Docponents on it, etc.) are made to it. In that case you do the following:

```
page.setInRotationMode(Constants.Page.InRotationMode.Portrait);
```

It is possible to use **both** methods `setInRotationMode` and `setRotation` / `setRotationMode` on the same page.

4.2.5. Processing Order

The following description clarifies in which order the various page modification operations are applied to a Page object.

4.2.5.1. In Operation

When a page is rendered, the first operations applied to it are those that are performed on page input, i.e.

- `setInRotation`
- `setInRotationMode`
- `setInShift`
- `setInTextObfuscation`

The order of the “In” operations themselves depends on the order in which the operations were applied in the profile, i.e. if `setInShift` was called before `setInRotationMode`, and then the shift operation would be performed first.

4.2.5.2. Put Operation

Next, all objects that have been placed onto the page using the method `put` will be rendered onto the page. The order of the operations depends on their order in the profile.

This is a recursive operation, i.e. if you put other pages onto the current one, those other pages will be rendered *on their own* first, according to the same rules as described here.

4.2.5.3. Variable Replacement

Next, any variables in the page will be replaced. **Note:** Variables contained in objects that have been put onto the page have already been replaced in the previous step.

4.2.5.4. Other Page Modifications

Any remaining page operations are applied now, e.g. scaling, conversion to image, rotation (without the “In”), etc. **Note:** The exact order of these operations is not defined. In other words, different releases may apply these modifications in different orders.

4.2.5.5. Operations on Page Write

Finally, when the page is being written, the following operations are applied:

- `setSeparateOverlays`
- `setPpmlFile`

4.2.6. AFP Copy Group

Copy Groups are specific to AFP, so the methods described here are only available when either the input or the output format is AFP. However, they are also available in AFP Mixed Mode (MMD) and when using the MFF filter SAPGOF together with the Formdef Processor (FDP). For other formats, the methods will be silently ignored or will return default values, e.g. empty copy group names.

The copy group is an attribute of the page:

```
print("Current copy group: " + page.getAfpCopyGroup());
```

Several pages can be in the same copy group. Therefore, they will all return the same copy group name. However, multiple copy groups can also have the same name.

To detect the first page of a copy group, you can code the following:

```
if(page.afpCopyGroupHasChanged())
    {
    print("Copy group has changed!");
    }
```

The name of the copy group can be changed:

```
page.setAfpCopyGroup("NEWGROUP2");
```

Changing the name of the copy group does **not** automatically start a new copy group in the output, though. You have to explicitly signal the start of a new copy group by calling:

```
page.changeAfpCopyGroup(true);
```

4.2.7. Horizontal and Vertical Image Flip

The method `setFlipImageHorizontal()` and `setFlipImageVertical()` rotate images of a page at its horizontal and vertical axis. You can combine the two methods to flip the images horizontally and vertically at the same time.

```
page.setFlipImageHorizontal();
```

```
page.setFlipImageVertical();
```

4.2.8. Reducing Image Colors

The method `setReduceImageColors()` converts RGB images to palette images in order to use the method `mapImageColorsToName()` in a subsequent processing step. You specify the number of colors that the resulting image should contain. **Note:** Currently, CMYK images are not supported. .

```
page.setReduceImageColors(12);
```

4.2.9. Mapping Image Colors to a Name

The method `mapImageColorsToName()` maps a defined color (HSB color space) to a name. The three values are specified separated by commas: color angle, threshold angle, and color name to be mapped to the color. **Note:** Currently, CMYK images are not supported.

```
page.mapImageColorsToName(12, 20, "blue");
```

4.3. Tray

4.3.1. Tray Setting

You can set both the **input tray** (i.e. the source of the paper to print on) and the **output tray** (i.e. where to put the paper after it has been printed on):

```
page = new Page();

page.setPaperInTray("UpperTray");

page.setPaperOutTray("LowerTray");
```

Note: The tray names must be defined in the profile of the MFF filter you are using, e.g. `mffafp.pro`, `mffpcl.pro`, etc.

The equivalent `get` methods are:

```
intray = page.getPaperInTray();

outtray = page.getPaperOutTray();
```

4.4. Index

4.4.1. Index Object

The Index object provides access to the index information (TLEs in AFP) of an object, e.g. a page.

The Index object works like a native JavaScript Array object. Therefore, you can use all the JavaScript Array methods (push, pop, etc.) on it.

```
// create an Index, add entries
idx = new Index();

entry = new String("Value of Index Entry");
entry.name = "Name of Index Entry";

idx.push(entry);
```

```
// get index from a page
// Note: Any changes made to idx1 will apply to the page's index
// as well, i.e. 'idx1' acts like a pointer
idx1 = page.getIndex();
```

```
// set new index for a page, replacing the page's existing index
idx2 = new Index();
p2 = new Page();
p2.setIndex(idx2);
```

```
// copy input pagegroup indexes and add extra index
pagegroup = page.getGroup();
currentIndex = pagegroup.getIndex();
tle004 = new String("value004js");
tle004.name = "name004js";
tle004.sequence = 3;
tle004.level    = 4;
currentIndex.push(tle004);
pagegroup.setIndex(currentIndex);
output.push(page);
```

```
// ignore input pagegroup indexes and add extra index
pagegroup = page.getGroup();
newIndex = new Index();
tle004 = new String("value004js");
tle004.name = "name004js";
tle004.sequence = 3;
tle004.level    = 4;
newIndex.push(tle004);
pagegroup.setIndex(newIndex);
output.push(page);
```

The Index object also supports two additional properties: `.sequence` and `.level`, representing a sequence number and a level number (aka Triplet X'80'):

```
entry = new String("Value of Index Entry");
entry.name = "Name of Index Entry";
entry.sequence = 12345678;
entry.level = 42;
```

Note: Both `.sequence` and `.level` are numeric properties. Their valid range is `0x00000000` through `0x7fffffff`.

So the setting `entry.sequence = 0xfffffffa;` will result in platform specific output. The properties are only supported for AFP and XIF input and output.

4.5. Comments

Comments (NOPs in AFP) provide a way to add non-printing information to a document.

Since comments can exist both before a page (i.e. in the data stream) and on a page, there are two ways of adding comments to a page, depending on where you want them to appear.

4.5.1. Comment

The JavaScript representation of a comment is a `Comment` object.

```
c = new Comment("My comment");
```

4.5.1.1. Binary Comments

The object `Comment` can hold either textual or binary content (the above example creates a text comment). To create a binary comment, you need a `Bytearray` object that holds the binary content (i.e. one byte per array element). You would normally read those from a file (using `FileInputBinary`, for example) or get them as part of the page anyway.

The following example manually creates a bytearray for demonstration purposes:

```
bin = new Array(1, 2, 3);

bc = new Comment();
bc.setDataBinary(bin);
```

Note: Currently, binary comments are supported for comments that go on a page, not for comments before a page, i.e. in the data stream.

To check whether a `Comment` object contains textual or binary data you use the following functions:

```
if(c.isString())
    {
    print("This comment is a string and contains: ");
    print(c.getString());
    }

if(c.isBinaryData())
    {
    print("This comment contains binary data");
    }
```

To get the contents of a textual comment (as a JavaScript string), use the method `getString()`. To get the contents of a binary comment you can use the method `getDataBinary`, which will return a bytearray.

4.5.1.2. Comments before Pages

The `Comments` object provides access to comments within the data stream. Either you can create a new `Comments` object or you can get access to the page's existing `Comments` object and modify it.

```
c = page.getComments();

print("This page has " + c.length + " comment(s)");
```

As with the `Index` object, a `Comments` object is a JavaScript array that holds all the page's comments. The contents of that array are `Comment` objects as explained above. In addition, since the `Comments` object is a fully working JavaScript array, you can add and remove comments just like you add and remove items from an array.

To attach your own comments to a page, create a new `Comments` object, add your comments to it, and attach it to the page.

```
comments = new Comments();

c1 = new Comment("My New Comment");
comments.push(c1);
// etc.

page.setComments(comments);
```

4.5.1.3. Comments on Pages

To put a comment on a page, use the page's `put` method.

```
c = new Comment("My comment");
page.put(c);
```

Note: Currently, binary comments are supported on pages, so this is the only way to add a binary comment to a document.

4.6. Group

4.6.1. Group Object

The object `group` provides access to the AFP page group information.

```
group = page.getGroup();

print(group.getName());
index = group.getIndex();
```

An index number is assigned to groups when being read from a file. Currently, you only have read access to that information:

```
print(group.getNumber());
```

Changing the name of a group will start a new AFP page group when that page is written.

```
group.setName("NEWNAME2");
```

You can also replace a page's group information entirely by setting a new group.

```
group = new Group();
page.setGroup(group);
```

Note: When changing the `Index` object, you currently have to call explicitly `setIndex` again afterwards so that the modified index information is copied back. This will be resolved at a later stage...

```
index = group.getIndex();

i = new String("NewIndexValue");
i.name = "NewIndexEntry";
index.push(i);

group.setIndex(index); // required
```

4.7. Barcode

Barcodes are available as JavaScript objects and can be freely positioned on pages.

```
barcode = new Barcode();

barcode.setDataString("978159059389");
barcode.setType("EAN 13", false, 1);
barcode.setMetrics(new Length("30mm"), new Length("10mm"), 3);
barcode.setRotation(90);
barcode.showHRI(true);

page.put(barcode, new Length("12mm"), new Length("3mm"));
```

- `setDataString` is used to set the actual data encoded by the barcode.
- `setType` defines the barcode type whether to use a quiet zone (`true`/`false`) and the number of check digits. The second and third parameters are optional. In addition, the "quiet zone" flag is not really used for the barcode and setting it to `true` will log a warning message.
- `setCheckDigits` is an alternative to above the method.

```
barcode = new Barcode();
barcode.setType("EAN 13");
barcode.setCheckDigits(1);
```

- `setMetrics` defines the width, height, and ratio.
- `setWidth`, `setHeight`, `setRatio` are an alternative to above the method. The three values can be set separately.

```
barcode.setWidth(new Length("30mm"));
barcode.setHeight(new Length("10mm"));
barcode.setRatio(3);
```

- `setRotation` defines the rotation in degrees. You do not have to call this function, if the barcode does not have to be rotated.
- `showHRI` shows (`true`) or hides (`false`, default) the human readable interpretation (HRI), e.g. the digits below a standard EAN barcode.
- The human readable interpretation will be displayed using a default font, unless you explicitly set a font using `setFont()`.

4.7.1. Barcode Types

Barcodes supported by Compart software products, see *Appendix D: Barcodes on page 284*.

The following methods are available for DataMatrix codes.

```
barcode = new Barcode();
barcode.setDataMatrixConvertToBCD(true);
barcode.setDataMatrixDimensions(2, 2);
barcode.setDataMatrixEncoding(Constants.Barcode.DataMatrixEncoding.ASCII);
```

The following encodings are supported for DataMatrix:

- Constants.Barcode.DataMatrixEncoding.ASCII
- Constants.Barcode.DataMatrixEncoding.Base256
- Constants.Barcode.DataMatrixEncoding.C40

For binary data (Base256 encoding), use `setDataBinary` instead of `setDataString`. The method expects a bytearray, i.e. a JavaScript array where each array element represents one byte. Bytearrays can be read and written using `FileInputBinary` and `FileOutputBinary` or you can create them manually:

```
data = new Array(1, 2, 3, 4); // bytearray of 4 bytes
barcode.setDataBinary(data);
```

4.7.1.1. DataMatrix Rows and Columns

The dimensions given in the method `setDataMatrixDimensions` are in the order: rows, columns. Alternatively, you can also use the methods `setDataMatrixRows` and `setDataMatrixColumns` to set the dimensions separately.

Get Methods

The `get` methods `getDataMatrixRows` and `getDataMatrixColumns` are available.

4.7.2. “Raw” Barcodes

For more fine-grained control over the contents of the barcode, there is a special mode for Code 128 barcodes were you provide the actual barcode reference numbers.

```
barcode = new Barcode();
data = new Array(35, 100, 79, 77, 80, 65, 82, 84);
barcode.setRawData(data);

barcode.setType("Code 128A", false, 1);
barcode.setMetrics(new Length("30mm"), new Length("6mm"), 3);
```

The above example switches from Code 128A to 128B after the first character: `35` is an uppercase ’C’, `100` is the code to switch to Code 128B, and the following are the codes for lowercase characters that do not exist in 128A.

Notes:

- It only works with Code 128 (A, B, or C) barcodes.
- You will either have to specify exactly which code to use (e.g. `Code 128A`, as in the above example) or you will have to add the start code in the data array (e.g. specify `Code 128` and start the data with 103, which is the start code for Code 128A).
- HRI are **not** supported for this variation. If you need a HRI text, you will have to use a Text item and add it yourself.

4.7.2.1. Get Methods

The following `get` methods for `setRawData` are available.

```
if(barcode.hasRawData()) ...
data = barcode.getRawData();
```

4.7.3. HRI Positioning

The position of the HRI can be set and changed.

```
barcode.setHRIPosition(Constants.Barcode.HRIPosition.Above);
```

Three positions for HRI are available:

- `Below` displays the HRI below the barcode. It is the default setting.
- `Above` displays the HRI above the barcode. **Note:** It may not work with some barcode types.
- `Default` is a special value needed only for some AFP BCOCA barcode types.

4.7.3.1. Get Method

The following `get` methods for `setHRIPosition` are available.

```
position = barcode.getHRIPosition();
```

4.8. OMR

OMR marks are available as JavaScript objects and can be freely positioned on pages.

```
omr = new Omr();

omr.setDataString("11 11  11111111111111 ");
omr.setMetrics(new Length("24pt"), new Length("18tw"), new Length("12pt"));
omr.setLocation(Constants.Omr.Location.Top);

page.put(omr, new Length("12mm"), new Length("3mm"));
```

- `setDataString` is used to set the actual data encoded by the OMR.
- `setMetrics` defines the height, line width, and distance (in that order).
- `setLocation` can be used to position the OMR mark on the top, bottom, left, or right border of a page.

The three metrics parameters can also be set separately:

```
omr.setHeight(new Length("24pt"));
omr.setLineWidth(new Length("18tw"));
omr.setDistance(new Length("12pt"));
```

4.8.1. Get Methods

The following `get` methods for `omr` are available.

4.8.1.1. Metrics

The three metrics parameters listed be can be queried separately and are returned as Length objects, see also *chapter 4.11. Length on page 108*.

```
omr.setMetrics(new Length("24pt"), new Length("18tw"), new Length("12pt"));

height    = omr.getHeight();    // 24pt
lineWidth = omr.getLineWidth(); // 18tw
distance  = omr.getDistance();  // 12pt
```

4.8.1.2. Data String

Note: It will throw an exception, if the data string has not been set or is empty.

```
data = omr.getDataString();
```

4.8.1.3. Location

```
if(omr.getLocation() == Constants.Omr.Location.Top) ...
```

4.8.1.4. NoLine Character

The character that represents the absence of an OMR line in the OMR data string, can be configured (previously hard-coded to a blank, ' ').

```
omr.setNoLineCharacter('0');

if(omr.getNoLineCharacter() == '0')
    {
    omr.setDataString("001100110011");
    }
else
    {
    omr.setDataString("  11  11  11");
    }
```

4.8.2. Whiteout Rectangle

If you need a rectangle to cover up blotches, punch holes, etc. on scanned pages before applying the OMR, you can use the method `Rect`, see also *chapter 4.10.1.6. Rect on page 105*.

But for the sake of completeness, the example below shows the syntax of the method `setWhiteOutRect`.

```
omr.setWhiteoutRect(new Length("0mm"), new Length("0mm"), new Length("1cm"),
                    new Length("12mm"));
```

Note: The coordinates of the rectangle were independent of the actual OMR coordinates and rotation also did not apply.

4.9. Bookmark and TOC

Bookmarks are currently only supported for PDF output files, i.e. they cannot be read and no other format can handle them!

A **TOC** (Table Of Contents) object holds a collection of Bookmark objects. The **Bookmark** itself consists of a label (a short description that will show up in the PDF viewer, e.g. Adobe Reader), a page number, and an optional position on the page.

4.9.1. Bookmark Generation

For a Bookmark, the label and page number are mandatory.

```
b = new Bookmark("My Label Text", 42);
```

In the above example, the Bookmark will associate with page 42 of the document. So, when the user clicks on the label text (e.g. in Adobe Reader), it will take them to page 42.

4.9.2. Position

You can also associate a bookmark with a certain position on the page. To do that, simply put it onto the page.

```
page.put(b, new Length("0"), new Length("10cm"));
```

Now, when you click on the bookmark, the reader is taken to a position 10cm from the top of the given page. **Note:** When you code `page.put(b)`, i.e. a put without any coordinates, it will not do anything.

4.9.3. Child Bookmarks

You can build entire hierarchies of bookmarks by appending one bookmark onto another.

```
b2 = new Bookmark("My Sub-Bookmark", 42);
b.appendChild(b2);
```

The new bookmark will show up as a node below the parent bookmark.

Child bookmarks do not have to be on the same page as the parent bookmark. That way, you can have a parent bookmark point to the start of a chapter while the child bookmarks point to subsections of that chapter, starting on subsequent pages.

4.9.4. Get Methods

The following `get` methods are currently available for bookmarks:

- `getText()`: It returns the bookmark's label text.
- `getPageNumber()`: It returns the page number the bookmark is associated with.
- `getPositionX()` and `getPositionY()` return the x-coordinate and y-coordinate of the bookmark on a page (as a Length).
- `hasChildNodes()`: It indicates whether a bookmark has any child bookmarks appended to it or not.

4.9.5. TOC Generation

For Bookmarks to be associated with a document, you need to create a TOC object and set it as the document's table of contents.

```
toc = new TOC();
```

All the parent bookmarks will have to be appended to the TOC.

```
toc.appendChild(b);
```

And finally, if set the TOC for the output document, you use either

```
output.setTOC(toc);
```

for the standard output queue or

```
outFile = new FileOutputDocument(...);
...
outFile.setTOC(toc);
```

for separate output files.

For `FileOutputDocuments`, you will have to set the TOC before the `close()`.

In a cpmill profile, the `documentend` or `endprocess` event handlers are probably the best spots to set the TOC.

4.9.5.1. Bookmark and TOC Sample Code

```
outAttr = new OutputDocumentAttributes();
outAttr.setType("PDF");
outFile = new FileOutputDocument("example.pdf", outAttr);

toc = new TOC();

b = new Bookmark("First Bookmark", 1);

b2 = new Bookmark("Second Bookmark", 1);
page.put(b2, new Length("0"), new Length("10cm"));
b.appendChild(b2);

b3 = new Bookmark("Third Bookmark", 1);
page.put(b3, new Length("0"), new Length("20cm"));

toc.appendChild(b); // and, implicitly, b2
toc.appendChild(b3);

outFile.write(page);

outFile.setTOC(toc);

outFile.close();
```

This example creates three bookmarks (`b`, `b2`, `b3`) for a page. Bookmark `b2` is a child of bookmark `b` and located 10cm down the page. Bookmark `b3` is independent of the other two and located 20cm down the page.

4.10. Items

4.10.1. Docponent Item Subclass

Note: The old names (ending in `...Item`) should be considered deprecated as of release 200708. They will continue to work for the time being, but new features will only be added to the objects without the `...Item` suffix.

4.10.1.1. Font

```
font = new Font("Courier", new Length("12pt"),
                   Constants.Font.FontWeight.Medium,
                   Constants.Font.FontStyle.Upright);
```

`Fonts` items can be reused, so if you need a font more than once, create it only once (e.g. in the `beginprocess` event handler).

Note: All the constructor's parameters are optional. Alternatively, you can do the following:

```
font = new Font();
font.setName("Courier");
font.setSize(new Length("12pt"));
font.setWeight(Constants.Font.FontWeight.Medium);
font.setStyle(Constants.Font.FontStyle.Upright);
font.setSerif(Constants.Font.FontSerif.SansSerif);
font.setSpacing(Constants.Font.FontSpacing.Proportional);
```

- **Font Weight**: Values for `setWeight` are: `Ultralight`, `Extralight`, `Light`, `Semilight`, `Medium`, `Semibold`, `Bold`, `Extrabold`, `Ultrabold`.
- **Font Style**: Values for `setStyle` are: `Upright`, `Italic`.
- **Serifs**: Values for `setSerif` are: `SansSerif`, `Serif`.
- **Spacing**: Values for `setSpacing`: `Monospaced`, `Proportional`.

Get Methods

The following `get` methods are available.

```
name = font.getName();
size = font.getSize(); // as a Length object
style = font.getStyle();
serif = font.getSerif();
spacing = font.getSpacing();
weight = font.getWeight();
```

Font Embedding and Font Subsetting

The following methods about font embedding and font subsetting are available:

```
isEmbeded = font.isEmbedded();
isSubset = font.isSubset();
```

Not all formats support the methods. When a method is called for an unsupported format, an exception is thrown.

Device Specific Name

The following method is available to query the “device-specific” name of a font.

```
name = font.getDeviceSpecificName();
```

As the function name implies, the actual value returned depends on the format being used. For AFP, this function can be used to get the AFP Character Set Name, e.g. C0T055A0 which should not be- not be confused with a character set like UTF-8 or ISO-8859-1.

Note: The function may return an empty string, i.e. it will not throw an exception when this property is not set or not available in the current context.

4.10.1.2. Color

```
color = new Color();
```

Note: The Color’s constructor does not take any arguments. To set a color, use one of the following methods:

```
color.setRGB(0xFF, 0x00, 0x7f);
color.setCMYK(100, 100, 100, 100);
color.setGray(33); // in percent
color.setName("Blue", 100); // name + intensity
```

After a color has been set, the color object will have properties containing the current color values.

```
js> color.setRGB(255, 0, 128);
js> color.red
255
```

There are also methods available to check which type of color the object currently holds as well as methods to compare colors:

```
js> color.isRGB();
true
js> color.isCMYK();
false
```

Additionally, there are also methods `isGray()` and `isName()`.

For a simple check to see if a `Color` object represents black or white, use:

```
if(color.isBlack()) ...
if(color.isWhite()) ...
```

Note: These two methods will not return meaningful results for colors set by name.

To compare two `Color` objects use the method `isEqual`:

```
if(color1.isEqual(color2)) ...
```

The method `isEqual` only works for colors of the same color space, though. When two colors from different color spaces (e.g. RGB and grayscale) are compared, you will always get a negative result, even if they do happen to represent the same color.

```
color1 = new Color();
color1.setRGB(0, 0, 0);

color2 = new Color();
color2.setGray(100);

if(color1.isBlack()) log("Color 1 is black");
if(color2.isBlack()) log("Color 2 is black");
if(color1.isEqual(color2))
    {
    log("Colors are equal");
    }
else
    {
    log("Colors are NOT equal!");
    }
```

The above example issues the following information:

```
Color 1 is black
Color 2 is black
Colors are NOT equal!
```

Note: Alpha transparency is not currently supported!

4.10.1.3. Text

A `Text` item is a piece of text that can be freely placed on a page, see also *chapter 4.10.1.1. Font) on page 101*.

```
txt = new Text("Hello, world!", font);
page.put(txt, new Length("1cm"), new Length("5cm"));
```

The method `get` and `set` are available:

```
currentText = txt.getText();
```

```
txt.setText("New item text");
```

The method `setText` is mainly meant to be used for selecting items, see *chapter 4.14.3. Selecting Items on page 134*.

The method `getFont` is available:

```
font = txt.getFont();
```

Note: This font cannot be changed, if the Text item was returned from the method `select`.

4.10.1.4. ExternalItem

An `ExternalItem` is used to embed some external object into a page. Available external items types are `Overlay`, `Segment`, `Macro`, and `Object`.

```
ext = new ExternalItem("somename", Constants.ExternalItem.Type.Object);
ext.setMimeType("application/pdf");
```

Note: `Overlay`, `Segment`, and `Object` are expressions of AFP, while `Macro` is used for PCL.

Overlays

The most common use case is to use an `ExternalItem` of type `Overlay` to refer to an overlay in an AFP datastream.

```
ext = new ExternalItem("OVERLAY1", Constants.ExternalItem.Type.Overlay);
page.put(ext);
```

As written in the above example you expect an overlay named `OVERLAY1` to exist in an available AFP resource. The overlay will be attached to the given page.

Another use case is to create a new overlay from an existing document page. For example, you can read a page from a PDF file and use the page as an AFP overlay.

```
file = new FileInputDocument("stamp.pdf");
p = file.readPage(1);
file.close();

ext = new ExternalItem("OVERLAY2", Constants.ExternalItem.Type.Overlay);
ext.attachOverlay(p);
page.put(ext);
```

As result of the example above a new overlay `OVERLAY2` is created from the PDF page on-the-fly.

In case of selfcontained AFP output, overlays will only be recognized when they exist in the **input** datastream. If they only exist in an external AFP resource but you want to create a selfcontained AFP file, you will need to explicitly read the AFP resource and attach the overlay, using the same method as with the PDF page in the above example.

Note: AFP overlays contain text, image, graphics, and barcode data, which can be reused/included in several pages. AFP page segments are normally just images. AFP is able to include single page PDF by using external item type `Object` and will be mapped to an AFP object container.

Objects

External Items of type `Object` are used to embed objects into the datastream. In this case, you should specify the MIME type of the embedded object.

```
ext = new ExternalItem("somename", Constants.ExternalItem.Type.Object);
ext.setMimeType("application/pdf");
```

The use of the method `setMimeType` is not required for the other object types.

Get Methods

The following `get` methods for `ExternalItem` are available.

```
extName = ext.getName();
mimeType = ext.getMimeType();
```

```
if(ext.getType() == Constants.ExternalItem.Type.Overlay) ...

if(ext.hasAttachedOverlay())
    {
    overlay = ext.getAttachedOverlay();
    }
```

4.10.1.5. Comment

See *chapter 4.5. Comments on page 90*.

4.10.1.6. Rect

Simple rectangles can be created using the `Rect` item.

```
rect = new Rect(new Length("10cm"), new Length("3cm"));
page.put(rect, new Length("5cm"), new Length("2cm"));
```

Note: There is currently no way to set a fill pattern for a `Rect` item.

Get Methods

The following `get` methods for `Rect` are available.

```
w = rect.getWidth();
h = rect.getHeight();
```

4.10.1.7. Line

Lines can be drawn using the `Line` item.

```
l = new Line();
l.setWeight(2);
l.setLineStyle(Constants.Line.LineStyle.Solid);
l.setVector(new Length("10cm"), new Length("10cm"));
page.put(l, new Length("1cm"), new Length("1cm"));
```

Note: `setVector` does not define the line's end point put rather the vector of the line. The line's starting point is defined by the coordinate where the line is `put` onto the page.

Available line styles are `Solid`, `Dotted`, `Dashed`, `Double`, `Groove`, `Ridge`, `Inset`, and `Outset`.

Get Methods

The following `get` method for `Line` is available.

```
if(line.getLineStyle() == Constants.Line.LineStyle.Dotted) ...

w = line.getWeight();
```

The vector is returned with separate values for the x-coordinate and y-coordinate.

```
vx = line.getVectorX();
vy = line.getVectorY();
```

4.10.1.8. Annotation

Note: Currently, annotation type Text for text annotations is implemented.

```
ann = new Annotation(Constants.Annotation.AnnotationType.Text);
ann.setTitle("Author");
ann.setText("Reiner User");
page.put(ann);
```

To hide an annotation, you do the following:

```
ann.setHidden(true);
```

Get Methods

The following `get` methods for `Annotation` are available.

```
annotationText = ann.getText();
annotationTitle = ann.getTitle();
if(ann.getType() == Constants.Page.ItemType.Text) ...
```

```
annHidden = ann.getHidden();
if(ann.isHidden()) ...
```

`getHidden()` and `isHidden()` are equivalent. Using `isHidden` may make the code more readable, though.

4.10.1.9. Common Methods

The following methods can be applied to all of the above items, although they may not always make sense for the specific item.

Set Rotation

```
item.setRotation(90);
```

Note: While the degree to rotate the item by can actually be specified as a float (double), not all drivers support this yet (e.g. AFP only supports 90, 180, and 270 degrees at the moment).

Get Rotation

A common `getRotation` method is available.

```
rotation = item.getRotation();
```

Set Color

After you specified a color, see *chapter 4.10.1.2. Color on page 102*, you assign it to an Item.

```
item.setColor(color);
```

Get Color

The common method `getColor` is available.

```
color = item.getColor();
```

To actually change the `Color` of an item, you have to use `setColor` again.

Set Transparency

```
item.setTransparency(true);
```

Note: The method `setTransparency` can only be used for file formats that support transparency. The default is `false`.

The transparency type can be further specified with the blending modes from `Constants.DocponentItem.BlendingMode`.

```
item.setTransparency(true, Constants.DocponentItem.BlendingMode.Multiply);
```

When the blending mode is omitted, `Constants.DocponentItem.BlendingMode.Multiply` is taken as default.

Note: Not all blending modes are supported by all formats and all item types. Some combinations will force an internal rasterization enlarging the output files. Others are ignored by filters. Make sure your selected blending mode does what you want by analyzing your output.

Get Transparency

There are two common methods available to check for transparency.

```
transparent = item.getTransparency();

if(item.isTransparent()) ...
```

The mthods `getTransparency()` and `isTransparent()` are equivalent. Using the method `isTransparent` can make the code more readable in some situations, though.

Set Position

Usually, the position of an Item is specified when you put it onto a page, see *chapter 4.2. Page on page 72*. However, if you want to change the position later or when selecting items, you can use the method `setPosition`.

```
item.setPosition(new Length("5cm"), new Length("3cm"));
```

The first parameter is the x-coordinate and the second parameter is the y-coordinate.

Get Position

You can query the position of an item separately for its x- coordinate and y-coordinate.

```
posX = item.getPositionX();
posY = item.getPositionY();
```

The result, in both cases, is a Length object, see also *chapter 4.11. Length on page 108*.

4.11. Length

The `Length` object provides a way to specify width or height values using natural value and unit pairs like `"15cm"`, see *Appendix E: Units of Measurements on page 285*.

4.11.1. Basic Usage

You can either create variables that hold the new length or you can also pass new objects directly, see example below.

```
width = new Length("15cm");

page = new Page(new Length("8in"), new Length("11in"));
```

4.11.2. Arithmetic

The `Length` class provides only basic methods for arithmetic. For more advanced calculation, you can always use the methods built into JavaScript itself.

4.11.2.1. Addition and Subtraction

An `add` method for addition is available.

```
js> len = new Length("12cm");
12cm
js> len2 = new Length("3cm");
3cm
js> len.add(len2);
js> len
15cm
```

A `sub` method for subtraction is available.

```
js> len.sub(len2);
js> len
12cm
```

To minimize rounding errors, both methods will try to keep the same unit for as long as possible. However, when you add length values with two different units, it will fall back to tcm (thousandths of a centimeter).

```
js> len
12cm
js> len.add(new Length("1in"));
js> len
14540tcm
```

You should be aware that both `add()`and `sub()`do change the value of the object and do **not** return a value.

Furthermore, you cannot keep the original object unchanged by making a copy of it like in the example below.

```
height = page.getHeight();
temp = height;
temp.sub(new Length("5cm"));
printf("height=" + height + ", temp=" + temp);
```

You will find that **both** `height` and `temp` will have been changed. This is due to JavaScript only keeping pointers (references) to objects instead of making copies.

4.11.2.2. Alternative Methods

The Length class provides additional methods to return the sum or difference of two `Length` objects without changing their value.

```
a = new Length("12cm");
b = new Length("5cm");

c = a.sum(b);
print(c); // prints "17cm", 'a' remains unchanged
```

```
a = new Length("12cm");
b = new Length("5cm");

c = a.diff(b);
print(c); // prints "7cm", 'a' remains unchanged
```

4.11.3. Comparing Length Values

To compare values you use the method `isEqual`.

```
js> len = new Length("72pt");
72pt
js> len.isEqual(new Length("1in"));
true
```

Additional methods are available, as well.

```
if(a.isLessThan(b)) { print("a &lt; b"); }

if(a.isGreaterThan(b)) { print("a &lt; b"); }
```

4.11.3.1. Other Operations

For more advanced mathematical operations, you can do the following

- Use the methods `getValue` and `getUnit` to get a length's value and unit string.
- Do the calculation in JavaScript then and concatenate the result with the original unit string to get a new `Length` object.

```
js> v = len.getValue();
72
js> u = len.getUnit();
pt
js> x = Math.floor(v * 1.7 / 3);
40
js> vnew = new Length (x + u);
40pt
```

getValue with Conversion

The method `getValue` optionally accepts a parameter: If you pass a unit, it will convert the length's value into the value for that unit.

```
js> a = new Length("1in");
1in
js> a.getValue("pt");
72
```

4.11.4. Decimal Values

Decimals values are supported.

```
size = new Length("11.5pt");
```

Note: In earlier releases, decimal values were silently discarded and the value was treated as an integer, i.e. `1.5mm` was really only `1mm`.

4.11.4.1. Rounding Control

Note: When displaying a `Length` value, the value is rounded to a three decimal points independent of the actual value that is used internally.

```
js> l = new Length("0.125in");
0.125in
js> l.getValue();
0.125
```

4.12. Vars (Varpool)

The JavaScript `Vars` object is a representation of the Compart `varpool` object. Its main use is for replacing variables in pages, see *chapter 4.2.1.7. Replacing Variables on page 75*.

Note: For those who are familiar with the "classic" cpmill profiles: All the variables, that were preceded by a `$` character there, are available in JavaScript as well by simply replacing the character with `sys`.

4.12.1. Creating a Varpool

You can either create an empty varpool or create one that inherits from the global `sys` varpool:

```
myvars = new Vars(); // empty varpool
sysvars = new Vars(sys); // contains a copy of the 'sys' varpool
```

Note: Use `new Vars(sys)`to create a copy of the `sys` varpool.

4.12.2. Adding Variables

To add new variables as properties, you do the following:

```
myvars = new Vars();

myvars.MyNewVariable = 42;
myvars.MyNewStringVariable = "Hello, world!";
```

Note: The `sys` varpool is **read-only**. If you need to add variables, make a copy of the `sys` varpool (as described above) and add your variables to that copy.

4.12.3. Listing All Variables

To list all variables (and their values) in a varpool, you can use the following loop:

```
for(var v in sys)
    {
    log(v + "=" + sys[v]);
    }
```

You use the name of your varpool instead of `sys` where appropriate.

4.13. File Handling

JavaScript, as a scripting language, does not provide functions for file handling. cpmill provides the following special objects as abstract “File” objects, separated by input and output files:

- `FileInputDocument`
- `FileOutputDocument`
- `FileInputText`
- `FileOutputText`
- `FileInputBinary`
- `FileOutputBinary`

The two `...Document` types provide access to all file types that the Compart MFF filters can read and write, while the `...Text` and `...Binary` types provide basic access to text and binary data files, respectively.

4.13.1. General Remark

Creating a new file object will always succeed, even when the file in question does not exist (for input files) or the target directory is write-protected (for output files). In fact, the constructor will not try to access the file at all. Only when you try to actually do something with the file it will try to read or write the file.

Therefore, when you need to catch exceptions (e.g. when it is not guaranteed that an input file already exists), you will have to use a `try ... catch` block around the first file operation, not around the constructor.

```
file = new FileInputText("myfile.txt"); // will always succeed

try
    {
    while(!file.eof()) // may throw an exception
        {
        line = file.readln();
        }
    }
catch (e)
    {
    log("Read error - check that file exists");
    }
```

4.13.2. Output Queue

4.13.2.1. Writing Pages

In cpmill, the standard way to write out pages is to push them into the output queue.

```
output.push(page);
```

The `output` object is available in every event handler.

4.13.2.2. Closing the Queue

At any time, you can close the output queue, thus indicating to cpmill that you want to abort processing.

```
output.quit();
```

Note: This does **not** abort processing of the current event handler. Only when the event handler ends will cpmill actually abort the processing.

To abort immediately, follow the call to `output.quit()` with a (JavaScript) `return`.

```
output.quit();
return;
```

You can also use the method `quit()` to specify a return value that cpmill will then return to its caller.

```
output.quit(42);
```

It will make cpmill (the application) return an exit code of 42 to its caller. You can use this in a shell script, for example.

4.13.2.3. Setting a Table Of Contents

When creating PDF output, you may want to create a Table Of Contents. To associate the object `TOC` with the current output file, use the method `setTOC ()`, see *chapter 4.9. Bookmark and TOC on page 98* for details.

```
output.setTOC(toc);
```

4.13.2.4. Changing the Output File Name

In normal operation, the name of the output file will depend on the name of the input file(s) and the cpmill driver type you are using. To explicitly set a name for the output file you use the method `setFilename()`.

```
output.setFilename("myfile.pdf");
```

This call is only valid in the event handler `openoutputdocument` and will raise an exception when used in any other event handlers.

4.13.3. FileInputDocument

The object `InputDocumentAttributes`, passed as the second parameter, provides a way to set various attributes for the file and to control the file's behavior. This parameter is only optional.

```
attribs = new InputDocumentAttributes();
attribs.setType('afp');

infile = new FileInputDocument("test.afp", attribs);
page = infile.readPage(1);
infile.close();
```

4.13.3.1. Bytearrays as FileInputdocument

`FileInputDocument` and `FileOutputDocument` can process bytearrays as input or output, see the example below where bytearrays are used both for input and output.

```
// The FileInputBinary is used to read a file into a JS bytearray
fileInBin = new FileInputBinary("electricblankets.afp");
data = fileInBin.readAll();
fileInBin.close();

// We have a bytearray ('data') which should be used as input file.
// You need to supply a virtual file name that starts with 'mem://'.
attr = new InputDocumentAttributes();
fileInMemory = new FileInputDocument("mem://testdoc", attr, data);

// Specifying 'mem://' and a dummy file name (used for logging) tells
// the FileOutputDocument, that a bytearray has to be created.
fileOutMem = new FileOutputDocument("mem://testdoc.pdf");

// Reading and writing is not affected.
iPage = 1;
while (fileInMemory.hasPage(iPage))
    {
    page = fileInMemory.readPage(iPage);
    fileOutMem.write(page);
    iPage++;
    }

// The FileInputDocument (in memory) should be closed (like any file).
fileInMemory.close();

// Now we have to retrieve the data (as bytearray) from the FileOutputDocument.
// You need to close the file first and then call getDataBinary
fileOutMem.close();
outData = fileOutMem.getDataBinary();

// Finally, we use the FileOutputBinary to write the bytearray into a file
fileOutBin = new FileOutputBinary("electricblankets.pdf");
fileOutBin.write(outData);
fileOutBin.close();
```

Compart **does not recommend** to convert a file on disk into a JavaScript bytearray and vice versa. It is memory and time consuming. The functionality is meant for bytearrays that are created or used within JavaScript.

4.13.3.2. Reading Pages and Page Numbers

Pages can be read using `readPage` and specifying the page number. **Note:** Page numbers start at **1**. A page number of 0 was silently accepted in earlier releases (and treated as page 1), but it will throw an error.

If you need to know, how many pages are in the document, you do the following:

```
numpages = infile.getNumberOfPages();
```

4.13.3.3. Closing Files

Closing a file is optional. Files are implicitly closed when the file object goes out of scope.

4.13.3.4. Special Settings and Methods

PDF File Creation Date

From the `FileInputDocument` object, you can use the `getPdfFileInputDocumentData` object to retrieve creation of the PDF file.

```
myPDFData = fileIn.getPdfFileInputDocumentData();
```

This object offers a method to access the creation date of a PDF document as string.

```
creationDate = myPDFData .getCreationDate();
```

PDF File Attachments

Attachments of PDF type `FileInputDocument` objects can be accessed. For example, attachments can be added to a PDF type `FileOutputDocument` object.

In the example below the attachment is retrieved from a `FileInputDocument` object which is queried from a `Page` object and writes the attachment to a `FileOutputDocument` object together with an empty page.

```
// only executed for the first input page here
if (sys.PAGE == 1)
{
    // query FileInputDocument object via current Page object
    fileIn = page.getFileInputDocument();

    // query number of attachments
    iAttachments = fileIn.getNumberOfAttachments();
    log(" ==> file has " + iAttachments + " attachments");

    // create a new FileOutputDocument
    name = sys.ROOT_DIR + "/output/Attachments_mill.pdf";
    fileOut = new FileOutputDocument(name);

    // loop over attachments
    for (iLoop = 0; iLoop < iAttachments; iLoop++)
        {
        // query attachment name
        attachmentName = fileIn.getAttachmentName(iLoop);
        log("     Attachment " + iLoop + " is '"+attachmentName+"'");

        // retrieve attachment as bytearray
        attachmentData = fileIn.getAttachmentData(iLoop);
        log("     Attachment " + iLoop + " has '"+attachmentData.length+"'");

        // add the attachment (bytearray) to the FileOutputDocument
        fileOut.addAttachmentBytearray("Attachement_"+iLoop+"_"+attachmentName, attachmentData);
        }
```

```
    // add an attachment via filename (instead of bytearray)
    fileOut.addAttachment(sys.ROOT_DIR + "/input/attachment/UtilsAttachments.pdf");

    // write an empty page because PDFs with 0 pages are not allowed
    newpage = new Page(Constants.Page.Paperformat.A5);
    fileOut.write(newpage);
    fileOut.close();
}
```

Notes:

- It is not allowed in PDF to have two attachments with the same name.
- Attachments can only be written to `FileOutputDocument` objects (and not to the generic output queue).

4.13.3.5. XMP Data

Extensible Metadata Platform (XMP) provides a way to add metadata to PDF documents. To read the metadata from a `FileInputDocument`, use these methods:

```
attribs = new InputDocumentAttributes();
attribs.setType('pdf');
infile = new FileInputDocument ("input.pdf", attribs);
author = infile.getAuthor();
title = infile.getTitle();
subject = infile.getSubject();
keywords = infile.getKeywords();
```

4.13.4. FileOutputDocument

`FileOutputDocument` is the equivalent to `FileInputDocument` and can be used to open additional output files. The object `OutputDocumentAttributes` provides a way to set various attributes for the file and control the file's behavior. The second parameter is optional. Leaving it out only makes sense, though, when the file type can be determined from the file extension. .

```
attribs = new OutputDocumentAttributes();
attribs.setType('afp');

outfile = new FileOutputDocument("test.afp", attribs);
outfile.write(page);
outfile.close();
```

4.13.4.1. Bytearrays as FileOutputDocument

`FileOutputDocument` can return a bytearray instead of writing a file to disk, see a detailed example in *chapter 4.13.3.1. Bytearrays as FileInputdocument on page 114*. .

4.13.4.2. Writing Pages

You can either use the method `write` to write pages directly, or you can push both the page and the output file into the output queue.

```
output.push(page, outfile);
```

Note: You cannot close the file yourself when you add the file to the output queue like that. Attempts to do so will throw an exception.

4.13.4.3. XMP Data

Extensible Metadata Platform (XMP) provides a way to add metadata to PDF documents. Currently, you can define a document's author, title, subject, and keywords.

When creating your own PDF output files using `FileOutputDocument`, you can define the metadata. You do the following:

```
attribs = new OutputDocumentAttributes();
attribs.setType ('pdf');
outfile = newFileOutputDocument ("output.pdf", attribts);
outfile.setAuthor(author);
outfile.setTitle(title);
outfile.setSubject(subject);
outfile.setKeywords(keywords);
outfile.write(page);
outfile.close();
```

4.13.4.4. Special Settings and Methods

Separate Overlays

To separate objects (currently limited to images) that occur on more than one page in the document, you can activate the option `SeparateOverlays`.

```
outfile.setSeparateOverlays(true);
```

Note: The option cannot be changed after the output file has been opened.

The equivalent parameter in cpmill is

```
<parameter name="out.file.separate.overlays" value="true"/>
```

Scale To Gray

You can use the method `setScaleToGray` to convert an image with 16 levels of gray.

```
outfile.setScaleToGray(true);
```

As with binary copy mode, this option cannot be changed after the output file has been opened.

The method `getScaleToGray` is available and returns the status of the method `setScaleToGray`.

PPML

To use a Personalized Print Markup Language (PPML) file for imposition it requires a call to the `setPpmlFile` function, as the PPML file specified in the `<ppml>` section of a cpmill profile is **not** used for files created using `FileOutputDocument`. For the example below, you can code the following:

```
outfile.setPpmlFile("impo_a3.ppml");
```

For more information on imposition and PPML, see also *Imposition Support User's Guide*.

The optional parameter `FlushPageCount` can be specified in combination with the method `setPpmlFile`.

```
outfile.setPpmlFile("impo_a3.ppml", 4);
```

The write process of the PPML file is terminated after the specified page and the write process starts again.

AFP

The following methods can be used with AFP documents:

- `setOutputFontId`
- `setAfpSelfContained`
- `setAfpCodePageTle`
- `setAfpCodePagePgm`

```
outfile.setOutputFontId("AA");
outfile.setAfpSelfContained(true);
outfile.setAfpCodePageTle("IBM424");
outfile.setAfpCodePagePgm("UTF-8");
```

These methods are the equivalents of the following cpmill profile parameters:

- `out.put.font.id`
- `out.file.resources.selfcontained`
- `out.afp.tle.ianacodepagename`
- `out.afp.pgm.ianacodepagename`

Likewise, the following methods are the equivalents for special IPDS settings:

IPDS

```
outfile.setIpdsPrinterName("InfoPrint1585");
```

This is the equivalent of the `out.ipds.printer.name` parameter and lets you set the name of the printer for IPDS output. It must match the printer name in the file mffipd.pro.

PDF File Attachments

Attachments can be added to `FileOutputDocument` objects, see also *chapter 4.13.3.4. Special Settings and Methods on page 115*.

4.13.5. Document Attributes

To give the user more control over the way documents are being opened and created while avoiding to introduce more and more optional parameters, two new objects were introduced:

- `InputDocumentAttributes`
- `OutputDocumentAttributes`

As the names imply, these are meant to be used with the `FileInputDocument` and `FileOutputDocument` objects, for which new constructors have been introduced.

```
attrIn = new InputDocumentAttributes();
infile = new FileInputDocument("filename.afp", attrIn);
```

```
attrOut = new OutputDocumentAttributes();
outfile = new FileOutputDocument("filename.afp", attrOut);
```

Note: The old constructors are still working and will not be deprecated for the forseeable future. However, new features and attributes will only be added via `InputDocumentAttributes` and `OutputDocumentAttributes`.

4.13.5.1. InputDocumentAttributes

The object `InputDocumentAttributes` describes the attributes of a `FileInputDocument` object. Currently implemented are:

```
attrIn.setType("iff");
attrIn.setSubType("tif");
attrIn.setPassword("swordfish");
```

```
file type = attrIn.getType();
subType = attrIn.getSubType();
```

setJobName

Xerox' LCDS (Line Conditional Data Stream) files can contain multiple so-called jobs. The jobs will usually be specified in the profile `mfflcd.pro`. In a case where that is not flexible enough, cpmill will let you select the job as follows:

```
attrIn.setJobName("JOB1");
```

The method `getJobName` is available that returns the last job set via the attribute `setJobName`.

Note: Job names are currently only supported for LCDS files and may result in undefined behavior when set for other input formats.

setJsl

In LCDS, the JSL can contain multiple jobs, see `setJobName`. In turn, you can also have multiple JSL files. The method `setJsl` will let you select which JSL file to use:

```
attrIn.setJsl("/path/to/MY.JSL");
```

There is also a method `getJsl` that returns the last JSL set via `setJsl`.

Note: JSL files are only used for LCDS and may result in undefined behavior when set for other input formats.

setFormdefProcessing

The method `setFormdefprocessing` enables or disables the formdef processor (FDP). This is the equivalent of the cpmcopy option `-enablefdp`. Formdef processing is currently disabled (`false`) by default. You call the function with `true` to enable it. **Note:** In cpmill, you need to call `setFormdefprocessing` in the `openinputdocument` handler, as it needs to be set **before** the input document is opened.

```
attrIn.setFormdefProcessing(true);
```

The method `getFormdefProcessing()`is available that returns the status of `setFormdefprocessing`.

setFormdefName

The method `setFormdefName` specifies a form definition (formdef) name for the input file.

```
attrIn.setFormdefName("F1EPS001");
```

The method `getFormdefName()` is available and returns the form definition name.

setPagedefName (AFP Line Mode Input)

The method `setPagedefName` specifies a page definition (Pagedef) name for the AFP line mode input file.

```
attrIn.setPagedefName("FP1TEST");
```

The method `getPagedefName()` is available and returns the page definition name.

getCopies

This method `getCopies` returns the number of copies of a page, as defined for the input file. This is mainly of interest if you want to copy that value over to the output file (or change it there).

```
attrIn.setFormdefName("F1EPS001");
```

You cannot change this value for the input file (obviously), so there is no method `setCopies` for the object `InputDocumentAttributes`.

setSequentialRead

This method `setSequentialRead` sets the object `FileInputDocument` to sequential (or non-sequential) read. The default is `false`, i.e. the MFF filter's default behavior is left unchanged.

```
attrIn.setSequentialRead(true/false);
```

The method `getSequentialRead()` is available and returns the sequential read mode.

addResourceLibrary

This method `addResourceLibrary` adds a resource library by file name that will be used by the input file opened with this attribute.

```
attrIn.addResourceLibrary("ResourceLibraryFileName1");
attrIn.addResourceLibrary("ResourceLibraryFileName2");
attrIn.addResourceLibraryMemory(bytearrayOfResourceLibrary1);
//...
```

addResourceLibraryMemory

This method `addResourceLibraryMemory` adds a resource library as bytearray that will be used by the input file opened with this attribute.

```
attrIn.addResourceLibraryMemory(bytearrayOfResourceLibrary1);
attrIn.addResourceLibraryMemory(bytearrayOfResourceLibrary2);
attrIn.addResourceLibrary("ResourceLibraryFileName2");
//...
```

setTextCodepage

This method `setTextCodepage` sets a code page for text (MFFTXT filter) documents. By default, the Compart internal code page name is used, see *Appendix B: Code Page Names on page 277*.

The method `getTextCodepage` is available and returns the current code page name.

setMemoryMap

When the method `setMemoryMap` is called, the input file is used as memory mapped file. The file is read into the computers memory.

```
attrIn.setMemoryMap(true);
```

This action speeds up all following calls to the file, like read, seek, and tell. But it requires that enough memory is available. Memory mapping can be globally switched on for all input files with the environment variable `set TOOLKIT_MMAP_INFILES=1`.

4.13.5.2. OutputDocumentAttributes

The object `OutputDocumentAttributes` describes the attributes of a `FileOutputDocument` object. Currently implemented are:

```
attrOut.setType("iff");
attrOut.setSubType("tif");
attrOut.setOwnerPassword("swordfish");
attrOut.setUserPassword("password");
attrOut.setKeyLength(128); // can be 40 or 128
```

```
file type = attrOut.getType();
subType = attrOut.getSubType();
```

setBinaryCopyMode

When using the object `OutputDocumentAttributes` and you want to change the binary copy mode, then you should set the method `setBinaryCopyMode` for the object `OutputDocumentAttributes` instead of setting it for `FileOutputDocument`. Possible values are:

- `Auto`: binary copy mode is used, but it will (silently) be switched to non-binary mode, if binary copy processing is not possible.
- `Always`: binary copy mode is used. The processing is aborted, if it is not possible. **Note:** Currently not implemented.
- `Never`: Do not use binary copy at all.

```
attrOut.setBinaryCopyMode(Constants.OutputDocumentAttributes.BinaryCopyMode.Never);
```

If you have specified `Auto` for your process run, you will find information in the log file about reasons why the binary copy process might have failed and completed correctly.

To check the current binary copy mode, you use the method `getBinaryCopyMode`.

Binary Copy Mode (Alternative Version)

It is recommended to set binary copy mode via the object `OutputDocumentAttributes`, as described above. For backward compatibility, there is also an alternative method, which calls `setBinaryCopyMode` on the object `FileOutputDocument`:

```
file.setBinaryCopyMode(Constants.FileOutputDocument.BinaryCopyMode.Never);
```

Note: Using both methods on the same file does not make a lot of sense and will cause, in fact, a warning message to be logged. The binary copy mode cannot be changed any more once you have started writing anything to the file.

setSelfContained

The method `setSelfContained` is the equivalent of the method `setAfpSelfContained` (note the name change!) to create self-contained AFP files that contain all required resources.

```
attrOut.setSelfContained(true);
```

To check the current self-contained mode, you use the method `getSelfContained` (default is `false`).

setEmbedFonts

The method `setEmbedFonts` signals that fonts should be embedded into the created files. Currently, it works for PDF output files. **Important:** It requires binary copy mode to be set to `never`!

```
attrOut.setEmbedFonts(true);
```

To check the current font mode, you use the method `getEmbedFonts` (default is `false`).

Simplex/Duplex Handling

setCompleteDuplex

The method `setCompleteDuplex` is the equivalent of the cpmcopy parameter `-completeduplex`. It ensures that in duplex mode, for every front page there is also a back page by inserting empty pages when necessary.

```
attrOut.setCompleteDuplex(true);
```

To check the current print mode, you use the method `getCompleteDuplex` (default is `false`). **Important:** It requires binary copy mode to be set to `never`!

setGeneralDuplex

The method `setGeneralDuplex` is the equivalent the cpmill profile parameter `general.duplex`. It switches the output document to duplex mode so that you do not have to explicitly set the front and back for each page.

```
attrOut.setGeneralDuplex(true);
```

To check the current print mode, you use the method `getGeneralDuplex` (default is `false`).

Note: A page's simplex/duplex setting takes precedence, so using the method `setSimplexDuplex` may interfere with this method.

setFormdefName

The method `setFormdefname` lets you specify form definition names on a per-file basis. It also overwrites the cpmill profile parameter `out.afp.formdef.name`.

```
attrOut.setFormdefName("F1EPS001");
```

To get the form definition name returned, you use the method `getFormdefName`.

setLinearized

The method `setLinearized` lets you generate a 'linearized', i.e. web-enabled PDF file. When a PDF viewer browser plug-in is used, a single page of a PDF file can be requested from a web server and displayed, only supported by MFF filter for PDF. **Note:** Enabling linearized writing will disable binary copy.

```
attrOut.setLinearized(true);
```

To get the status of the linearized operation returned, you use the method `getLinearized`.

setCopies

The method `setCopies` is the equivalent of the cpmcopy parameter `-copies`. It specifies the number of copies for every page in the document. **Note:** Only some output formats have proper support for that option (e.g. PCL) in that they only store a single copy but print the requested number of copies. Other formats, e.g. PDF, will actually store multiple copies of every page.

```
attrOut.setCopies(5);
```

You can use the complementary method `getCopies`, too.

setDocumentCopies

The method `setDocumentCopies` is the equivalent of the cpmcopy parameter `-documentcopies`. It specifies the number of copies of the document. **Note:** Only the PCL output format has proper support for that option. In the PCL datastream a single copy is stored, but the print job contains the requested number of copies.

```
attrOut.setCopies(5);
```

You can use the complementary method `getDocumentCopies`, too.

setSeparateOverlays

The method provides the same functionality as `setSeparateOverlays` for `FileOutputDocument` objects. However, it is recommended to use this incarnation if you are using `OutputDocumentAttributes` anyway.

Calling `setSeparateOverlays(true)` will instruct the output filter to try and separate images into overlays, if possible. For this to work, you may also have to set other attributes, e.g. `setSelfContained` when working with AFP files.

```
attrOut.setSeparateOverlays(true);
```

You can use the complementary method `getSeparateOverlays`, too.

setFontIdAutoAppendString

This method provides the same functionality as `setOutputFontId` for `FileOutputDocument` objects. Compart, however, recommends using this incarnation if you are using `OutputDocumentAttributes` anyway.

```
attrOut.setFontIdAutoAppendString("XY");
```

Note: The name has been changed to better reflect its functionality. It does not actually set a font ID, but only provides a suffix that is used by the MFF filter MFFAFP when it assigns font IDs.

You can use the complementary method `getFontIdAutoAppendString`.

setOverlayIdAutoAppendString

The use of the method `setOverIdAutoAppendString` only makes sense when you also use it in combination with `setSeparateOverlays(true)`. It provides a suffix that is appended to IDs for those overlays that are created when separating images into overlays.

```
attrOut.setOverlayIdAutoAppendString("XY");
```

You can use the complementary method `getOverlayIdAutoAppendString`.

applyIccProfile

To apply the ICC profile from the given file to all the pages in the output file you can code the following:

```
attrOut.applyIccProfile("AdobeRGB1998.icc")
```

You can use the complementary method `getAppliedIccProfile`.

setMergeFonts

The method `setMergeFontsDocumentCopies` is the equivalent of the cpmcopy parameter `-mergefonts` and reduces the number fonts used in a page merging overlapping font subsets into a font.

```
attrOut.setMergeFonts(true);
```

Compart recommeds to use this method instead of the Page method with the same name. The merging the fonts over the entire document achieves better results than doing it separately for every single page.

You can use the complementary method `getMergeFonts`.

setPpmlFile

The method `setPpmlFile` specifies a PPML file to be used for the output. When the PPML file does not allow determining after how many pages the output can be flushed out, the optional integer paramter `iFlushPageCount` should be set to avoid having too many pages in memory.

```
attrOut.setPpmlFile("<filename>");
 // or
attrOut.setPpmlFile("<filename>", iFlushPageCount);
```

4.13.5.3. PDF Security Settings

For security reasons PDF can be specified with a set of documents restrictions, e.g. not allowing contents to be printed.

The section describes the cpmill / Docponent equivalents of the cpmcopy options of the same name, e.g. `setCopyNotAllowed` does the same as the cpmcopy option `-pdf.copy.notallowed`, see *chapter 3.2. cpmcopy Command Line Parameters on page 26*.

OutputDocumentAttributes

For `OutputDocumentAttributes`, the following methods can be used to control PDF security settings:

```
attrOut.setNoChange(true);
attrOut.setTextNotesNoChange(true);
attrOut.setCopyNotAllowed(true);
attrOut.setPrintNotAllowed(true);
attrOut.setFillNotAllowed(true);
attrOut.setAccessibilityNotAllowed(true);
attrOut.setAssembleNotAllowed(true);
attrOut.setDigitalCopyNotAllowed(true);
```

The default value for all of these settings is `false`, i.e. as long as they are not called; the specific option or feature **is** allowed.

There are also complementary `get` methods, e.g. `getNoChange()`etc., for all of these `set` methods.

InputDocumentAttributes

For `InputDocumentAttributes`, only the `get` methods exist and can be used to test the status of a page or document:

```
attributes = page.getFileInputDocument().getInputDocumentAttributes();

if(attributes.getPrintNotAllowed()))
    {
    log("Printing of this page is not allowed");
    }
```

Option copysecurity

Note: cpmill does **not** have a `-copysecurity` option as in cpmcopy. Instead, cpmill will check that the security settings for the input and output file are compatible and will log a warning if they are not.

So if, for example, the input file does not allow printing, then the setting `setPrintNotAllowed(true)` should be specified for the output file. Otherwise, cpmill will abort the conversion process.

Merging Security Settings

The function `mergeSecuritySettings` is available for convenience reasons so that users can easily create a set of security settings that is compatible with all input documents.

```
inAtts1 = page1.getFileInputDocument().getInputDocumentAttributes();
inAtts2 = page2.getFileInputDocument().getInputDocumentAttributes();

outAtts = new OutputDocumentAttributes();
outAtts.mergeSecuritySettings(inAtts1);
outAtts.mergeSecuritySettings(inAtts2);
```

In the above example, whatever the restrictions (security settings) for the two pages (`page1` and `page2`) are would be inherited by the attributes for the output document.

You can, of course, always set more restrictions afterwards. **Note:** However, you **cannot** have fewer restrictions for the output file than for the input files.

4.13.5.4. Integration with cpmill

The document attributes are also meant to replace the cpmill profile `<parameter>` attributes (in the long term). For this, cpmill has two new events, `openinputdocument` and `openoutputdocument`.

Event openinputdocument

The event `openinputdocument` is being called for every document that cpmill opens for input. The event handler has access to that file's name (`filename` variable) and its document attributes (`attributes` variable).

```
<eventhandler type="openinputdocument">
    <script type="text/javascript">

        log("Filename: " + filename);
        log("Type    : " + attributes.getType());
        log("Subtype : " + attributes.getSubType());

        attributes.setPassword("swordfish");

    </script>
</eventhandler>
```

Currently, you can only set the password to be used for reading encrypted documents. Future versions will support more options here, including all the current `<parameter>` attributes.

Event openoutputdocument

The event `openoutputdocument` works in pretty much the same way, only that it is being called for the output documents that cpmill creates.

```
<eventhandler type="openoutputdocument">
    <script type="text/javascript">
        log("Filename: " + filename);
        log("Type    : " + attributes.getType());
        log("Subtype : " + attributes.getSubType());
        attributes.setOwnerPassword("swordfish");
        attributes.setUserPassword("password");
    </script>
</eventhandler>
```

Event inputdocumentopened

The event handler `inputdocumentopen` is called **after** the input file has been opened but before the first page has been read from it. At this point, you can query the file's PDF security settings, for example, to properly set them for the output file, see also *chapter 4.13.5.3. PDF Security Settings on page 125*.

```
<eventhandler type="inputdocumentopened">
    <script type="text/javascript">

        // remember input attributes for later use
        inputAttr = attributes;

    </script>
</eventhandler>

<eventhandler type="outputdocumentopened">
    <script type="text/javascript">

        // set other output attributes, e.g.
        attributes.setPrintNotAllowed(true);

        // merge with input attributes
        attributes.mergeSecuritySettings(inputAttr);

    </script>
</eventhandler>
```

Event outputdocumentopened

The event handler `outputdocumentopen` is called **after** the output file has been opened but before the first page has been written. This is the very last point to set the file's PDF Security Settings. **Note:** They cannot be changed any more after a page has been written, see also *chapter 4.13.5.3. PDF Security Settings on page 125*.

4.13.6. FileInputText

```
file = new FileInputText("test.txt");
while(!file.eof())
     {
     line = file.readln();
     }
```

`FileInputText` provides access to text files, which can be read line by line, see the example above. You can also select a code page to use for the text file:

```
file = new FileInputText("test.txt", "UTF-8");
```

For code pages, see *Appendix B: Code Page Names on page 277*.

4.13.7. FileOutputText

```
file = new FileOutputText("output.txt");
file.writeln("Hello, world!");
file.write(str.length, str);
```

`FileOutputText` lets you write to files. **Note:** `writeln` will write lines (including line feeds), while `write` only writes as many bytes as specified by the first parameter.

You can also select a code page to use for the text file:

```
file = new FileOutputText("test.txt", "UTF-8");
```

For code pages, see *Appendix B: Code Page Names on page 277*.

Another optional parameter is indicating whether the file should be opened in append mode.

```
file = new FileOutputText("test.txt", "UTF-8", true);
```

Note: If you want to open a file for append but do not want or need to set a code page, you can simply pass an empty string.

```
file = new FileOutputText("test.txt", "", true);
```

To keep line ends free of operating specifics you set a line ending string as in example below.

```
file = new FileOutputText("test.txt", "UTF-8", false, "\r\n\tLineEnd\r\n");
```

It is appended for every line written with `file.writeln()`. When no line ending is specified, the platform default line ending is used.

4.13.8. FileInputBinary

```
file = new FileInputBinary("test.bin");
data = file.read(42); // read 42 bytes
file.close();
```

```
file = new FileInputBinary("test.bin");
alldata = file.readAll();
file.close();
```

`FileInputBinary` lets you read binary files. You can either read a certain amount of bytes (using `read`) or read the entire file at once (using `readAll`). The data will be returned in a bytearray, which is a JavaScript array where every element is one data byte.

Note: Use the array's `length` property to get the amount of data bytes read from the file.

4.13.8.1. File Position

You can use the method `getPosition()`to get the current position (in bytes) within the file and the method `setPosition()`to set an arbitrary position within the file.

```
file = new FileInputBinary("test.bin");
data1 = file.read(42);

pos = file.getPosition(); // will return 42

file.setPosition(2);
data2 = file.read(40); // read the last 40 bytes again
```

The method `rewind()`lets you set the current file position back to the start of the file. It is effectively identical to using `setPosition(0);`.

4.13.9. FileOutputBinary

```
file = new FileOutputBinary("test.bin");
file.write(data);
```

`FileOutputBinary` lets you write binary files. **Note:** `data` is supposed to be a "binary" JavaScript array, see *chapter 4.13.8. FileInputBinary on page 128*.

Another optional parameter is indicating whether files can also be opened for append.

```
file = new FileOutputBinary("test.bin", true);
```

4.13.10. In-Memory Files

The "output" variants of the above (i.e. `FileOutputBinary`, `FileOutputDocument`, and `FileOutputText`) can also write into memory. To indicate the use of an in-memory file, the file name has to start with `mem://`:

```
attribs = new InputDocumentAttributes();
attribs.setType('pdf');

memfile = new FileOutputDocument("mem://acrobat.pdf", attribs);
memfile.write(page);
memfile.close();
data = memfile.getDataBinary();
```

Note: For `FileOutputDocument` objects, the file has to be closed before you can access the binary data it created. This is because some output filters will only finalize their output when the file is closed.

The contents of an in-memory file only exist as long as the respective file object exists. On the other hand, that also means that they may take up memory even when you do not need them any more. Therefore, to indicate that such a file object is not needed any longer, you can call `delete` on it:

```
delete memfile;
```

This will make it clear to the garbage collector that it can remove that object. Keep in mind that the `delete` call itself only marks the object for deletion but does not delete it immediately. cpmill will call the garbage collector at regular intervals (but not while you are inside an event handler) to ensure timely removal of unused objects.

4.14. Data Capturing

4.14.1. Capturing Text

The method `captureText` to capture text on a page specifies a rectangle (`x`, `y`, `w`, `h`). The result of the capture operation is a generic object that can be queried for the number of lines that have been captured and for a JavaScript array containing the actual text.

```
// capture rectangle
x = new Length("20mm");
y = new Length("5in");
w = new Length("60mm");
h = new Length("35mm");

result = page.captureText(x, y, w, h);

// check number of captured barcodes
if(result.getNumberOfLines() &gt; 0)
    {
    // get the actual text (as a JavaScript array)
    text = result.getText();

    for(var i = 0; i &lt; text.length; i++)
        {
        log(text[i]);
        }
    }
```

Note: The position specified by `x` and `y` must be the upper left corner of the respective bounding box.

Compart strongly recommends that you use the method `getNumberOfLines` to check if `captureText` did actually capture any text. Calling the method `getText` on an empty result set may cause undefined behavior and throws an exception.

Each captured text string also has the properties `x`, `y`, and `rotation`, containing the x and y position and the rotation at which the text string was captured.

```
  log("Position: (" + text[i].x + ", " + text[i].y + ")");
```

Note: `x` and `y` are `Length` objects, see *chapter 4.11. Length on page 108*. The position refers to the font's baseline.

Four additional properties are available: `boundingx`, `boundingy`, `boundingw`, and `boundingh`. When they are specified, the parameters set up the text's bounding box.

4.14.1.1. Optional Parameters

MinSpace

The method `captureText` has an optional fifth parameter: A percent value that expresses the amount of white space between two captured characters that should be considered as blank (i.e. a space character). The value is given in percent in relation to the width of a space character in the current font, where 100 (= 100%) means the full width. This parameter is useful in cases where space characters are not properly recognized, i.e. when the capture operation fails to separate words.

Usually, only values less than 100 will make sense. The default value (when the fifth parameter is not used) is 100. A value of 0 is ignored and treated as if a value of 100 was specified.

CaptureMode

`CaptureMode` is another optional parameter. It specifies how to handle characters that intersect with the given capture rectangle.

```
result = page.captureText(x, y, w, h, 100, Constants.Page.CaptureMode.AllIntersect);
```

Valid modes are:

- `AllInside`: all items inside the specified rectangle
- `AllInsideNoBorder`: items will not be selected that share one or more edges with the rectangle
- `AllIntersect` (default): all items are selected that intersect with the rectangle, i.e. even those items that are only partially within the rectangle
- `AllIntersectRaiseError`: an exception will be raised when there are any items that are not fully within the given rectangle. This is useful to ensure that the given coordinates are correct and to not accidentally remove contents from a page.

The modes listed above can be OR'ed together with the following modifiers:

- `NoDuplicateCharacters`: characters that are positioned multiple times at almost the same position are suppressed
- `NoWordSeparator`: spaces between words are suppressed

Note: The default is `AllIntersect`. This is for backward compatibility. For comparison: `select()` and `eraseRect` use `AllInside` as the default.

BaselineThreshold

`BaselineThreshold` is another optional parameter. It allows you to set a tolerance in case the text you want to capture does not line up (vertically) properly, i.e. there are some small differences in the y coordinates of the character's baselines. The default parameter value is 0.

```
result = page.captureText(x, y, w, h, 100, Constants.Page.CaptureMode.AllIntersect,
         new Length("2pt"));
```

4.14.1.2. Overlay Handling

Overlays are no longer resolved when the method `captureText` is called, so that text located on overlays can also be captured.

Overlays must be explicitly read if you want them to be included in the capturing, see also *chapter 4.2.1.19. Overlays on page 82*.

```
newpage = page.readOverlays();
result = newpage.captureText(x, y, w, h);
```

The method `captureText` is called on the new page that is returned by the method `readOverlays`. The original page remains unchanged. Using this procedure users have the choice to include overlays into the capturing (or not) and also which page should actually be written to the output file (with or without the overlays resolved).

4.14.2. Capturing Barcodes

Additional dynamic library files are required for this function, which are not part of the default DocBridge Mill software shipment. To use this function, please ask Compart AG for a special bundle of DocBridge Mill.

To capture barcodes on a page, call the method `captureBarcode`, specifying a rectangle (`x`, `y`, `w`, `h`). The result of the capture operation is a generic object that can be queried for the number of barcodes that have been captured and for a JavaScript array containing the actual barcode information:

```
// capture rectangle
x = new Length("20mm");
y = new Length("5in");
w = new Length("60mm");
h = new Length("35mm");
result = page.captureBarcode(x, y, w, h);
// check number of captured text lines
if(result.getNumberOfBarcodes() &gt; 0)
    {
    // get the actual barcode information (as a JavaScript array)
    barcodes = result.getBarcodes();

    for(var i = 0; i &lt; barcodes.length; i++)
        {
        log(barcodes[i]);
        }
    }
```

Note: Unlike `captureText`, this may return more than one barcode information structure even when there is only one barcode within the capture rectangle. This is because `captureBarcode` scans that region for several barcode types and may recognize it in more than one encoding.

Every barcode text returned by `getBarcodes` also has two properties to help you identify the barcode:

- `epsilon` contains a probability value. The bigger `epsilon` is, the more likely is it that this is the correct interpretation of the barcode. The array returned by the method `getBarcodes` is sorted by this value, so that the first entry contains the most likely barcode interpretation. **Note:** In releases 200712 and earlier, the array was sorted in the wrong order, with the least likely interpretation in the first entry.
- `type` contains a text string describing the barcode type, e.g. EAN 13.

4.14.2.1. Capturing DataMatrix Code

You use the method `captureDataMatrixBarcode` to capture DataMatrix code from a page. The result of the capture method is a barcode object. Example code:

```
// capture rectangle
x = new Length("8cm");
y = new Length("25.5cm");
w = new Length("15mm");
h = new Length("15mm");

// capture a barcode
bc = page.captureDataMatrixBarcode(x, y, w, h);

// query the barcodes data
print("Barcode: " + bc.getDataBinary());

// query position, size and rows and columns of the barcode
print("   X:Y, width, height  " + bc.getPositionX() + "," + bc.getPositionY()
                               + "   "+bc.getWidth() + "," + bc.getHeight());
print("   Columns/Rows: " + bc.getDataMatrixColumns() + "/" + bc.getDataMatrixRows());
```

To convert from bytearray to string you do the following:

```
DMbyteArray = bc.getDataBinary();
var DMstring = "";
for(var element in DMbyteArray)
{
      DMstring += String.fromCharCode(DMbyteArray[element]);
}
```

Notes:

- The readout DataMatrix code is always filled as binary encoded data. Currently the encoding cannot be evaluated unambiguously.
- When the region that is captured contains more than one DataMatrix code, the barcode found first is used. Make sure the region contains only one DataMatrix code.
- When there is no barcode in the region, the program tries to identify other types of DataMatrix codes. This process is rather time-consuming. Finally, it fails anyway.
- The returned values for the position and the size are only precise within the used resolution. Example: Do not expect a DataMatrix code with the specified position `4cm/4cm` to be returned with a position difference less than 1mm.

4.14.2.2. Optimizing Barcode Capturing

As described in *chapter 4.14.2. Capturing Barcodes on page 132*, barcode capturing will try to find any sort of barcodes, which may take some time and give ambiguous results. To improve the time needed and/or the accuracy, use the additional (optional) parameters.

The optional fifth parameter to `captureBarcode` can take the type of barcode we are actually looking for:

```
result = page.captureBarcode(x, y, w, h, "EAN 13");
```

The optional sixth parameter to `captureBarcode` lets you specify how the page is scanned for barcodes:

- `OnlyHorizontal`
- `OnlyVertical`
- `OneDirection`

Note: `OnlyHorizontal` and `OnlyVertical` are to be used exclusively. You can tell the capture function to only scan horizontally or vertically (but not both, obviously).

The third parameter, `OneDirection`, can be combined with one of the other two, though. It tells the capture function that when it found a barcode to only try and read it in one direction.

```
opt = Constants.Page.OptimizeScanning.OnlyHorizontal |
      Constants.Page.OptimizeScanning.OneDirection;
result = page.captureBarcode(x, y, w, h, "EAN 13", opt);
```

Note: Barcode capturing currently requires certain dynamic link libraries are available that are **not** part of a DocBridge Mill default shipment: `libcpbu*.so` and `cpbu*.dll` respectively, depending on the operating system.

4.14.2.3. Barcode Items

If your document contains native barcode items (as opposed to images representing barcodes), you can also use the method `select()`to capture the barcodes. This feature requires the use of a format that supports barcode items (e.g. SAPGOF or AFP with special profile settings).

4.14.3. Selecting Items

The method `select()`allows you to define a rectangular area of the page to inspect and manipulate all the page elements (items) within that rectangle. Currently, a small subset of the planned features has been implemented and is described below.

Similar to `captureText`, see *chapter 4.14.3. Selecting Items on page 134*, the method `select` will return a generic object that can then be queried for more information.

```
result = page.select();
numItems = result.getNumberOfItems();
```

The method `select()`accepts additional parameters to specify a rectangle and a flag to signal how to treat items intersecting that rectangle (similar to `eraseRect`, see *chapter 4.2.1.18. Removing Items on page 81*):

```
result = page.select(x, y, w, h);
result = page.select(x, y, w, h, Constants.Page.SelectMode.AllInside);
```

By default, only items that are placed entirely inside the given rectangle are selected. You can control this behavior with a fifth parameter.

- `AllInside`: default value used when the fifth parameter is omitted.
- `AllInsideNoBorder`: items will not be selected that share one or more edges with the rectangle.
- `AllIntersect`: all items are selected that intersect with the rectangle, i.e. even those items that are only partially within the rectangle.
- `AllIntersectRaiseError`: an exception will be raised when there are any items that are not fully within the given rectangle. This is useful to ensure that the given coordinates are correct and to not accidentally remove contents from a page.

4.14.3.1. getNumberOfItems

Applied to the result of the `select` operation, the method `getNumberOfItems` will return the number of items (images, texts, etc.) on the page. **Note:** Depending on the file format, texts may be separated into words or even single characters and may add to the number of items. You will usually want to use the method `captureText` to get the text from a page.

4.14.3.2. getItem

You can iterate over the items on a page using a simple loop and the method `getItem`. You can do the following:

```
result = page.select();
numItems = result.getNumberOfItems();

for(var i = 0; i &lt; numItems; i++)
    {
    var item = result.getItem(i);
    }
```

The method `getItem` returns an object representing an item on the page.

4.14.4. Common Operations

The following operations can be performed on items returned from `getItem`. **Note:** Some of these operations are only available as of certain releases.

4.14.4.1. Identifying Items

getClassID

You can identify the type of item returned from `getItem`. You use the method `getClassID` on that item.

```
if(item.getClassID() == Constants.Docponent.ClassID.Text) {
    log("This is a text item!");
}
```

getClassName

The method `getClassName` also lets you identify an item's type, but returns the type as a text string. **Note:** Since many items have changed their name in release 200708, this method only exists for backward compatibility. All new code should use the method `getClassID` instead.

```
if(item.getClassName() == "TextItem") {
    log("This is a text item!");
}
```

Item Types and Class IDs

The table below lists the types of items that may be found on a page.

Table 11: Items Types and Class IDs

Item	ClassID	ClassName	Notes
Annotation	Annotation	AnnotationItem	
Arc	Arc	ArcItem	1
Barcode	Barcode	Barcode	2, 3
Comment	Comment	CommentItem	2
ExternalItem	ExternalItem	ExternalItem	
Field	Field	FieldItem	1
Image	Image	ImageItem	1
Line	Line	LineItem	
Link	Link	LinkItem	1
Path	Path	PathItem	1
Rect	Rect	RectItem	
Shading	Shading	ShadingItem	1
Text	Text	TextItem	

Notes:

1 Not fully implemented yet. You may encounter items of this type on a page but you cannot do a lot with them yet.

2 Barcode and Comment items are ignored (and not returned) in releases 200707 and earlier.

3 Native barcode items are only supported by some file formats, e.g. AFP (requires special profile settings) and SAPGOF. Otherwise, barcodes will be returned as images, see *chapter 4.14.2. Capturing Barcodes on page 132*.

4.14.4.2. Removing Items

Items can be removed from the page using the method `removeFromPage`.

```
item.removeFromPage();
```

You will notice that this is an operation on the **item**, not on the page. Effectively, you are telling the item to "remove itself" from the page it has been selected from. Therefore, this operation only works on items returned from `getItem` after a `select`. It does **not** work on items that have been placed on a page using the `put` operation. You will have to do a `select` first.

4.14.4.3. Removing Selections

You can also remove an entire selection of items from a page at once.

```
result = page.select(x, y, w, h);
result.removeFromPage();
```

This does effectively the same as the method `eraseRect` for pages, see *chapter 4.2.1.18. Removing Items on page 81*.

You can, however, remove items from the selection first and only then remove the (remaining) selection from the page, which may come in handy for some applications.

```
result = page.select(x, y, w, h);

numItems = result.getNumberOfItems();
for(var i = numItems - 1; i &gt;= 0; i--)
    {
    var item = result.getItem(i);
    if(item.getClassID() == Constants.Docponent.ClassID.Image)
        {
        result.removeItem(item);
        // or: result.removeItemByIndex(i);
        }
    }

result.removeFromPage();
```

In the above example, we remove all images from the selection and then remove the remaining selection from the page. Therefore, after this process, the given rectangle would only contain images and all other types of items have been removed.

Note: The `for` loop is running backwards since the number of items decreases and would require us to adjust both `numItems` and `i` after every `removeItem` call.

4.14.5. Other Operations

4.14.5.1. Color

The color of an item can be queried and changed using the method `getColor` and `setColor`, see *chapter 4.10.1.9. Common Methods on page 106.*

4.14.5.2. Position

The position of an item can be queried (separately for x-coordinate and y-coordinate) and changed using the methods `getPositionX`, `getPositionY`, and `setPosition`, see *chapter 4.10.1.9. Common Methods on page 106.*

4.14.5.3. Text

The text of a Text item can be queried and changed using the method `getText` and `setText`, respectively. To query the text's font you can code the following:

```
font = item.getFont();
```

Note: You cannot change that font. If you want to render the text with a different font, you should create a new text item with a new font (the current text item and font can both be queried for their parameters which can then be used to create the new text item and font) and remove the original text item from the page.

4.15. Built-in Functions

The following chapter describes additional functions built into the JavaScript interpreter, i.e. they are extensions of the JavaScript language.

4.15.1. print

`print` displays a text on the screen / console only:

```
js> print("Hello, world");
Hello, world
js> print("Current value of variable 'a': " + a);
Current value of variable 'a': 42
```

4.15.2. log

`log` displays the text on both the screen / console and adds it to the application's log file. Other than that, it is identical to `print`.

4.15.3. logMsg

`logMsg` is a sort of "expert" version of `log` as it lets you specify the error number and the severity level:

```
logMsg("JSC4321F Custom fatal error message");
```

Note: The first three letters of the message will always be overwritten with `"JSC"`.

4.15.4. setLogIgnoreLevels

The method `setLogIgnoreLevels` lets you define which log messages with their define severity levels should be ignored **from JavaScript code**. In other words, it lets you hide **JavaScript only** log messages based on their severity level.

```
setLogIgnoreLevels("VID");
log("Log message (treated as an Info level message)");
logMsg("JSC4123I For your information only");
logMsg("JSC4321F Fatal error!");
```

In the above example, all Verbose (V), Info (I), and Debug (D) messages will be suppressed. Hence the example would only print messages of level Fatal (F) and Error (E).

4.15.5. getLogIgnoreLevels

The method `getLogIgnoreLevels` returns the currently suppressed severity levels. It returns an empty string when no messages are suppressed.

4.15.6. system

`system` provides a simple way to execute commands and to start other applications from within JavaScript:

```
system("mkdir newfolder");
system("cpmcopy -i " + infile + " -o " + outfile);
```

Note: The syntax for the command is dependent on the target system. The `system` function only passes the parameter it gets on to the target system for execution.

`system` returns an integer. In accordance with the ANSI-C `system()` function, a return code `-1` means that the command cannot be executed. Any other return code is system and application dependent.

In general, a return code of 0 indicates successful execution. Negative values often denote system-specific errors, while positive values denote application-specific errors.

4.15.7. decodeBase64

As the name implies, `decodeBase64` decodes a Base64-encoded string and returns the data in a bytearray.

```
data = decodeBase64("SGVsbG8sIHdvcmxkIQo=");
```

In the example above, `data` is a bytearray, containing 14 bytes. Creating a string from those bytes prints out `Hello, world!`.

```
decodedText = "";
for(var i = 0; i &lt; data.length; i++)
    {
    decodedText += String.fromCharCode(data[i]);
    }
print(decodedText);
```

Most of the time, though, you would handle Base64-encoded binary data anyway, so the extra step would not be necessary.

4.15.8. fileModifiedDate

The method `fileModifiedDate` returns the last modification date of a file and returns it as a string.

```
creationDate = fileModifiedDate("hello.c", iFormat);
```

Valid values for `iFormat` are:

- `0`: returns date in format: `YYYYMMDD`
- `1`: returns date in format: `DD.MM.YYYY`
- `2`: returns date in format: `DD.MM.YY`
- `3`: returns date in format: `DDMMYYYY`
- `4`: returns date in format: `YYMMDD`
- `5`: returns date in format: `DD-MM-YYYY`
- `6`: returns date in format: `DD-MM-YYYY HH:MM:SS`
- `7`: returns date in format: `HH:MM:SS`

When the file does not exist, an exception is thrown.

4.16. Bytearray

A JavaScript array is a collection of objects. It is not a “flat” memory area, as e.g. a C array.

Since there are some situations where a “flat” array is needed, the concept of a bytearray has been introduced. Effectively, a bytearray is a JavaScript array where each element contains exactly one byte (i.e. an integer value of 0..255).

Usually, a bytearray will be returned from an object’s method, e.g. when reading from a binary file.

In the rare case that you need to create a bytearray manually, you can do so, but it is tedious:

```
data = new Array(1, 2, 3, 4); // bytearray of 4 bytes
```

Alternatively, you can do something like:

```
bin = new Array();

for(var i = 0; i &lt; 16; i++)
    {
    bin[i] = (i * 2) + 1;
    }
```

Note: Using values smaller than 0 or larger than 255 is **not supported** and may **cause unpredictable behavior**.

Since a bytearray is just a JavaScript array, you can use all of the JavaScript array operations on it, e.g. the `length` property to get the number of bytes in the array.

The following objects operate on or return bytearrays:

- Barcode, *chapter 4.7. Barcode on page 93*
- Comments, *chapter 4.5. Comments on page 90*
- FileInputBinary, *chapter 4.13.8. FileInputBinary on page 128*
- FileOutputBinary, *chapter 4.13.9. FileOutputBinary on page 129*
- In-memory files, *chapter 4.13.10. In-Memory Files on page 129*
- decodeBase64, *chapter 4.15.7. decodeBase64 on page 140*

5. Customization of Profiles

The functioning of DocBridge Mill is based on the CpCOLD profile, on MFF filter profiles and on the cpmill profile. The profiles' syntax is described in the following chapters.

5.1. CpCOLD Profile

CpCOLD is extensively configurable via a profile. The format of the profile is SGML. The special syntax is described in file CPCOLD.DTD. This file is included in delivery and can be used by 'high sophisticated users' for determination of ranges of values and can be used (limited) with syntax problems.

The first valid statement in the profile is:

```
<!doctype CPCOLD system>
```

The type CPCOLD references CPCOLD.DTD.

The string 'xxxx xxx xxxx xxxxx xxx xx x x xxxxx' is defined as comment.

```
<!-- xxxx xxx xxxx xxxxx  xxx xx x x  xxxxx  -->
```

The characters <!-- and --> as start and end character strings enclose the comment which can run over several lines.

```
<!-- xxxx xxx xxxx xxxxx  xxx xx x x  xxxxx  -->
```

is the same as

```
<!--
xxxx xxx
xxxx xxxxx
xxx xx x x  xxxxx
 -->
```

You are not allowed to nest comments. Within a list of attributes, no comments are allowed.

5.1.1. CpCOLD Profile – Elements and Attributes

For readability reasons elements and attributes of CpCOLD profile are described in tables. The description order (element/child element/attribute) represents the hierarchical structure of the DTD.

Element <coldunit>		
Element	**Description**	
`<coldunit>`	Any number of units (`<coldunit>...</coldunit>`) can be specified, but nesting is not allowed. Within the element `<coldunit>`, attributes as well as other elements can be specified. The search for available spools is initiated by the values in the element: `<coldunit attribute1 = "value1" attribute2 = "value2"...>`.	
	Child Elements	
	`<capturelist>`, `<classlist>`, `<filter>`, `<fontlist>`, `<formdef>`, `<journal>`, `<outputlist>`, `<picklist>`, `<recordlist>`, `<rules>`, `<searchlist>`, `<triggerlist>`, `<varlist>`	
	Required Attributes	
	defaultclass	Name of the used class, if an empty class is detected because of the search condition for the classes. The class specified here has to be defined in `<classlist>`.
	docfile	Names (rule for generating the names) of the created documents. The first document is named 00000001.tif. Additional documents names are counted up.
	spooldir	Directory where AFP spools are located. Relative to the current directory or absolute, `spooldir` relative to the current directory. Directory must be available!
	Alternative Attributes (Flags)	
	doctype	
	AFPOUT | TIFF	AFPOUT: Document output format AFP. TIFF: Document output format TIFF.
	exptype	
	CEYONIQ | COMPREDIUM | EASY | EASY2 | FILENET | FILENET2 | FILENET3 | IMGMASTER | IXOS | ONDEMAND | SGML | SOLITAS | WEBARCHIVE	See *Table 12: CpCOLD Profile: Attribute exptype - Format Examples on page 164*.
	mffitype	
	MFFIAFP | MFFIGOF | MFFIPDF	Attribute is specified to define the format of the input file. If it cannot be identified by the file extension, see also `spoolmask`.
	mffotype	
	MFFOASC | MFFOAFP | MFFOIFF | MFFOIJP | MFFONUL | MFFOPCL | MFFOPDF | MFFOPOS | MFFOTXT | MFFOXIF |	Attribute is specified to define the format of the output file. If it cannot be identified by the file extension, see also `spoolmask`. If `MFFOIFF` is specified, an additional attribute `mffosubtype` must be specified.
	mffosubtype	
	SUBBMP | SUBJPG | SUBGIF | SUBPNG | SUBTIF	Formats: BMP, GIF, JPG, PNG, TIF
	Preset Attribute	
	spooltype	Value: `MFFIN`. The value need not to be specified.

Element <coldunit>		
	Optional Attributes	
	addresfileexp	AFP only: Additionally to the spool file a resource library can be specified. The expression is calculated by a formula.
	appendexport	If the attribute is specified, a new export file is not generated. Data is appended to the existing file.
	arcdir	Directory where the spool file is stored, if the processing was error free. If this attribute is not specified, the corresponding spool file will only be deleted, `arcdir` relative to the current directory. If `arcdir` is not specified, error free processed files are not saved. The directory is created, if it does not already exist.
	binarycopy	If the input and output format are the same, data is copied binary.
	cpindex	Name of the code page used for the index file. A maximum of 8 characters is allowed.
	cptle	AFP input only: Name of the code page for TLEs. A maximum of 8 characters is allowed.
	converttoimage	If the attribute is specified, the content of the page is rasterized and stored as image.
	createtoc	**Note:** Currently not implemented. PDF output only: Based on archive indexes a table of contents is generated.
	delayedoutput	Not supported.
	delspoolres	If the attribute is specified, CpCOLD deletes the file for external resources (`spoolresfile`) after the processing of the spools.
	docdir	Directory where the separated (and converted) documents are stored. `outdir` is relative to the current directory same as `expdir`. The directory is created, if it does not already exist.
	docdirexp	Expression that describes as result the directory where the separated documents are stored. The expression is recalculated for any new document. Thus depending on the input data it is possible to store each document in another directory. The directory is created, if it does not already exist. **Note:** Either `docdir` or `docdirexp` has to be specified. If both attributes are specified, `docdirexp` overwrites the value of `docdir`.
	docresdir	Directory where CpCOLD manages the external resources of the spool files. CpCOLD uses the following method: If a resource file is found (via `spoolresfile`), CpCOLD searches for the directory `docresdir` and compares binary all files found with the export file. If a file is found that is (binary) identical with the file spoolresfile, CpCOLD uses this file to solve references to resources. If CpCOLD does not find such a file, the import file is copied to directory `docresdir` with a name specified in the profile. If `docresdir` is not specified, the resource management of CpCOLD is suppressed. The directory is created, if it does not already exist.
	docresfile	Rule for file names that CpCOLD uses in `docresdir` to generate new resources.
	easysep	Character that is used as separator for the format EASY. Only valid, if EASY format is used for the export file.
	errdir	Directory where the spool file is stored, if the job is terminated because of an error. Relative to the current directory or absolute, `errdir` relative to the current directory. If `errdir` is not specified, files processed incorrectly are not saved.
	expdir	Directory where the export file is created, as `outdir` relative to the current directory. The directory is created, if it does not already exist.

Element <coldunit>		
	expdirexp	Expression that describes as a result the directory where the corresponding import file is stored. The expression is recalculated for any new document. Thus depending on the input data it is possible to generate a separate import file for any document. The directory is created, if it does not already exist. **Note:** Either `expdir` or `expdirexp` has to be specified. If both attributes are specified, `expdirexp` overwrites the value of `expdir`.
	expfile	Name (expression) of the export file. Example: name `Spool1.afp`, name of the export file `expdir\spool1.exp`.
	extractresexp	AFP only: Inline resources can be extracted into a resource library. The location in file system is calculated by a formula.
	filterdir	Directory, where MFF filter file (*.dll) are stored.
	formdef	AFP input and output only: formdef name. A maximum of eight characters is allowed.
	indexlastpage	Index is generated for a document. Usually index information extraction and calculation is based on the first page. If the attribute is specified, the calculation is performed for each page of the document and overwritten.
	logappend	If the attribute is specified, no new log file is generated. Data is stored into an existing log file.
	logdate	If the attribute is specified, a date stamp is inserted into the log file.
	logdir	Directory where the log file is stored. Relative to the current directory or absolute, `logdir` relative to the current directory. If `logdir` is not specified, no logging is achieved. The directory will be created, if it does not already exist.
	loglevel	Specified level for logging (0: Only errors; 1: Errors and warnings; 2: Operating sequence (0 and 1 together); 3: all). If `loglevel` is not specified, no logging is achieved.
	logtime	If the attribute is specified, a time stamp is inserted into the log file.
	maxfilecount	If the attribute is specified, the number of files is defined that are processed during each pass.
	metadoccount	IXOS archive format only, see Alternative Attributes of `<coldunit>`. If the attribute is specified, the number of files can be defined that can be inserted into a meta document (IXOS).
	mffprodir	Directory, where MFF profile files (*.pro) are stored.
	mffprodirexp	Expression that calculates a directory where CpCOLD searches for MFF profile files. The expression is calculated by a formula.
	newdocdircond	If the attribute is specified, a new output directory is generated. **Note:** A Boolean value must be specified (`TRUE/1` or `FALSE/0`).
	newexpfilecond	If the attribute is specified, a new output export file is generated. **Note:** A Boolean value must be specified (`TRUE/1` or `FALSE/0`).
	noafpds	AFP only: The generation of AFP/DS bytes is suppressed.
	nochanges	Document restrictions, PDF only: No changes in document.
	noextract	Document restrictions, PDF only: No content copying.
	noprint	Document restrictions, PDF only: No printing.
	notextnotechanges	Document restrictions, PDF only: No comment editing.
	otypeexp	Type of output format. The expression is calculated by a formula.
	ovldir	Directory that CpCOLD scans for overlays, which are specified in the child element `<overlay>` of element `<overlaylist>` and that must be copied into the appropriate document page.
	ovldirexp	Expression that defines the directory that CpCOLD scans for overlays, which are specified in the child element `<overlay>` of element `<overlaylist>` and that must be copied into the appropriate document page. If `ovldir` and `ovldirexp` is specified, CpCOLD uses the expression in `ovldirexp`.

Element <coldunit>		
	ovlext	File extension that CpCOLD is looking for when searching overlays for the specified directory. For example, if `ovl` is specified for `ovlext`, an overlay named `o101.oly` cannot be found.
	pagecx	Page width (only for raster image format). If specified, all pages of the spool are scaled to this width, for example 297 mm. If this attribute is omitted, CpCOLD uses the width specified in the spool.
	pagecy	Page length (only for raster image format). If specified, all pages of the spool are scaled to this length, for example 210 mm. If this attribute is omitted, CpCOLD uses the length specified in the spool.
	pageindexing	See also `indexlastpage`. If the attribute is specified, an index set is calculated for each document page and written to the index file.
	pageoffsetx	If the attribute is specified, the output page can be moved horizontally to the position of the input page.
	pageoffsety	If the attribute is specified, the output page can be moved vertically to the position of the input page.
	passwordexp	Document restrictions, PDF only: owner password. The expression is calculated by a formula.
	psgdir	Directory where CpCOLD scans for page segments, which are specified in child element `<pagesegment>` of `<pagesegmentlist>`. They must be copied into the appropriate document page.
	psgdirexp	Expression that calculates a directory, where CpCOLD scans for page segments, which are specified in child element `<pagesegment>` of `<pagesegmentlist>`. They must be copied into the appropriate document page. If `psgdir` and `psgdirexp` are specified, CpCOLD uses the expression in `psgdirexp`.
	psgext	File extension that CpCOLD is looking for when searching for page segments in the specified directory. If `psg` is specified for `psgext`, a page segment e.g. `p101.pse` cannot be found.
	queryfilesize	If the attribute is specified, the system variable for file size is written.
	querypagecount	If the attribute is specified, the system variable for the over-all page number is written the input spool file.
	queryproperties	PDF output only: If the attribute is specified, the document properties are written.
	rasterizefonts	If the attribute is specified, the character information with the font resources is rasterized. Each character is provided as raster image.
	readreverse	If the attribute is specified, the input file is read in reverse direction.
	repcount	Number of attempts of CpCOLD to process a unit. For any attempt the directory `spooldir` is analyzed by means of `spoolmask` and the found spools are processed.
	resolution	Resolution of the generated documents (only for output format TIFF), e.g. 240 dpi. **Note:** If output format TIFF was selected, it makes sense to select the same resolution which is used in the AFP file. It achieves the best output results.
	separatepagetle	AFP output only: If the attribute is specified, TLEs are split in group and page TLEs. The names of the appropriate TLEs get the prefix `PAGE`. Example: In the profile a variable `PAGE_ADDRESS` is generated for the name `ADDRESS`.
	selfcontain	AFP output only: If the attribute is specified, the resources are written into the output file.
	singlepage	If the attribute is specified, CpCOLD generates a separate file for any page of the output document.
	splitdoc	If the attribute is specified, the input file can be split into several output file.

Element <coldunit>		
	spoolmask	Mask for searching the spools in the specified directory, e.g. all `*.afp` are searched in spooldir. **Note:** `spooldir` is concatenated with `spoolmask` and a directory search is done with this pattern. Required backslash characters are inserted.
	spoolmaskexp	Expression describing the mask defined for spool search. It is a logical expression and calculated by a formula.
	spoolresfile	File name of the file that contains the external resources for the current spool (e.g. fonts) and which is provided with the spool, e.g. `spooldir\resource.res`.
	startdate	Date for CpCOLD to start processing of the corresponding unit.
	starttime	Time for CpCOLD to start processing of the corresponding unit.
	storepages	If the attribute is specified, the pages are piled until a new page group is started. See also `<groupstart>`.
	timediff	Time interval between two processing attempts of CpCOLD in respect to a unit.
	Remarks to repcount, startdate, starttime, and timediff: The attributes can be used to realize a polling of CpCOLD for one or more directories. The attributes can be specified for any unit (`<coldunit>`) independently from each other. When CpCOLD start, it checks any unit for the starting time (`startdate` and `starttime`). If these values are not specified, the current time (date; time) is used. At the same time, CpCOLD reads the number of attempts (`repcount`) and the time interval (`timediff`) for any unit from the profile. All units with a valid starting time (`starttime` and `startdate`), will be processed. If the number of repetitions for a unit is reached (`repcount`), the processing is terminated. If this is true for all units, CpCOLD terminates itself. If `repcount` is not specified, CpCOLD assumes the value `1`. If CpCOLD shall poll continuously, value `0` has to be specified for `repcount`. `timediff` specifies the time interval between two processing attempts of CpCOLD. Then CpCOLD checks all units for processing and processes them if applicable. If this cycle is finished, the processing is stopped for the time specified in `timediff`. If for `timediff 0` is specified, another check of the time parameter is done immediately.	
	tlevaltowin	Attribute is equivalent to `cpindex='-1'`. For index code page ISO-8859 is specified.
	tmpdocfile	If the attribute is specified, CpCOLD writes the files with these names and file extension .TMP into a directory. Files are renamed later, i.e. after writing last page, see also `docfile`. Example: It can be used for decoupling a fax server. The server polls files with the extension .TIF. Access conflict may occur, if CpCOLD and fax server tray to use the same file.
	tracedir	Directory where CpCOLD stores the trace files. Relative or absolute to the current directory. The directory will be created, if it does not already exists.
	upasswordexp	Document restrictions, PDF only: user password. The expression is calculated by a formula.
	useannotation	If the attribute is specified, all annotations found are inserted in the CpCOLD list of variables (array), see system variable `$PAGEANNOTATION`.
	usecomment	If the attribute is specified, all comments found are inserted in the CpCOLD list of variables (array), see system variable `$PAGECOMMENT`. AFP only: A NOP field can be a comment.

Element <coldunit>		
	usemultires	If the attribute is specified, the file `spoolresfile` is recalculated for any spools found in `spooldir`. It means that for each spool a separate resource file can be specified. The checking with the resource directory of CpCOLD (`docresdir`) is done independently of that. If `usemultires` is not specified, CpCOLD checks only once – at processing start of the corresponding unit – the resource file and assumes that all additional spools are using the same external resources.
	usetle	If the attribute is specified, all TLEs found are inserted in the CpCOLD list of variables. Within the child element `<index>` of element `<indexlist>` a TLE can be referenced by its name for the attribute `value`.

Element <capturelist> and Child Elements		
Element	**Description**	
`<capturelist>`	A list of variables is specified within the element. The values are read by a positional data capture from a document page.	
	Child Elements	
	`<addresscheck>`, `<capture>`	
	Optional Attribute	
	cond	Rule that controls the execution of the capture list. A Boolean value (`TRUE/1` or `FALSE/0`) must be specified.
Element	**Description**	
`<addresscheck>`	Element checks an address in a defined area localizing address lines. Additional checks (13) can be selected for those areas. `<addresscheck>` functionalities are similar to those of `<capture>`.	
	Child Elements	
	`<addresszone>`, `<allowedfonts>`, `<checks>`, `<clearzone>`, `<content>`, `<result>`	
	Required Attribute	
	name	Variable name
Element	**Description**	
`<addresszone>`	Element specifies the area where the address lines are located.	
	Required Attributes	
	posx	Rectangle size and location is defined by position and dimension values.
	posy	
	width	
	height	
Element	**Description**	
`<allowedfonts>`	Element contains names of fonts used for the address lines.	
	Optional Attributes	
	font1	Up to 20 fonts can be specified, e.g. `font1="Helvetica"`, `font2="Times Roman"`, `font3="Courier"`.
	font2	
	...	
	font19	
	font20	
Element	**Description**	
`<checks>`	Element contains the names of the selected checks.	
	Optional Attributes	
	ri_001_sin001	Clearzone is empty
	ri_002_sin001	Single font used
	ri_002_sem001	Only certified fonts used
	ri_003_sin001	No font decorations used
	ri_004_sin001	Single point size used
	ri_005_sin001	Single font width used

Element <capturelist> and Child Elements		
	ri_006_sin001	Single character spacing
	ri_007_sem001	Word space range
	ri_008_sin001	Used colors
	ri_009_act001	Baseline increments (in tcm)
	ri_010_act001	Row starting positions (in tcm)
	ri_011_sin001	Address complete in addresszone
	ri_011_sin002	Only text items in addresszone
Element	**Description**	
`<clearzone>`	Element defines an empty area around the addresszone. The clearzone rectangle is larger than the addresszone rectangle and encloses it.	
	Required Attributes	
	posx	Rectangle size and location is defined by position and dimension values.
	posy	
	height	
	width	
Element	**Description**	
`<content>`	Definition of variables containing contents for address lines	
	Optional Attributes	
	contentnamear	If the attribute is specified, the address lines are put in an array that has a defined name. e.g. `contentnamear="Adrarray"`. Line 1: `Adrarray [0]`, Line 2: `Adrarray [1]`etc.
	contentnamestr	If the attribute is specified, the address lines are written in a variable (string). The address lines are separated with a defined character, see attribute `contentseparator`.
	contentseparator	The attribute specifies a character that is used to separate the address lines.
Element	**Description**	
`<result>`	The element contains the variables that represent the results of the checks defined in `<checks>`.	
	Optional Attributes	
	resultnamerec	If the attribute is specified, the results of `<checks>` are stored in a structured form (record). The record name (root element) is defined by the variable, e.g. `resultnamerec="MyCheck"`, complete name `MyCheck.RI_001_SIN001`.
	resultnamestr	If the attribute is specified, the results are written in a variable (string). The results are separated with a defined character, see attribute `resultseparator`. Furtheron, a variable (string) is generated, that contains the global result for each check (“Passed” or “Not Passed”) in form of `TRUE` and `FALSE`.
	resultseparator	The attribute specifies a character that is used to separate the result values.
Element	**Description**	
`<capture>`	A variable is specified inside the structure that was generated by a positional data capture form a document page. To capture page elements they must be positioned completely inside the defined rectangle.	
	Required Attributes	
	name	Variable name
	posx	Rectangle size and location is defined by position and dimension values.
	posy	
	height	
	width	
	Alternative Attribute (Flags)	
	capturetype	

Element <capturelist> and Child Elements		
	CAPTUREBARCODE | CAPTUREIMAGE | CAPTURELINE | CAPTURERECT | CAPTURETEXT |	If one of the attributes is specified, element types can be selected for a positional data capture.
	Optional Attributes	
	capseparator	Attribute defines a separator character that separates the captured subelements, see also `separateitems`.
	fudgex	If the attribute is specified, the defined area form where elements can be captured, is widened in x-direction.
	fudgey	If the attribute is specified, the defined area from where elements can be captured is widened in y-direction.
	noinsertblanks	Usually, `<capture>` works as follows: Items positioned separately seem to have one or more blanks between the end of one item and the beginning of the next one, but they do not contain any blanks. In this case blanks are inserted. If the attribute is specified, no blanks are inserted. Example: `'xxx' 'yyy'`: distance between items is wider than the width of one blank. CpCOLD generates `'xxx' 'yyy'`, if attribute is not specified, otherwise `'xxxyyy'`.
	notrim	If the attribute is specified, left-hand and right-hand blanks are not removed.
	remove	If the attribute is specified, the element will be deleted after capturing process has ended.
	separateitems	If the attribute is specified, captured elements are separated, see also `capseparator`.

Element <classlist> and Child Elements		
Element	**Description**	
`<classlist>`	Element allows the definition of classes.	
	Child Element	
	`<class>`	
Element	**Description**	
`<class>`	Element defines a class.	
	Child Elements	
	`<formlist>`, `<indexlist>`, `<itemlist>`, `<overlaylist>`, `<pagedef>`, `<pageinsertlist>`, `<pagesegmentlist>`, `<picklist>`, `<rules>`, `<searchlist>`, `<tlelist>`, `<toclist>`, `<writelist>`	
	Required Attribute	

	name	Name of a class that can be defined inside the element `<rules>` as attribute `value` of the element `<classname>`. For example, if a name `SCH` is determined in `<rules>`, a definition for class `SCH` has to be found in `<classlist>`.
		`<classlist>`
		`<class name=”SCH” rotation=90>`
		`<indexlist>`
		`.`
		`</indexlist>`
		`</class>`
		`</classlist>`
		Note: If in element `<rules>` a empty string for a class name is specified, CpCOLD uses the class name that is specified in attribute `defaultclass` (`<coldunit>`). Then a corresponding definition has to be made as `<class>`.

Element <classlist> and Child Elements		
	Alternative Attributes (Flags)	
	imgconvertmono	
	CLUSTERED | DISPERSED | FLOYDSTEINBERG | NONE | THRESHOLD	Before converting a color raster image into grayscale image a conversion method can be selected. For `Clustered` and `Dispersed`: a matrix size. For `THRESHOLD`: a percentage figure.
	monocompression	
	MMR | FAXG4	If one of the attributes is specified, the compression algorithm for bi-level bitmaps (two colors) is defined.
	Optional Attributes	
	classid	ImageMaster only: If the attribute is specified, a metadoc id can be defined.
	conversionthreshold	If the attribute is specified, a percentage for black/white conversion can be defined, e.g.: `conversionthreshold='50'`.
	doccomment	If the attribute is specified, comments can be written into the output document. Any term can be specified.
	emptypages	If the attribute is specified, empty pages are inserted, before the class is started. Example: `emptypages='2'`.
	formnameexp	AFP input only: The expression, name of Formdef, can be calculated by a formula.
	indexesfromrecord	If an export type is selected, see attribute `exptype` of `<coldunit>`. The attribute is specified to write the contents of a record to the index. Alternatively `<indexlist>` can be used to create an index list.
	indexesfromtle	If an export type is selected, see attribute `exptype` of `<coldunit>`, the attribute is specified to write TLEs to the index. Alternatively `<indexlist>` can be used to create an index list.
	matrixsize	Attribute specifies a number that defines the size of an area that is used to calculate the grayscale value. Example: `matrixsize='3'`.
	pagecx	If the attribute is specified, the width of output page (x-direction) can be specified.
	pagecy	If the attribute is specified, the length of output page (y-direction) can be specified.
	pagegroupnameexp	If the attribute is specified, a name for a page group name can be defined. The expression, page group name, can be calculated by a formula.
	postprocessing	After a class empty pages with TLEs can be inserted.
	preprocessing	Before a class empty pages with TLEs can be inserted.
	printmode	Type of printing: `SIMPLEX` or `DUPLEX`. With `SIMPLEX` the pages of the document are considered page by page for sorting, with `DUPLEX` always pair wise (front and back page).
	recref	If the attribute is specified, a reference for a record definition can be defined, see also `<recordlist>` and child elements.
	resx	If the attribute is specified, the resolution in x-direction for raster images can be defined. Unit is dpi.
	resy	If the attribute is specified, the resolution in y-direction for raster images can be defined. Unit is dpi.
	rotation	Angle defined in degrees. It is used to rotate all pages of the document (of the class). Valid values are `0`, `90`, `180`, `270` degrees and auto. The rotation is counterclockwise (mathematically positive), see also threshold.
	rotationexp	If the attribute is specified, the rotation angle can be defined. The expression, angle value, can be calculated by a formula.

Element <classlist> and Child Elements		
	rotcond	Rule that specifies the rotation angle for a document page. **Note:** The result of the formula is a Boolean value (`TRUE/1` or `FALSE/0`).
	sortexp	Type `EMRSORT` of archive system FileNet only: Expression, whose result of calculation specifies the type of sorting. For type `EMRSORT` the following applies: If `sortexp` has the value `V`, the sorting sequence is not changed. If `sortexp` has the value `R`, the last page (or the two last pages) is positioned at the beginning - depending on attribute `printmode`.
	sorttype	Type `EMRSORT` of archive system FileNet only: Type of sorting of document output pages. Current version of CpCOLD only knows the type `EMRSORT`. If the attribute is specified, the sequence of the pages in the document is changed during writing of the import file.
	tlesfromrecord	AFP output only: If the attribute is specified, TLEs are written from data record into the output file.
	tlesfromtle	AFP output only: If the attribute is specified, TLEs are written as TLEs into the output file.
	trayexp	Expression, whose result of calculation specifies the name of the tray of the printer.
	treshold	If the attribute is specified, a default value is set: `auto='50'`, i.e. on a document page at least 50 characters must be available positioned in one orientation to rotate the page in this direction, see also rotation.
Element	**Description**	
`<formlist>`	Out of function in current version.	
Element	**Description**	
`<form>`	Out of function in current version.	
Element	**Description**	
`<indexlist>`	Element defines a list of indexes.	
	Child Element	
	`<index>`	
Element	**Description**	
`<index>`	Element defines an index.	
	Required Attributes	
	name	Name of the index. The name can be specified without restriction and can be copied into the respective export file. The name can be calculated by a formula.
	value	Value of the index inside the document. The value can be calculated by a formula.
	Optional Attributes	
	array	Out of function in current version.
	col	
	continueonerror	
	global	
	recref	
	remove	
	useinstoredpages	
	varname	

Element <classlist> and Child Elements		
Element	**Description**	
`<itemlist>`	Element defined objects, e.g. barcode, comment, OMR, and text objects.	
	Child Elements	
	`<barcodeitem>`, `<commentitem>`, `<omritem>`, `<textitem>`	
	Required Attributes	
	cond	Rule that controls the execution of the list. A Boolean value (`TRUE/1` or `FALSE/0`) must be specified.
	pagefrom	First page of the document, where the object is included. The specification of the page number is 1-based. **Note:** The attribute is obsolete. Use the attribute `cond`.
	pageto	Last page of the document, where the object is included. The specification of the page number is 1-based. **Note:** The attribute is obsolete. Use the attribute `cond`.
Element	**Description**	
`<barcodeitem>`	Element specifies the barcode symbol. In the Presentation Area (PA) exists the object type Barcode. The object type is interpreted depending on the output filter. If it fails, the barcode is included in the output data stream as image.	
	Required Attributes	
	height	Rectangle size and location is defined by position and dimension values. **Note:** Attribute `ratio` specifies the ratio of bar height to symbol length.
	width	
	posx	
	posy	
	ratio	
	value	Barcode string
	Alternative Attributes (Flags)	
	bcfeatures	
	CHECKDIGITS | CONVERTTOBCD | QUIETZONE |	`CHECKDIGITS`: Check digit included within a barcode for performing a mathematical check of the validity. `CONVERTTOBCD`: Barcode string is converted to BCD. Each decimal digit will be represented by a Binary coded decimal number (BCD) of 4 bits. `QUIETZONE`: Quiet zone is the blank margin on either side of a bar code that's used to tell the barcode reader where a barcode's symbology starts and stops. The purpose of a quiet zone is to prevent the reader from picking up information that does not pertain to the bar code that is being scanned.
	bctype	
	CODE128 | CODE128A | CODE128B | CODE128C | CODE25INDUSTRIAL | CODE25INTERL | CODE25MATRIX | CODE39 | CODE39EXTENDED | CODE93 | CODE93EXTENDED | DATAMATRIX | EAN128 | EAN13 | EAN8 | MSI | MSI10 | MSI11 | PATCHCODEI | PATCHCODEII | PATCHCODEIII | PATCHCODEIV | PATCHCODET | PATCHCODEVI | PDF417 | POSTNET | POSTNET11 | POSTNET5 | POSTNET9 | UPCA | UPCE See also *Appendix D: Barcodes on page 284.*	
	Optional Attributes	
	columns	DataMatrix only: Number of columns of DataMatrix Code.
	itemrot	Rotation of barcode symbol. Values: `0`, `90`, `180`, `270`.
	rgbcolor	If the attribute is specified, the color value is specified in hexadecimal notation, e.g. `rgbcolor="#FFFFFF"`. The color value has seven digits and starts with # and goes on with two digits for red, green, and blue.
	rows	DataMatrix only: Number of rows of DataMatrix Code.

Element <classlist> and Child Elements		
Element	**Description**	
`<commentitem>`	AFP output only: Element defines NOPs.	
	Required Attribute	
	value	Value or contents of `<commentitem>`. The value can be calculated by a formula.
	Alternative Attribute (Flags)	
	commpos	
	EOD | INSIDE | OUTSIDE	Attribute defines the position of the comment. `EOD`: end of document. `INSIDE`: inside of a page. `OUTSIDE`: outside of a page.
Element	**Description**	
`<omritem>`	Element specifies the OMR symbol. In the Presentation Area (PA) exists the object type OMR. The object type is interpreted depending on the output filter. If this is not possible, the OMR symbol is included in the output data stream as image.	
	Required Attributes	
	distance	Distance between two neighboring OMR marks. Measurement unit is twips (1/1440 inch).
	height	Length of OMR mark. Measurement unit is twips (1/1440 inch).
	linewidth	Linewidth of OMR Mark. Measurement unit is twips (1/1440 inch).
	value	The value of a mark is represented as string. Each character that is not a `'nolinechar'`acter generates a mark. Each character that is a `'nolinechar'`acter generates a blank (blank mark).
	Alternative Attribute (Flags)	
	location	
	BOTTOM | LEFT | RIGHT | TOP	Attribute defines the reading direction of the OMR mark depending on the position on the page: `BOTTOM`: Read from right to left – footer margin `LEFT`: Read from bottom to top – left page margin `RIGHT`: Read from top to bottom – right page margin `TOP`: Read from left to right – header margin
	Optional Attributes	
	nolinechar	If the attribute is specified, any kind of character can be used as `'nolinechar'`acter to generate a blank mark, see also attribute value of element `<omritem>`.
	posx	If the attributes are specified, the values define the horizontal and vertical origin of the OMR mark.
	posy	
	rgbcolor	If the attribute is specified, the color value is specified in hexadecimal notation, e.g. `rgbcolor="#FFFFFF"`. The color value has seven digits and starts with # and goes on with two digits for red, green, and blue.

Element <classlist> and Child Elements		
Element	**Description**	
`<textitem>`	Element defines text (constant text or generated by variables) and positions it on a document page.	
	Required Attributes	
	posx	If the attributes are specified, they define the horizontal and vertical origin of the text.
	posy	
	value	Value or contents of `<textitem>`. The value can be calculated by a formula.
	Optional Attributes	
	fontid	If the attribute is specified, the specified value, e.g. local font ID `fontid="1"`, selects a font that is defined within element `<fontlist>` with this font id.
	itemrot	Text rotation. Values: `0`, `90`, `180`, `270`.
	pointsize	If the attribute is specified, the point size of the font is defined, e.g. `pointsize="8"`.
	rgbcolor	If the attribute is specified, the color value is specified in hexadecimal notation, e.g. `rgbcolor="#FFFFFF"`. The color value has seven digits and starts with # and goes on with two digits for red, green, and blue.
Element	**Description**	
`<overlaylist>`	Element defines a list of one or more overlays.	
	Child Element	
	`<overlay>`	
	Optional Attribute	
	cond	Rule that controls the execution of the overlay list. A Boolean value (`TRUE`/`1` or `FALSE`/`0`) must be specified.
Element	**Description**	
`<overlay>`	Element defines an overlay for one or more pages of a document. The overlay has to be found by the attributes (`<coldunit>`) `ovldir` and `ovlext`.	
	Optional Attributes	
	name	Name of the overlay. The name can be specified without restriction and has to match to the file name of the overlay in the directory of the overlays (`ovldir`). The extension is extracted from the attribute `ovlext`.
	nameexp	The expression, name of overlay, can be calculated by a formula.
	posx	Position (x-direction) of the overlay on the page. If the attribute is not specified, `0` is default.
	posy	Position (y-direction) of the overlay on the page. If the attribute is not specified, `0` is default.
	pagefrom	First page of the document, where the overlay has to be included. The specification of the page number is 1-based. If the attribute is not specified, the inclusion of the overlay starts on the first page of the output. **Note:** The attribute is obsolete. Use the attribute `cond` of the element `<overlaylist>`.
	pageto	Last page of the document, where the overlay has to be included. The specification of the page number is 1-based. If the attribute is not specified, the overlay is included on the last page of the output. **Note:** The attribute is obsolete. Use the attribute `cond` of the element `<overlaylist>`.
	rotation	Rotation of the overlay on the page. Values: `0`, `90`, `180`, and `270`. The value is defined in degrees with positive direction of rotation (counterclockwise). If the attribute is not specified, `0` is default.

Element <classlist> and Child Elements		
Element	**Description**	
`<pagedef>`	Out of function in current version.	
Element	**Description**	
`<pagesegmentlist>`	Element defines a list of page segments.	
	Child Element	
	`<pagesegment>`	
	Optional Attribute	
	cond	Rule that controls the execution of the page segment list. A Boolean value (`TRUE`/`1` or `FALSE`/`0`) must be specified.
Element	**Description**	
`<pagesegment>`	Element defines a page segment for one or more pages of a document. The page segment must found by `<coldunit>` attributes `psgdir` or `psgext`.	
	Optional Attributes	
	name	Name of the page segment. The name can be specified without restriction and has to match the file name of the page segments in the directory of page segments (`psgdir`). The extension is extracted from the attribute `psgext`.
	nameexp	The expression, name of page segment, can be calculated be a formula.
	posx	Position (x-direction) of the page segment on the page. If the attribute is not specified, `0` is default.
	posy	Position (y-direction) of the page segment on the page. If the attribute is not specified, `0` is default.
	pagefrom	First page of the document, on which the page segment has to be included. The specification of the page number is 1-based. If the attribute is not specified, the page segment is included on the first page of the output. **Note:** The attribute is obsolete. Use the attribute `cond` of the element `<pagesegmentlist>`.
	pageto	Last page of the document, on which the page segment hat to be included. The specification of the page number is 1-based. If the attribute is not specified, the last page of the output corresponds to the last page of the document. **Note:** The attribute is obsolete. Use the attribute `cond` of the element `<pagesegmentlist>`.
	rotation	Rotation of the page segment on the page. Values: `0`, `90`, `180`, and `270`. The value is specified in degrees with positive direction of rotation (counterclockwise). If the attribute is not specified, `0` is default.
Element	**Description**	
`<printlist>`	Out of function in current version.	
Element	**Description**	
`<printblock>`	Out of function in current version.	
Element	**Description**	
`<printtext>`	Out of function in current version.	
Element	**Description**	
`<pageinsertlist>`	Element contains page definitions.	
	Child Element	
	`<page>`	
	Optional Attribute	
	cond	Rule that controls the execution of the page insertion list. A Boolean value (`TRUE`/`1` or `FALSE`/`0`) must be specified.

Element <classlist> and Child Elements		
Element	**Description**	
<page>	Element defines a page to be inserted.	
	Required Attributes	
	classref	Name of class referring to the insert pages
	pagecx	If the attribute is specified, the width the output page (x-direction) can be specified.
	pagecy	If the attribute is specified, the length of the output page (y-direction) can be specified.
	Alternative Attributes (Flags)	
	insert	
	BEFORE | AFTER	Position of new pages related to the current page.
	Optional Attribute	
	pagecount	Number of inserting pages.
Element	**Description**	
<picklist>	Out of function in current version.	
Element	**Description**	
<pick>	Out of function in current version.	
Element	**Description**	
<rules>	Element controls the separation of the spool into single documents.	
	Child Elements	
	<classname>, <docstart>, <formstart>, <groupstart>	
Element	**Description**	
<classname>	Element controls the determination of the name of a document class. A line break inside the structure is allowed.	
	Required Attribute	
	value	Rule that determines the name of a class. The rule can consist of formulas and can refer to all specified variables. The result of the formula has to be a string that can be used by CpCOLD as class name.
Element	**Description**	
<docstart>	Element defines the beginning of a new document.	
	Required Attribute	
	condition	Rule that controls the beginning of the document. The rule can consist of formulas and can refer to all anyhow specified variables. The result of the formula must be a Boolean value (TRUE/1 or FALSE/0). A new document is generated, if the result of the formula is TRUE.
Element	**Description**	
<formstart>	Element defines inserting of an IMM (Invoke Medium Map) into AFP data stream.	
	Required Attribute	
	condition	Rule controls insert an IMM. The rule can consist of formulas. The result of the formula must be a Boolean value (TRUE/1 or FALSE/0).
Element	**Description**	
<groupstart>	Element enables logical structuring (grouping) that can correspond to AFP data streams.	
	Required Attribute	
	condition	Rule defines the start of a group. The rule can consist of formulas. The result of the formula must be a Boolean value (TRUE/1 or FALSE/0).
Element	**Description**	
<searchlist>	Element defined a list of one or more variables. Its values contain a whole line of the input data stream. The corresponding line can be identified by a specified search pattern.	
	Child Element	
	<string>	

Element <classlist> and Child Elements		
Element	**Description**	
`<string>`	Element specifies a variable. It is assigned to the content of a line as value by a search pattern.	
	Required Attributes	
	name	Name of the variable, in which the content of the line found has to be written. The variable can be referred in other structures by its name.
	value	Search pattern that has to be located inside the line. Jokers (like * or ?) are not allowed. If no line with the respective search pattern is found, an empty string is assigned to the variable.
	Optional Attributes	
	col	If the attribute is specified, the position can be defined where the search action begins.
	changeinput	Attribute defines string using a formula. If the attribute is specified, the found string is replaced by the defined one.
	patternexp	Attribute defines a condition. If the condition is `TRUE`/`1`, the searched item is found. If the attribute `patterexp` is specified, the attribute value must contain an empty string.
	recref	Reference for record definition. If the attribute is specified, a name must be defined in element `<field>` which is a child element of element `<record>`.
	remove	The item found by element `<string>` is removed from the page.
	replaceby	Attribute defines string using a formula. The item found by the element `<string>` is replaced by the string of `replaceby`.
Element	**Description**	
`<tlelist>`	Element defines a list of index elements. The values for the current page or the current group are stored into the data stream (AFP only).	
	Child Element	
	`<tle>`	
Element	**Description**	
`<tle>`	Element defines an index element with name and value.	
	Required Attributes	
	name	Attribute defines the name. The name is used in the data stream.
	value	Rule that specifies the index value. The rule can consist of formulas.
	Optional Attributes	
	cond	Rule defines if a TLE is generated. The rule can consist of formulas. The result of the formula must be a Boolean value (`TRUE`/`1` or `FALSE`/`0`).
	page	If the attribute is specified, TLE is valid for the current page, otherwise for the current group.
Element	**Description**	
`<toclist>`	To be defined.	
Element	**Description**	
`<toclist1>`	To be defined.	
Element	**Description**	
`<toclist2>`	To be defined.	
Element	**Description**	
`<toclist3>`	To be defined.	
Element	**Description**	
`<tocitem>`	To be defined.	

Element <classlist> and Child Elements		
Element	**Description**	
`<writelist>`	Element defines a list of output instructions (`<write>`) for various output files. The files must be defined in child element `<output>` of element `<outputlist>`.	
	Child Element	
	`<write>`	
Element	**Description**	
`<write>`	Element defined output instruction.	
	Required Attributes	
	name	Attribute defines the name.
	value	Value or contents of `<write>`. The value can be calculated by a formula
	Optional Attributes	
	cond	Rule defines if an output generated. The rule can consist of formulas. The result of the formula must be a Boolean value (`TRUE/1` or `FALSE/0`).
	cpwrite	Attribute defines a code page for the output.

Element <filter> and Child Elements		
Element	**Description**	
`<filter>`	Filter suppresses single pages or entire documents. Suppression means that the respective pages or documents are not converted into the output format and are not written into the export file.	
	Child Elements	
	`<docsup>`, `<pagesup>`	
Element	**Description**	
`<docsup>`	Element controls the suppression of an entire document.	
	Required Attribute	
	condition	Rule that controls the suppression of a document. The rule can consist of formulas and can refer to all anyhow specified variables. Keep in mind that the result of the formula has to be a logic value (`TRUE/1` or `FALSE/0`). The document is suppressed, if the result of the formula is `TRUE`. **Note:** In the current version of CpCOLD the docsup condition is not used.
Element	**Description**	
`<pagesup>`	Element controls the suppression of single pages.	
	Required Attribute	
	condition	Rule that controls the suppression of a page. The rule can consist of formulas and can refer to all specified variables. Keep in mind that the result of the formula has to be a logic value (`TRUE/1` or `FALSE/0`). A page has to suppressed, if the result of the formula is `TRUE`.

Element <fontlist> and Child Elements		
Element	**Description**	
`<fontlist>`	Fonts can be listed within the element. Its usage is optional and assigns local font IDs inside an AFP data stream that contains no specification for the corresponding font IDs. (The assignment of the IDs to the respective fonts is done e.g. in the job control). `<fontlist>` is analyzed using the input format AFPIN and the output format TIFF.	
	Child Element	
	`<font>`	
Element	**Description**	
`<font>`	With the element a local font ID is assigned to an AFP font. A line break inside the structure is allowed.	
	Required Attribute	
	fontid	Local font ID that is used inside the AFP data stream for the output text.
	Optional Attributes	
	charset	Name of the character set. A maximum of eight characters is allowed.
	codedfont	Name of the coded font. A maximum of eight characters is allowed.
	codepage	Name of the code page. A maximum of eight characters is allowed.
	family	Name of the font family. A maximum of eight characters is allowed.
	Note: A font in AFP is either specified by characterset/code page or by a coded font. Therefore, it is possible to specify either `charset + codepage` or `codedfont` together with `fontid` for assignment.	

Element <formdef> and Child Elements	
Element	**Description**
`<formdef>`	Out of function in current version.

Element <journal> and Child Elements		
Element	**Description**	
`<journal>`	Element defines a list of journal entries. `<journal>` and `<journalentry>` are alternatives to `<writelist>` and `<write>`, but with two additions, see attribute `tmpjournalfile` and element `<journalentry>`.	
	Child Element	
	`<journalentry>`	
	Optional Attributes	
	journaldirexp	Expression that describes as result the directory where the journal is stored.
	journalfile	Name of journal file. The name can be calculated by a formula.
	journalheaderexp	Expression that contains as a result header line(s) for the journal.
	tmpjournalfile	Name of the temporary journal file. If the attribute is specified, CpCOLD stores the journal in a file. If the processing is complete the temporary journal file is rename to the final journal file. **Note:** The attribute can be used as signal ('conversion completed').
Element	**Description**	
`<journalentry>`	Element defines a journal entry, see also element `<write>`.	
	Required Attributes	
	name	Name of journal entry.
	value	Value of element `<journalentry>`. The value can be calculated by a formula. If `'%d'` is found in the calculated string, CpCOLD replaces it by the number of processed pages at the end of processing.

Element <outputlist> and Child Elements		
Element	**Description**	
`<outputlist>`	Element defines a list of output elements.	
	Child Element	
	`<output>`	
Element	**Description**	
`<output>`	Element defines an outfile file (text format).	
	Required Attributes	
	name	Name of output file. In element `<write>` the name is referenced.
	outfileexp	File specification of output file. The name can be calculated by a formula.
	Alternative Attribute (Flags)	
	openmode	
	APPEND | CREATE	Attribute defines how file is to be opened. `APPEND`: If the file exists, the output is appended. If the file does not exist, it is generated. `CREATE`: The file is generated. If the file already exists, it is overwritten.

Element <recordlist> and Child Elements		
Element	**Description**	
`<recordlist>`	Description	
	Child Element	
	`<record>`	
Element	**Description**	
`<record>`	Description	
	Child Elements	
	`<field>`, `<subrecord>`	
	Required Attribute	
	name	
Element	**Description**	
`<field>`	Description	
	Required Attribute	
	name	
	Optional Attribute	
	array	
Element	**Description**	
`<subrecord>`	Description	
	Required Attributes	
	name	
	ref	
	Optional Attribute	
	array	

Element <triggerlist> and Child Elements		
Element	**Description**	
`<triggerlist>`	Element defines a list with one or more trigger elements.	
	Child Element	
	`<trigger>`	
Element	**Description**	
`<trigger>`	The named element monitors a value that can be calculated by a formula in the attribute `value`. The value of the previous page is noted and is compared with the value of the current page. If the values are not equal, the trigger is activated.	
	Required Attribute	
	name	Name of the trigger. At the same time a variable with that name is defined. Depending on the state of the trigger the value is `0` or `1`.
	value	Value type is `string`. The value can be calculated by a formula. According to the trigger function the value can be monitored, see `<trigger>`.
	Optional Attributes	
	array	Out of function in current version.
	col	
	continueonerror	
	global	
	recref	
	remove	
	useinstoredpages	
	varname	

Element <varlist> and Child Elements		
Element	**Description**	
`<varlist>`	A list of variables is specified within the element. Multiple variable lists can be generated.	
	Child Element	
	`<var>`	
Element	**Description**	
`<var>`	Element is the most important instrument to generate user defined variables. **Note:** A variable defined in a var statement can be referenced in the following var statement.	
	Required Attributes	
	name	Name of the variable. The name is used to reference the variable in the CpCOLD profile. **Note:** Stick to the entry order, i.e. a variable cannot be referenced, it is defined after the current var statement, The entry order is valid for variable lists and within variable lists, but not for other statements.
	value	Value of the variable. The value can be calculated by a formula. The formula contains an error function for value which can be used exclusively there, i.e. in the attribute `value` of `<var>`, i.e. `error(99)`. If the function is executed, CpCOLD terminates the processing with a return code corresponding to the error number. If the attribute `continueonerror` is specified, CpCOLD's behavior is different.

Element <varlist> and Child Elements		
	Optional Attributes	
	array	Out of function in current version.
	col	
	continueonerror	If the attribute is specified, the processing of the current spool is terminated. The processing continues with the next spool. **Note:** No return code is issued.
	global	Usually all variables are initialized for each page. If the attribute is specified, the variables keep its values for all pages or until they are reset.
	recref	Out of function in current version.
	remove	
	useinstoredpages	The attribute has an impact, if the attribute `storepages` in `<coldunit>` is specified. It does the following: A value that is valid for the last page of the unit is set for all pages of the unit. A value can be set for the first page of the unit, which actually is determined on the last page. **Note:** It applies to output related formulas, i.e. `<classlist>` and all child elements.
	varname	Out of function in current version.

Table 12: CpCOLD Profile: Attribute exptype - Format Examples

Export Type	Export Format

CEYONIQ	`name="Hugo Maier"`
	`plz=47111`
	`kundennummer=100.093.13`

COMPRENDIUM	`Hugo Maier^47111^100.093.13^dummy.nul^8`
EASY	`@IMPHDR,^^,name,plz,kundennummer,` `^Hugo Maier^,^47111^,^100.093.13^,^../../data/civa/output\dummy.nul^`
EASY2	`@FOLDER,FI:name,FI:plz,FI:kundennummer,FI:` `^Hugo Maier^,^47111^,^100.093.13^,^../../data/civa/output\dummy.nul^`
FILENET	`dummy.nul;Hugo Maier;47111;100.093.13`

FILENET2	`1`
	`dummy.nul`
	`3`
	`name=Hugo Maier`
	`plz=47111`
	`kundennummer=100.093.13`

FILENET3	

IMGMASTER	`[Index]`
	`MetaDocTypeId=(null)`
	`FieldCount=3`
	
	`[Field0]`
	`FieldId=name`
	`Value=Hugo Maier`
	
	`[Field1]`
	`FieldId=plz`
	`Value=47111`
	
	`[Field2]`
	`FieldId=kundennummer`
	`Value=100.093.13`

IXOS	

ONDEMAND	`COMMENT: ------------------------------------------------------`
	`COMMENT: name:    CPCOLD_UNIT`
	`COMMENT: creator: CPCOLD`
	`COMMENT: created: Fri Oct 29 09:34:42 2004`
	`COMMENT: by:      Compart COLD Execution Unit`
	`COMMENT: ------------------------------------------------------`
	`COMMENT:`
	`CODEPAGE:850`
	`COMMENT: ------------------------------------------------------`
	`COMMENT: -- Starting page group`
	`COMMENT: ------------------------------------------------------`

Table 12: CpCOLD Profile: Attribute exptype - Format Examples	
SGML	<?xml version="1.0" encoding="IBM850" standalone="yes"?> <docbridge> <doclist name="CPCOLD_UNIT" creator="CPCOLD" created="Fri Oct 29 09:31:43 2004" desc="Created by Compart COLD Execution Unit"> <document class="DEF"> <alist> <attribute name="name" value="Hugo Maier"/> <attribute name="plz" value="47111"/> <attribute name="kundennummer" value="100.093.13"/> </alist> <flist pages='8'> <file type="unknown" name="../../data/civa/output\dummy.nul" /> </flist> </document> </doclist> </docbridge>
SOLITAS	Hugo Maier;47111;100.093.13;8;dummy.nul
WEBARCHIVE	<BAT> <COM>Created by Compart COLD Execution Unit at Fri Oct 29 09:33:45 2004 </COM> <DOC> <FTY>unsupported</FTY> <PTH>../../data/civa/output</PTH> <EXT>xxx</EXT> <FIL> <FNA>dummy.nul</FNA> <PCT>8</PCT> </FIL> <IDX> <IDN>name</IDN> <IDT>S</IDT> <IDV>Hugo Maier</IDV> <IDP>1</IDP> </IDX> <IDX> <IDN>plz</IDN> <IDT>S</IDT> <IDV>47111</IDV> <IDP>1</IDP> </IDX> <IDX> <IDN>kundennummer</IDN> <IDT>S</IDT> <IDV>100.093.13</IDV> <IDP>1</IDP> </IDX> </DOC> </BAT>

5.1.2. CpCOLD Profile Examples

<coldunit> Examples

```
<coldunit
   spooldir       ="spooldir"
   spoolmask      ="*.afp"
   logdir         ="logdir"
   loglevel       =3
   errdir         ="errdir"
   arcdir         ="arcdir"
   expdir         ="outdir"
   expfile        ="$EXPDIR + '\' + filenamepart($SPOOLFILE, 'N') + '.exp'"
   AFPIN
   extresfile     ="$SPOOLDIR + '\' + 'RESOURCE.RES'"
   SGML
   docdir         ="outdir"
   docfile        ="maketemp($DOCDIR + '\' + '%08d.tif')"
   TIFF
   resolution     =240
   pagecx         =297
   pagecy         =210
   defaultclass   ="NOTFOUND"
   >
   <searchlist>

   <!-- The variable line is assigned to the content of a line, if this contains the word
        LBSARCHIV. -->

      <string name="line" value="LBSARCHIV">

   <!-- The variable separator is assigned to the content of a line, if this contains the string
        your account no. -->

      <string name="separator" value="your account no.:">
   </searchlist>

   <filter>

   <!-- Page is suppressed, if the content of variable line is a empty string, i.e. if no line of
        the page contains the string LBSARCHIV. -->

      <pagesup   condition="length(line) == 0">

   </filter>

   <rules>

   <!-- A new document is generated, if the variable separator contains a value, i.e. if there is
        one line on the page which contains your account no.:. -->

      <docstart  condition="length(separator) != 0">

   <!-- The name of the class of the document is the 2nd word of the variable line. -->

      <classname value="word(line, 2)">
   </rules>
   <fontlist>

   <!-- Font-ID 1 is assigned to C0D@GT10/T1D@BASE. -->

      <font fontid=1  charset="C0D@GT10" codepage="T1D@BASE" >
      <font fontid=13 charset="C0H2006M" codepage="T1D@BASE" >
   </fontlist>
   <classlist>

   <!-- Class VER is specified with a rotation of 90 degrees. So, an index list is generated with
        indexes DocType and AccountNo. -->

      <class name="VER" rotation=90 >
         <indexlist>

   <!-- The third of line is assigned to index DocType. -->
```

```
            <index name="DocType"  value="word(line, 3)">
            <index name="AccountNo" value="word(line, 4)">
         </indexlist>
      </class>

      <class name="SCH" rotation=90 >
         <indexlist>
            <index name="DocType"  value="word(line, 3)">
            <index name="AccountNo" value="word(line, 4)">
         </indexlist>
      </class>

   <!-- Definition of the class NOTFOUND. This class is used, if a empty string for classname is
        found in <rules>.-->

      <class name="NOTFOUND" rotation=90 >
         <indexlist>
            <index name="DocType"  value="'not found'">
            <index name="AccountNo" value="'Doc:'+$DOCNO + '  Page:'+$DOCPAGE">
         </indexlist>
      </class>

   </classlist>

</coldunit>
```

5.1.3. Formula Interpreter

For the evaluation of expressions, CpCOLD uses a self-developed formula interpreter that has access to all specified variables. In conjunction with a set of functions, these can be used in expressions with syntax similar to REXX. The following list contains all attributes that can be used for assignment of values in this sense. Keep in mind that with the usage of variables they have to be specified in the expressions of processing or that they already have a value. Thus e.g. the usage of variable *$DOCPAGE* makes not sense in a term for assigning a value to the attribute *extresfile*, because at the time, when the term for *extresfile* is evaluated, still no document is processed.

5.1.3.1. Expressions

In expressions constants (*string*; *number*), variables, and functions can be used. All components can be combined or influenced by operators respectively unary operators.

Constants

Constants of type *string* are to be embedded in single quotes.
Example: ‘this is a string constant’.

Constants of type *number* are represented by a sequence of numbers.
Example: 123.

Variables

Variables are represented by their names.
Example: line; $SPOOLDIR etc.

Functions

Functions are represented by a keyword (function name) with a left parenthesis, a parameter list (which can be empty) and a right parenthesis.
Example: abs(-12)

The table below lists attributes, in which expressions can be used.

Attribute	In Element	Type	Time of Evaluation
Expfile	coldunit	String	Spool name is determined.
Extresfile	coldunit	String	Spool name is determined.
Docfile	coldunit	String	First page of the current document is read. ➔ Page has passed the pagesup filter.
Condition	pagesup (filter)	Boolean	Current page of the spool being processed is read. Segment and overlay are known.
Condition	Docsup (filter)	Boolean	Currently not implemented
Condition	docstart (rules)	Boolean	Like *docfile* (*coldunit*)
Value	classname (rules)	Boolean	Like *docfile* (*coldunit*)
Value	index (indexlist)	String	Current page of the spool being processed is read; Segment and overlay are known. ➔ Page has passed the pagesup filter.
Name	index (indexlist)	String	Current page of the spool being processed is read; Segment and overlay are known. ➔ Page has passed pagesup filter.

5.1.3.2. Operators

Logical AND:	&&
Unequal:	!=
Less or equal:	<=
Equal (Comparison):	==
Greater or equal:	>=
Logical OR:	||
Modulo:	%
Bit by bit AND:	&
Multiplication:	*
Addition (number):	+
Addition (string):	+
Subtraction:	-
Division:	/
Lower:	<
Greater:	>
Bit by bit XOR:	^
Bit by bit OR:	|

Unary Operators

Positive:	+
Negative:	-
Logical NOT:	!
Complement:	~

5.1.4. CpCOLD Functions

If an attribute can be assigned to a term, which can be evaluated, then the following functions can be used anytime (with regard to the type).

The type of functions is either *string* or *number*. The type *number* realizes also the type *Boolean* (special numeric value: `!= 0 -> TRUE; 0 -> FALSE`).

- *Chapter 5.1.4.1. REXX Related Functions on page 170* contains functions regarding to name, parameter, and functionality correspond to those of the REXX interpreter.
- *Chapter 5.1.4.2. Compart Functions on page 180* contains general Compart functions.
- *Chapter 5.1.4.3. CpCOLD Specific Functions on page 185* contains additional specific CpCOLD functions.

5.1.4.1. REXX Related Functions

Function abs

Call: **abs(Value);**

Type of function: number

Parameter	Type	Status	Remark
`Value`	number	Required	Value should be an integer.

Description: **abs** returns the absolute value of the (signed) integer specified in `Value`.

Function center

Call: **center(String, Len, Pad);**

Type of function: string

Parameter	Type	Status	Remark
`String`	string	Required	
`Len`	number	Required	
`Pad`	string	Optional	Character to be inserted, if not blank.

Description: **center** centers a string within the specified size (`Len`). If `Len` is smaller than the length of `String`, it will be truncated. If `Len` is longer than the length of `String` and `Pad` is not specified, blanks are inserted. Otherwise, the first character of `Pad` is used.

Function compare

Call: **compare(String1, String2, Pad);**

Type of function: number

Parameter	Type	Status	Remark
String1	string	Required	
String2	string	Required	
Pad	string	Optional	Character not significant for comparison.

Description: **compare** compares two strings (`String1` and `String2`) and returns a value unequal 0, if the strings are different (0). The returned value corresponds to the position of the first character identified to different. Blanks appended to strings are ignored, e.g.`'ABC  ' = 'ABC'`. If `Pad` is specified, the first character of *Pad* acts as not-significant character, e.g. `Pad='a'`results in `'ABC' = 'ABCaaa'`.

Function copies

Call: **copies(String, Repeat);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
Repeat	number	Required	Repeat has to be positive and integer.

Description: **copies** copies the specified string `String` with the specified value of `Repeat` into the resulting string. If `Repeat` has the value 0 (or less), the resulting string is deleted.

Function delstr

Call: **delstr(String, Pos, Len);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
Pos	number	Required	
Len	number	Optional	Len has to be positive and integer.

Description: **delstr** deletes all characters starting from `Pos` with length `Len` from the specified string `String`.

Function delword

Call: **delword(String, PosWord, CountWords);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
PosWord	number	Required	
CountWords	number	Optional	CountWords has to be positive and integer.

Description: **delword** deletes in the specified string `String` as many words as specified in `CountWords`, starting with the word specified by `PosWord`. If `CountWords` is not specified, all words are deleted up to the end of the string. If `CountWords` is greater than the number of words in `String`, `String` remains unchanged.

```
delword('now is the time', 2, 2)    -> 'Now time'
delword('now is the time', 3)       -> 'Now is'
delword('now is the time', 5)       -> 'Now is the time'
```

Function insert

Call: **insert(InsertString, String, Pos, Len, Pad);**

Type of function: string

Parameter	Type	Status	Remark
InsertString	String	Required	
String	String	Required	
Pos	number	Optional	Pos must be positive and integer.
Len	number	Optional	Len must be positive and integer.
Pad	String	Optional	

Description: **insert** inserts the string `Insertstring` into `String` after the character positioned at `Pos` that is filled up to the length of `Len`. If `Pos` is greater than the length of `String`, fill characters are inserted. If `Pad` (fill characters) is not specified, blank is the fill character. If `Pos` is not specified, 0 is default. `InsertString` is inserted before the first character of `String`.

```
insert(' ',   'abcdef', 3)          -> 'abc def'
insert('123', 'abc', 5, 6)          -> 'abc  123   '
insert('123', 'abc', 5, 6, '+')     -> 'abc++123+++'
insert('123', 'abc')                -> '123abc'
insert('123', 'abc', ,5, '-')       -> '123--abc'
```

Function lastpos

Call: **lastpos(Search, String, Pos)**

Type of function: number

Parameter	Type	Status	Remark
Search	string	Required	
String	string	Required	
Pos	number	Optional	Pos has to be positive and integer.

Description: **lastpos** passes the position of the last appearance of `Search` in `String`. If `Search` is not found in `String`, `lastpos` returns 0. The search starts at the last character of `String`. To define a different start position `Pos` has to be specified.

```
lastpos(' ', 'abc def ghi')         -> '8'
lastpos(' ', 'abcdefghi')           -> '0'
lastpos(' ', 'abc def ghi', 7)      -> '4'
```

Function left

Call: **left(String, Len, Pad);**

Type of function: string

Parameter	Type	Status	Remarks
String	string	Required	
Len	number	Required	Len has to be positive and integer.
Pad	string	Optional	

Description: **left** passes a string of length `Len` that contains the most left positioned characters in `String`. The passed string is provided with fill characters (`Pad`) starting at the right most character. If `Pad` is not specified, blank is default.

```
left('abc d', 8)            -> 'abc d   '
left('abc d', 8, '.')       -> 'abc d...'
left('abc def', 6)          -> 'abc de'
```

Function length

Call: **length(String);**

Type of function: number

Parameter	Type	Status	Remark
String	string	Required	

Description: **length** passes the length of `String`.

```
length('abcdefgh')          ->  8
length('abc  efgh')         ->  8
length('')                  ->  0
```

Function max

Call: **max(Value1, Value2);**

Type of function: number

Parameter	Type	Status	Remark
Value1	number	Required	
Value2	number	Required	

Description: **max** passes the greater value of `Value1` and `Value2`.

```
max(12, 21)          -> 21
```

Function min

Call: **min(Value1, Value2);**

Type of function: number

Parameter	Type	Status	Remark
Value1	number	Required	
Value2	number	Required	

Description: **min** passes the lesser value of `Value1` and `Value2`.

```
min(12, 21)          -> 12
```

Function overlay

Call: **overlay(Overlay, String, Pos, Len, Pad);**

Type of function: string

Parameter	Type	Status	Remark
Overlay	string	Required	
String	string	Required	
Pos	number	Optional	Pos has to be positive and integer.
Len	number	Optional	Len has to be positive and integer or 0.
Pad	string	Optional	

Description: **overlay** passes a string where `String` at position `Pos` is superposed by `Overlay`. `Overlay` is truncated or filled up by the specified length of `Len`. If `Pos` is greater than the target string, fill characters are inserted ahead of `Overlay`. If `Pad` (fill character) is not specified, blank is used. If `Len` is not specified, the length of `Overlay` is used.

```
overlay(' ', 'abcdef', 3)          -> 'ab def'
overlay('x', 'abcdef', 3, 2)       -> 'abx ef'
overlay('qq', 'abcd')              -> 'qqcd'
overlay('qq', 'abcd', 4)           -> 'abcqq'
overlay('123', 'abc', 5, 6)        -> 'abc+123+++'
```

Function pos

Call: **pos(Search, String, Pos);**

Type of function: number

Parameter	Type	Status	Remark
Search	string	Required	
String	string	Required	
Pos	number	Optional	Pos has to be positive and integer.

Description: **pos** calculates the position of `Search` in `String`. The search starts at the specified position of `Pos`. If `Pos` is not specified, the calculation starts with the first character.

```
pos('day', 'Saturday')              -> 6
pos('x', 'abc def ghi')             -> 0
pos(' ', 'abc def ghi')             -> 4
pos('x', 'abc def ghi', 5)          -> 8
```

Function reverse

Call: **reverse(String);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	

Description: **reverse** passes the string `String` in reverse character order.

```
reverse('Abc.')      -> '.cBA'
reverse('XYZ')       -> '.ZYX'
```

Function right

Call: **right(String, Len, Pad);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
Len	number	Required	Pos has to be positive and integer.
Pad	string	Optional	

Description: **right** passes a string with the length `Len` that contains the characters positioned at the most right position in `String`. The passed string is provided with fill characters (`Pad`) starting at the left most character. If `Pad` is not specified, blank is used.

```
right('abc d', 8)           -> '   abc d'
right('abc def', 5)         -> 'c def'
right('12', 5, '0')         -> '00012'
```

Function space

Call: **space(String, CountSpaces, Pad);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
CountSpaces	number	Optional	CountSpaces has to be positive and integer or 0.
Pad	string	Optional	

Description: **space** formats words in `String` that are separated by blanks with `CountSpaces` fill characters. If 0 is specified for `CountSpaces`, all fill characters are deleted. If `CountSpaces` is omitted, 0 is default. If `Pad` (fill characters) is not specified, blank is used.

```
space('abc    def    ')            -> 'abc def'
space('  abc def', 3)              -> 'abc   def'
space('  abc def   ', 1)           -> 'abc def'
space('  abc def   ', 0)           -> 'abcdef'
space('  abc def   ', 2, '+')      -> 'abc++def'
```

Function strip

Call: **strip(String, Opt, Char);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
Opt	string	Optional	Opt can have the following values: `b`: leading and following `l`: leading `t`: following
Char	string	Optional	

Description: Based on `Opt`, **strip** deletes all leading and/or following characters from `String`. If `Opt` is omitted, `b`(oth) is default. If `Char` is not specified, blank is used.

```
strip('  ab c    ')                -> 'ab c'
strip('  ab c    ', 'L')           -> 'ab c   '
strip('  ab c    ', 't')           -> '  ab c'
strip('12.7000',, '0')             -> '12.7'
strip('0012.700',, '0')            -> '12.7'
```

Function substring

Call: **substring(String, Pos, Len, Pad);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
Pos	number	Required	Pos has to be positive and integer.
Len	number	Optional	
Pad	string	Optional	

Description: **substring** passes the part of `String` that starts at the specified position of `Pos` and has the length of `Len`. If necessary, `Pad` (fill characters) is used. If `Pad` is not specified, blank is used. If `Len` is not specified, the rest of `String` is passed.

```
substring('abc', 2)               -> 'bc'
substring('abc', 2, 4)            -> 'bc  '
substring('abc', 2, 6, '.')       -> 'bc....'
```

Function subword

Call: **subword(String, PosWord, CountWords);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
PosWord	number	Required	PosWord has to be positive and integer.
CountWords	number	Optional	

Description: **subword** passes the part of `String` starting at the specified position of `PosWord` and that consists of `CountWords` words separated by blanks. If `CountWords` is not specified, the rest of the words remaining in `String` are passed. The passed string has no leading or following blanks. The blanks between the words remain.

```
subword('Now is the    time', 2)    -> 'is the'
subword('Now is the    time', 3)    -> 'the    time'
subword('Now is the    time', 5)    -> ''
```

Function translate

Call: **translate(String, Out, In, Pad);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
Out	string	Optional	
In	string	Optional	
Pad	string	Optional	

Description: **translate** translates the characters in `String` into other characters or rearranges characters in a string. If none of the translation tables (`Out` (output table), `In` (input table)) is specified, `String` is converted to uppercase. The character range *\x00 .... \xFF* is default for `In`. The output table contains the zero character string by default and is filled up with the specified fill characters (`Pad`) or truncated, if required. If `Pad` is not specified, blank is used. The tables can have any length. If characters are coded more than once in a table, the first found character is used.

```
translate('abcdef')                         -> 'ABCDEF'
translate('abbc', '&', 'b')                 -> 'a&&c'
translate('abcdef', '12', 'ec')             -> 'ab2d1f'
translate('abcdef', '12', 'abcd','.')       -> '12..ef'
translate('4123', 'abcd', '1234')           -> 'dabc'
```

Function word

Call: **word(String, PosWord);**

Type of function: string

Parameter	Type	Status	Remark
String	string	Required	
PosWord	number	Required	PosWord has to be positive and integer.

Description: **word** passes the word from `String` specified at position `PosWord`. If fewer words are inside `String` than specified in `PosWord`, an empty string is returned.

```
word('now is the time', 3)          -> 'the'
word('now is the time', 5)          -> ''
```

Function wordindex

Call: **wordindex(String, PosWord);**

Type of function: number

Parameter	Type	Status	Remark
String	string	Required	
PosWord	number	Required	PosWord has to be positive and integer.

Description: **wordindex** passes the position of the first character of the identified word in `String` separated by blank.

```
wordindex('now is the time', 3)     -> 8
wordindex('now is the time', 6)     -> 0
```

Function wordlength

Call: **wordlength(String, PosWord);**

Type of function: number

Parameter	Type	Status	Remark
String	string	Required	
PosWord	number	Required	PosWord has to be positive and integer.

Description: **wordlength** passes the length of identified word in `String` separated by blanks.

```
wordlength('now is the time', 2)          -> 2
wordlength('now comes the time', 2)       -> 5
wordlength('now is the time', 5)          -> 0
```

Function words

Call: **words(String);**

Type of function: number

Parameter	Type	Status	Remark
String	string	Required	

Description: **words** passes the number of words in `String` separated by blanks.

```
words('now is the time')     -> 4
words('      ')              -> 0
```

5.1.4.2. Compart Functions

Function date

Call: **date(Type);**

Type of function: string

Parameter	Type	Status	Remark
Type	number	Optional	Values for Type: 0: 'YYYYMMDD' (Default) 1: 'DD.MM.YYYY' 2: 'DD.MM.YY' 3: 'DDMMYY' 4: 'YYMMDD' 5: 'YYYYDDD' (day of the year, i.e. 2008313)

Description: **date** returns the current date as string. The format of the returned date depends on the parameter `Type`. If `Type` is omitted, **date** specifies the value 0.

Function EXISTS

Call: **EXISTS(Var);**

Type of function: number (boolean)

Parameter	Type	Status	Remark
Var	any	Required	

Description: **EXISTS** checks the existence of the passed variables.

Function filenamepart

Call: **filenamepart(FileName, Components);**

Type of function: string

Parameter	Type	Status	Remark
FileName	string	Required	
Components	string	Required	Values for Components: f: complete file specification d: drive p: path n: name e: extension Any combination is possible. Example: `'dne'`.

Description: **filenamepart** returns the components specified in `Components` of the file specification of a passed file. **filenamepart** identifies the complete file specification based on the passed (incomplete) specification and writes the specified components into the result string.

Function IF

Call: **IF(TestVal, TrueVal, FalseVal);**

Type of function: number (boolean) or string

Parameter	Type	Status	Remark
TestVal	number	Required	
TrueVal	number	Required	
FalseVal	number	Required	

Description: The type of **IF** complies with the parameters `Trueval` and `FalseVal`. If both parameters have the type `number`, **IF** works like the function **ifnum**. The parameters `Trueval` and `FalseVal` must have the same type.

Function ifnum

Call: **ifnum(TestVal, TrueVal, FalseVal);**

Type of function: number (boolean)

Parameter	Type	Status	Remark
TestVal	number	Required	
TrueVal	number	Required	
FalseVal	number	Required	

Description: **ifnum** tests the passed value `TestVal` and returns `TrueVal`, if the value is unequal 0 (otherwise `FalseVal`). `Trueval` and `FalseVal` can be any numeric values.

Note: The function **IF** can be used instead of functions **ifnum** and **ifstr. IF** detects the type of parameter automatically and returns the correct type in `Trueval` or `FalseVal`. The parameters are the same as those of functions **ifnum a**nd **ifstr**.

Function ifstr

Call: **ifstr(TestVal, TrueVal, FalseVal);**

Type of function: string

Parameter	Type	Status	Remark
TestVal	number	Required	
TrueVal	string	Required	
FalseVal	string	Required	

Description: **ifstr** tests the passed value `TestVal` and returns `TrueVal`, if the value is unequal 0 (otherwise `FalseVal`). `Trueval` and `FalseVal` can be any kind of string.

Note: The function **IF** can be used instead of functions **ifnum** and **ifstr**. **IF** detects the type of parameter automatically and returns the correct type in `Trueval` or `FalseVal`. The parameters are the same as those of functions **ifnum a**nd **ifstr**.

Function int

Call: **int(StrVal);**

Type of function: number

Parameter	Type	Status	Remark
StrVal	string	Required	

Description: **int** converts a string into a number. The string may contain the digits 0-9 only. If `StrVal` contains other characters, the result is not defined. The function corresponds to **toString**.

Function isnumber

Call: **isnumber(StrVal);**

Type of function: number (boolean)

Parameter	Type	Status	Remark
StrVal	string	Required	

Description: **isnumber** returns the value `TRUE`, if `StrVal` contains digits 0-9 only. If `StrVal` contains other characters, the value `FALSE` is returned. The function can be used to check the function **int**.

Function item

Call: **item(Record, ItemPos, Delimiter);**

Type of function: string

Parameter	Type	Status	Remark
Record	string	Required	String containing one or more words separated by delimiter.
ItemPos	number	Required	Position of the searched items.
Delimiter	string	Required	Character used as delimiter.

Description: **item** returns the word in the `Record` specified by `ItemPos`. The returned string is empty, if `ItemPos` is positioned outside of `Record` or exactly between two sequent delimiters.

```
item('now;is;the;time', 2, ';')     -> 'is'
item(';now;is;the;time', 2, ';')    -> 'now'
item('now;;the;time', 2, ';')       -> ''
```

Function maketemp

Call: **maketemp(FileMask);**

Type of function: string

Parameter	Type	Status	Remark
FileMask	string	Required	Mask should contain within a valid file name %d or %0xd (for x 1...8). Example: CP%05d.tmp

Description: Using the defined mask **maketemp** generates a valid but not existing file name in the specified directory. The directory is also passed as part of the mask. If `FileMask` does not specify a directory, the file is generated in the current directory.

Function time

Call: **time(Type);**

Type of function: string

Parameter	Type	Status	Remark
Type	number	Optional	Values for Type: 0: 'HHMMSShh' (Default) 1: 'HH:MM' 2: 'HH:MM:SS' 3: 'HH:MM:SS.hh'

Description: **time** returns the current time as string. The format of the returned time depends on the parameter `Type`. If `Type` is omitted, **time** specifies the value 0.

Function toString

Call: **toString(NumVal);**

Type of function: string

Parameter	Type	Status	Remark
NumVal	number	Required	

Description: **toString** converts a number into a string. The function corresponds to the function **int**.

Function toUpperCase

Call: **toUpperCase(StrVal);**

Type of function: string

Parameter	Type	Status	Remark
StrVal	string	Required	

Description: **toUpperCase** converts all lower case letters of `StrVal` into capital letters.

```
toUpperCase('Abc')   -> 'ABC'
```

Function VARVALUE

Call: **VARVALUE(Var, InitVal);**

Type of function: string

Parameter	Type	Status	Remark
Var	string	Required	
InitVal	string	Required	

Description: Similar to **EXISTS** the function **VARVALUE** checks the existence of variables in the element `<var>`. If the variable exists, the value is returned. If the variable does not exist, the value of `InitVal` is returned. **Note:** The function can be used to define variables whose existence is not taken for granted, e.g. NOPs and TLEs. If a TLE does not exist, the processing of CpCOLD does not stop and the value of `InitVal` is used for the TLE variable.

5.1.4.3. CpCOLD Specific Functions

Function CommStr

Call: **CommStr(StrVal);**

Type of function: string

Parameter	Typ	Status	Remark
StrVal	string	Required	

Description: **CommStr** encloses a string with quotation characters (""). The use of the quotation characters is limited to specific profile positions, e.g. a value of the attribute `value`.

```
CommStr('abc')       -> '"abc"'
```

Note: Alternatively the system variable `$QUOTATIONMARK` can be used.

```
<var name="Test" value="$QUOTATIONMARK+'abc'+$QUOTATIONMARK">    -> '"abc"'
```

Function Dimension

Call: **Dimension(StrVar);**

Type of function: number

Parameter	Type	Status	Remark
StrVar	string array	Required	

Description: **Dimension** returns the number of field elements of `StrVar`, e.g. the number of NOPs that contains the variable `$PAGECOMMENT`.

Function error

Call: **error(NumVal);**

Type of function: string

Parameter	Type	Status	Remark
NumVal	number	Required	Any value, but not a valid return code.

Description: **error** can initiate a defined termination of CpCOLD processing. **Note:** The function is position sensitive in the profile. Valid position is in `value` attributes of the element `<var>`.

```
<var name="overrun_error" value="IF($GROUPPAGE==15,error(99),'valid') ">
```

Function FileDate

Call: **Filedate(FileName,Type);**

Type of function: string

Parameter	Type	Status	Remark
FileName	string	Required	File name of existing (valid) file
Type	number	Required	Values for Type: 0: 'YYYYMMDD' (Default) 1: 'DD.MM.YYYY' 2: 'DD.MM.YY' 3: 'DDMMYY' 4: 'YYMMDD'

Description: **FileDate** returns the current date as string. The format of the returned date depends on the parameter `Type`. If `Type` is omitted, **FileDate** specifies the value 0. The date of the last file access is used.

Function FileTime

Call: **FileTime(FileName,Type);**

Type of function: string

Parameter	Type	Status	Remark
FileName	string	Required	File name of existing (valid) file
Type	number	Required	Values for Type: 0: 'HHMMSShh' (Default) 1: 'HH:MM' 2: 'HH:MM:SS' 3: 'HH:MM:SS.hh'

Description: **FileTime** returns the current time as string. The format of the returned date depends on the parameter `Type`. If `Type` is omitted, **FileTime** specifies the value 0. The time of the last file access is used.

Function findword

Call: **findword(Record, Word);**

Type of function: number

Parameter	Type	Status	Remark
Record	string	Required	String containing one or more Words separated by blanks.
Word	string	Required	Word scans in string. No blanks are allowed.

Description: **findword** scans `Record` for `Word` and returns the position in `Record` where `Word` is found. If `Word` cannot be found, **findword** returns 0.

```
findword('now is the time', 'is')          -> 2
findword('now is the time', 'ist')         -> 0
```

Function Modulo

Call: **Modulo(NumVal, Base);**

Type of function: number

Parameter	Type	Status	Remark
NumVal	number	Required	
Base	number	Required	

Description: **Modulo** returns the divide remainder of the number specified in `NumVal` divided by the number specified in `Base`.

```
Modulo(6,2)          -> 0
Modulo(6,2)          -> 1
```

Function retString

Call: **retString(NumVal);**

Type of function: number

Parameter	Type	Status	Remark
NumVal	number	Required	Valid return code

Description: **retString** returns the description of a valid return code.

5.2. cpmill Profile Elements

The following tables list the profile elements and contain descriptions of elements and attributes. Elements are sorted in alphabetical order.

Element	Description
<docmill>	Root element
Subelements (direct and indirect)	
<application>, <application-renderer>, <application>, <application-renderer>, <barcode>, <barcode-positionx>, <barcode-positiony>, <barcode-rotation>, <barcode-string>, <compute>, <date>, <datetime>, <deletefilecondition>, <driver>, <erroraction>, <erroractionlist>, <eventhandler>, <eventhandlerlist>, <executable>, <execute>, <execute-on-error-formula>, <execute-on-ok-formula>, <expression>, <expressionlist>, <file>, <filename>, <filetype>, <globals>, <height>, <indexentry>, <indexlist>, <input-page-shiftvalue>, <input-page-shiftvalue-cx>, <input-page-shiftvalue-cy>, <lineextract>, <logchannel-file>, <logchannel-smtp>, <logfile>, <logging>, <lotus-notes>, <omr>, <omrbgareah>, <omrbgareaw>, <omrbgareax>, <omrbgareay>, <omrlocation>, <omrpositionx>, <omrpositiony>, <omrstring>, <page-insertion>, <page-insertion-condition>, <page-insertion-condition-after>, <page-insertion-empty-back>, <page-insertion-file-after>, <page-insertion-grant-first-page>, <page-insertion-template-name>, <page-insertion-template-position-y>, <page-insertion-template-position-x>, <pagefrom>, <pagerange>, <pagesize>, <pageto>, <paper-in-tray>, <paper-in-tray-set-condition>, <paper-in-tray-set-name>, <paper-out-tray>, <paper-out-tray-set-condition>, <paper-out-tray-set-name>, <papersize>, <parameter>, <parameterlist>, <parameters>, <ppml>, <ppml-file>, <processlist>, <reversecondition>, <script-repository>, <script-repository-item>, <selection-formula>, <skipcondition>, <stamp>, <stampcondition>, <stampposition>, <stamppositionx>, <stamppositiony>, <stamptemplatename>, <successaction>, <successactionlist>, <system>, <task>, <tasklist>, <tempdir>, <time>, <timer>, <timerlist>, <tray>, <tray-set-condition>, <tray-set-name>, <triggerlist>, <unit>, <unitlist>, <userscript>, <width>	

Element	Description
<application>	In contrast to element <systems> it contains application specific information.
Subelement	
<globals>	

Example: Element <application> and subelements

```
<docmill>
   <system sharedbinaries="..\shared"
           smartfileoperations="false">
   </system>
   <application>
      <globals singlepass="no" keepinput="no" audiosignal="no" verbose="yes">
         <triggerlist>
            <timer minutes = "0" seconds="10" symbolicname="fastpoll"/>
            <!-- file  name    = "trigger.dat"   symbolicname="manual"/ -->
         </triggerlist>
      </globals>
   </application>
</docmill>
```

Element	Description	
<application-renderer>	Configuration of Application Renderer	
Attributes		
Name	**Value**	**Description**
checkmsofficefiles	false	Default. No checking
	true	Check for Microsoft Office file correctness (DOC, .XLS) to avoid processing of improper files with Microsoft Office extensions.
killuserexitcmd-after	e.g. killa.cmd	You can specify a command to delete Microsoft Office files after terminating the application, e.g. temporary files.
killuserexitcmd-before	e.g. killb.cmd	You can specify for example a command to generate a screenshot and use it for error analysis.
recycleapplication-count	e.g. 128	Integer. With special configuration of the Microsoft application, it stays active. You can specify the number of completed Application Renderer cycles after that application is restarted, recommended values: 64 or 128.
suppresskillapplication	false	Default. Command cannot be deactivated.
	true	Disable killuserexitcmd.
timeoutseconds	e.g. 20	If a print job is not completed after a specified period, the application is aborted. Rendition process fails.

Element	Description	
<barcode>	Barcode definition	
Subelements		
<barcode-positionx>, <barcode-positiony>, <barcode-rotation>, <barcode-string>		
Attributes		
Name	**Value**	**Description**
check-digits	e.g. 1	Integer. Number of check digits depending on barcode type.
datamatrix-columns	e.g. 3	Integer. Number of columns for two-dimensional barcode.
datamatrix-convert-to-bcd	false	Default. No conversion
	true	Conversion from string to BCD (Binary Coded Digital)
datamatrix-encodings	ASCII, BASE256, C40	Supported encodings
datamatrix-rows		Integer. Number of rows for two-dimensional barcode.
display-human-readable-information	false	No displaying of HRI.
	true	Display HRI, e.g. digits below a standard EAN barcode.
enabled	false	Default. Barcode is not positioned on document page.
	true	Barcode data is analyzed but not positioned on document page.
height	e.g. 12mm	Height
quiet-zone	false	Default. No quiet zone.
	true	Zone being before and after the barcode.
ratio	e.g. 3	Integer. Ratio of black and white areas.
type	e.g. EAN 13	For Compart supported barcodes, see *Appendix D: Barcodes on page 284*.
width	e.g. 22mm	Width

Element	Description
<barcode-positionx>	Horizontal starting point of barcode
Subelement	**Description**
<script>	String. Defined formula for computing. Result of formula computing is integer with defined unit of measurement, if not otherwise specified. Always computed per page.

Element	Description
<barcode-positiony>	Vertical starting point of barcode
Subelement	**Description**
<script>	String. Defined formula for computing. Result of formula computing is integer with defined unit of measurement, if not otherwise specified. Always computed per page.

Element	Description
<barcode-rotation>	Barcode orientation
Subelement	**Description**
<script>	String. Defined formula for computing. Result of formula computing is integer with defined unit of measurement, if not otherwise specified. Allowed values are: 0, 90, 180, and 270. Always computed per page.

Element	Description
<barcode-string>	Barcode data without check digits
Subelement	**Description**
<script>	String. Defined formula for computing. Result of formula computing is integer with defined unit of measurement, if not otherwise specified. Always computed per page.

Example: Element <barcode> and subelements

```
<barcode width="22mm" height="12mm" type="EAN 13" ratio="3" check-digits="1"
         quiet-zone="false" enabled="true"
         display-human-readable-information="true">
   <barcode-positionx>
      <script>
         "12cm"
      </script>
   </barcode-positionx>

   <barcode-positiony>
      <script>
         "1cm"
      </script>
   </barcode-positiony>

   <barcode-rotation>
      <script>
         "0"
      </script>
   </barcode-rotation>

   <barcode-string>
      <script>
         "978020170073"
      </script>
   </barcode-string>
</barcode>
```

Element	Description
<compute>	After reading the page, variables are to be replaced. Before variable replacing formulas can be computed.

Element	Description
<driver>	It describes a custom specific processing driver with its own value set. Drivers can be configured using <parameterlist> and <parameter>. Each parameter has at least a name and value attribute. Specification of names, default values, and ranges of values is determined by the custom specific driver implementation.
Subelements	
<barcode>, <expressionlist>, <indexlist>, <lineextract>, <page-insertion>, <pagesize>, <parameterlist>, <ppml>	
Attribute	

Name	Value	Description
type	121	One-to-one conversion
	APPREND	Application Renderer: DOC, XLS → Document
	BATCH	Directory containing raster images converted to Compart Batch Format (XML, scan batches)
	CLU	Cluster or CSV
	N21	"N to 1"m cumulate a lot of files into one single file
	NOTES	DocBridge Notes installation is required
	UXT	User-defined executable

Example: Element <driver> and subelements

```
<processlist>
   <driver type="BATCH">
      <parameterlist>
         <parameter name="in.search.mask" value="tif2batch\in\*.tif"/>
         <parameter name="out.put.dir"    value="tif2batch\out"/>
         <parameter name="special"        value="johndoe"/>
      </parameterlist>
   </driver>
</processlist>
```

Element	Description
<eventhandlerlist>	List of event handlers
Subelement	
<eventhandler>	

Element	Description
<eventhandler>	Event Handler
Subelement	
<script>	
Attribute	

Name	Value	Description
type	beginprocess	Once per cpmill run
	openoutputdocument	Once for every output file, before the file is opened
	openinputdocument	Once for every input file, before the file is opened
	inputdocumentopened	Once for every input file, after the file was opened
	outputdocumentopened	Once for every output file, after the file was opened
	documentstart	Once per document
	documentpageread	Once for every page
	documentend	Once per document
	endprocess	Once per cpmill run
	documentindex	Applicable if driver type="NOTES".

Element	Description
<execute>	Execution definition. Driver `type="UXT"` can call contents of element `<executable>`, applicable if driver `type="UXT"`.
Subelement	
`<executable>`	

Element	Description
<executable>	Name of executable program, applicable if `driver type="UXT"`.
Subelement	
`<script>`	

Example: Element <execute> and subelements

```
<execute>
   <!--
      Use Ghostscript to convert *.pdf to *.ps
      -sDEVICE=pswrite  = Postscript output
      -sDEVICE=pdfwrite = PDF output
   -->
   <executable>
      <script>"gs"</script>
   </executable>

   <parameters>
      <script>"-dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=" + $ROOT_DIR
            + "/output/" + $DOCUMENT_NAME_MAIN + ".ps " + $ROOT_DIR + "/input/"
            + $DOCUMENT_NAME_MAIN + ".pdf"</script>
   </parameters>
</execute>
```

Element	Description
<expressionlist>	List of `<expression>` elements.
Subelement	
`<expression>`	

Element	Description	
<expression>	Element computes formula and stores the result of computing in a named variable. The variables can be used for index or cover page generating.	
Attribute		
Name	**Value**	**Description**
`name`	e.g. `group`	Variable name where the result is stored.
Subelement		
`<script>`		

Example: Element <expressionlist> and subelements

```
<expressionlist>
   <expression name="DEMO_02">
      <script>
         Cprintf("Value Propagation 02 %s",CSVCOMPUTED);CSVCOMPUTED
      </script>
   </expression>
   <expression name="DEMO_03">
      <script>
         Cprintf("Value Propagation 03 %s",TestVar);TestVar
      </script>
   </expression>
</expressionlist>
```

Element	Description
<filetype>	Applicable if `driver type="CLU"`. Element contains subelement that specifies a file type. Valid returns values are: `ARE` (Application Renderer), `MFF` (MFF Filter Profile).
Subelement	
`<script>`	See example below: If the field `FileName` ends with `.DOC`, `ARE` is used, otherwise `MFF`.

Example: Element <filetype> and subelements

```
<driver type="CLU">

   <filetype>
      <script>
         IF(translate(filenamepart(FileName,"E")) == ".DOC", "ARE", "MFF")
      </script>
   </filetype>

</driver>
```

Element	Description	
<globals>	Element specifies unit or task independent settings.	
Subelement		
`<triggerlist>`		
Attributes		
Name	**Value**	**Description**
`audiosignal`	`no`	No acoustic signal
	`yes`	Acoustic signal
`detached`	`no`	cpmill runs in foreground.
	`yes`	cpmill process runs as an service/daemon/detached context, no console log etc.
`emulate`	`no`	No emulation.
	`yes`	For test runs. No processing is executed.
`keepinput`	`no`	For test runs. cpmill runs in standard operating mode. After successful completion of processing the input file(s) is (are) deleted.
	`yes`	cpmill processes the input files and does not delete them after processing. It results in repeated processing of the same input file.
`singlepass`	`no`	cpmill runs 'forever' and adheres to the corresponding timers. The setting can be overridden by command line parameters.
	`yes`	For test runs. cpmill runs only once, executing each unit independently of timer and then ends.
`verbose`	`no`	cpmill generates normal journaling. The setting can be overridden by command line parameters.
	`yes`	cpmill generates detailed journaling.

Element	Description
<triggerlist>	List with trigger mechanisms. If nothing is specified, trigger list specified in element `<globals>` is valid.
Subelements	
`<file>`, `<time>`, `<timer>`	

Element	Description	
<file>	Specified file triggers the unit. Main loop checks the existence of `name="name"` and triggers the processing of the appropriate unit.	
Attributes		
Name	**Value**	**Description**
`name`	e.g. `trigger.dat`	File name
`symbolicname`	e.g. `manual`	Symbolic name

Element	Description	
<time>	Starting point of Unit processing	
Attributes		
Name	**Value**	**Description**
day	e.g. 1	Day
hour	e.g. 2	Hour
minute	e.g. 20	Minute
second	e.g. 0	Second
Note: Specify above attributes with standard value range.		
symbolicname	e.g. fastjob	Symbolic name

Element	Description	
<timer>	Point in time after the unit was started last time. Currently one <timer> entry is being supported.	
Attribute		
Name	**Value**	**Description**
days	e.g. 1	Days
hours	e.g. 23	Hours
minutes	e.g. 0	Minutes
seconds	e.g. 1	Seconds, smallest values 1 second, i.e. you can trigger events every second
Note: Specify above attributes with standard value range.		
symbolicname	e.g. slowpoll	Symbolic name

Example: Element <globals> and subelements

```
<globals singlepass="no"
         keepinput="no"
         audiosignal="no"
         verbose="yes">
   <triggerlist>
      <timer minutes="0"
             seconds="10"
             symbolicname="fastpoll"/>
      <file name="trigger.dat"
            symbolicname="manual"/>
   </triggerlist>
</globals>
```

Element	Description
<input-page-shiftvalue>	Subelements specify input document shifting after a possible rotation. The value is applied per page, evaluation per page as well, so each page can have its own shifting.
Subelements	
<input-page-shiftvalue-cx>, <input-page-shiftvalue-cy>	

Element	Description
<input-page-shiftvalue-cx>	Element specifies with subelement shifting in x-direction, unit 1/10mm.
Subelement	
<script>	

Element	Description
<input-page-shiftvalue-cy>	Element specifies with subelement shifting in y-direction, unit 1/10mm.
Subelement	
<script>	

Example: Element <input-page-shiftvalue> and subelements

```
<input-page-shiftvalue>
   <input-page-shiftvalue-cx>
      <script>
         "0"
      </script>
   </input-page-shiftvalue-cx>

   <input-page-shiftvalue-cy>
      <script>
         "0"
      </script>
   </input-page-shiftvalue-cy>
</input-page-shiftvalue>
```

Element	Description
<lineextract>	Only valid if driver type="CLU". Before evaluating one of the subelements a complete line is read from input CSV/cluster file. The values are made available as variables corresponding to the column titles for computing of the lineextract expressions.
Subelements	
<deletefilecondition>, <filename>, <filetype>, <pagefrom>, <pagerange>, <pageto>, <reversecondition>, <skipcondition>	

Element	Description
<deletefilecondition>	If the formula returns a non-null string, the file specified in element <filename> is deleted. If a null string is returned, the file is not deleted.
Subelement	**Description**
<script>	String. Defined formula for computing. Result of formula computing is integer with defined unit of measurement, if not otherwise specified.

Element	Description
<filename>	Element contains subelement that specifies a file name of CSV file used for <lineextract>.
Subelement	
<script>	

Element	Description
<pagefrom>	Simple specifications of the input file extraction from the just read CSV file. If <pagefrom> and <pageto> are not specified, the complete document is processed as described for element <filename>.
Subelement	
<script>	

Element	Description
<pagerange>	Specification of page range. **Note:** If you use <pagerange>, it overrides data specified with <pagefrom> and <pageto>.
Subelement	
<script>	

Element	Description
<pageto>	Simple specifications of the input file extraction from the just read CSV file. If <pagefrom> and <pageto> are not specified, the complete document is processed as described for element <filename>.
Subelement	
<script>	

Element	Description
<reversecondition>	Specification order of <pagefrom> and <pageto> is inverted.
Subelement	
<script>	

Element	Description
<skipcondition>	If <skipcondition> is specified and it results in a non-null string, the CSV input line is ignored. If <skipcondition> is not specified and it results in a null string, the CSV input line is processed.
Subelement	
<script>	

Example: Element <lineextract> and subelements

```
<!-- CSV extract -->
<lineextract>
<!-- allow to do some computations after line has been read, before
     values will be obtained -->
   <filename>
      <script>
         $ROOT_DIR + "\\docmill\\swk\\in\\"+$DOCUMENT_NAME_MAIN + ".afp"
      </script>
   </filename>

   <filetype>
      <script>
         "ARE"   <!-- MFF, ARE -->
      </script>
   </filetype>

   <pagefrom>
      <script>
         DruckseiteVon+1
      </script>
   </pagefrom>

   <pageto>
      <script>
            DruckseiteBis
      </script>
   </pageto>

   <skipcondition>
      <script>
         ""
      </script>
   </skipcondition>

   <deletefilecondition>
      <script>
         ""
      </script>
   </deletefilecondition>

   <reversecondition>
      <script>
         ""
      </script>
   </reversecondition>

</lineextract>
```

Element	Description	
<lotus-notes>	Lotus Notes processing	
Subelements		
<execute-on-error-formula>, <execute-on-ok-formula>, <selection-formula>		
Attributes		
Name	**Value**	**Description**
database-name	e.g. \...\DEMO.NSF	Path and name of database
docbridge-notes-tracefile-name	e.g. \...\testxx.trc	Path and name of trace file
render-attachments	false	Default. No attachment
	true	Attachment to be rendered

Element	Description
<execute-on-error-formula>	Applicable if driver type="NOTES". Subelement specifies behavior when processing is faulty.
Subelement	
<script>	

Element	Description
<execute-on-ok-formula>	Applicable if driver type="NOTES". Subelement specifies behavior when processing is fault free.
Subelement	
<script>	

Element	Description
<selection-formula>	Applicable if driver type="NOTES". Subelement specifies selection behavior.
Subelement	
<script>	

Example: Element <lotus-notes> and subelements

```
<lotus-notes
      database-name="c:\projects\svn_trees\current\dbnotes\sample\RDEMO2002.NSF"
      docbridge-notes-tracefile-name="c:\testxx.trc"
      render-attachments="false">
   <selection-formula>
      <script>
         "SELECT @if(cpdbnotes=\&quot;ok2\&quot;;@true;@true);"
      </script>
   </selection-formula>

   <execute-on-ok-formula>
      <script>
         "FIELD cpdbnotes:= \&quot;ok1\&quot;;@true"
      </script>
   </execute-on-ok-formula>

   <execute-on-error-formula>
      <script>
         "FIELD cpdbnotes:= \&quot;ko\&quot;;@true"
      </script>
   </execute-on-error-formula>
</lotus-notes>
```

Element	Description
<omr>	Optical Mark Recognition
Subelements	
<omrbgareah>, <omrbgareaw>, <omrbgareax>, <omrbgareay>, <omrlocation>, <omrpositionx>, <omrpositiony>, <omrstring>	
Attributes	

Name	Value	Description
distance	e.g. 12pt	Distance of OMR marks
enabled	false	Default. OMR is not positioned on document page.
	true	OMR data is analyzed but not positioned on document page.
height	z. B. 24pt	Height
linewidth	z. B. 18tw	Line width
whiteout	false	Default. No Blocking out.
	true	Blocking out blotches, punch holes, etc. on scanned pages

Element	Description
<omrbgareah>	Element specifies with subelement the height of area in which OMR mark is placed.
Subelement	
<script>	

Element	Description
<omrbgareaw>	Element specifies with subelement the width of area in which OMR mark is placed.
Subelement	
<script>	

Element	Description
<omrbgareax>	Element specifies with subelement the starting point of area in which OMR mark is placed in x-direction (horizontal).
Subelement	
<script>	

Element	Description
<omrbgareay>	Element specifies with subelement the starting point of area in which OMR mark is placed in y-direction (vertical).
Subelement	
<script>	

Element	Description
<omrlocation>	Element specifies with subelement where OMR mark is located on the page.
Subelement	
<script>	Valid values: bottom, left, right, top.

Element	Description
<omrpositionx>	Element specifies with subelement the starting point of OMR mark in x-direction.
Subelement	
<script>	

Element	Description
<omrpositiony>	Element specifies with subelement the starting point of OMR mark in y-direction.
Subelement	
<script>	

Element	Description
<omrstring>	Element specifies with subelement the OMR mark string.
Subelement	
<script>	

Example: Element <omr> and subelements

```
<omr height="7mm" linewidth ="28tw" distance="24pt" enabled="true">
   <omrpositionx>
      <script>
         "10mm"
      </script>
   </omrpositionx>

   <omrpositiony>
     <script>
        "0mm"
     </script>
   </omrpositiony>

   <omrstring>
      <script>
         IF ($DOCUMENTNUMBER > 2 , "  11  11 11 11 11 11111111",
             "11 11  11111111111111 ")
      </script>
   </omrstring>

   <omrlocation>
      <script>
         "left"
      </script>
   </omrlocation>
</omr>
```

Element	Description
<page-insertion>	Insertion of pages
Subelements	
<page-insertion-condition>, <page-insertion-condition-after>, <page-insertion-empty-back>, <page-insertion-file-after>, <page-insertion-grant-first-page>, <page-insertion-template-name>, <page-insertion-template-position-x>, <page-insertion-template-position-y>	

Element	Description
<page-insertion-condition>	Condition for pages to be inserted
Subelement	
<script>	

Element	Description
<page-insertion-condition-after>	Position of pages to be inserted
Subelement	
<script>	

Element	Description
<page-insertion-empty-back>	Inserting an empty back page for duplex printing.
Subelement	
<script>	

Element	Description
<page-insertion-file-after>	Name of file to be inserted
Subelement	
<script>	

Element	Description
<page-insertion-grant-first-page>	Inserting a front page for duplex printing.
Subelement	
<script>	

Element	Description
<page-insertion-template-name>	Name of template file
Subelement	
<script>	

Element	Description
<page-insertion-template-position-x>	Position of template file in x-direction (horizontal).
Subelement	
<script>	

Element	Description
<page-insertion-template-position-y>	Position of template file in y-direction (vertical).
Subelement	
<script>	

Example: Element <page-insertion> and subelements

```
<!-- coversheet -->
<page-insertion>
   <page-insertion-condition>
      <script>IF($PAGE==1, "yes", "")</script>
   </page-insertion-condition>

   <page-insertion-template-name>
      <script>"pagesep.xif"</script>
   </page-insertion-template-name>

   <page-insertion-template-position-x>
      <script>"100"</script>
   </page-insertion-template-position-x>

   <page-insertion-template-position-y>
      <script>"100"</script>
   </page-insertion-template-position-y>

   <page-insertion-grant-first-page>
      <script>"yes"</script>
   </page-insertion-grant-first-page>

   <page-insertion-empty-back>
      <script>"yes"</script>
   </page-insertion-empty-back>

   <page-insertion-condition-after>
      <script type="text/javascript">
         if(sys.$PAGE == sys.$PAGESINDOCUMENT) {
              "";
          } else {
```

```
            "";
         }
      </script>
   </page-insertion-condition-after>

   <page-insertion-file-after>
      <script>"insert.afp"</script>
   </page-insertion-file-after>

</page-insertion>
```

Element	Description
<pagesize>	Page size. Instead of specifying the width/height explicitly, the element <papersize> uses names for the predefined paper sizes (A4, letter, etc.). **Note:** This will override width/height.
Subelements	
<height>, <width>	

Element	Description
<height>	Subelement that specifies page vertical dimension.
Subelement	
<script>	

Element	Description
<width>	Subelement that specifies page horizontal dimension.
Subelement	
<script>	

Example: Element <pagesize> and subelements

```
<pagesize>
   <width>
      <script>"21cm"</script>
   </width>

   <height>
      <script>"42cm"</script>
   </height>
</pagesize>
```

Element	Description
<paper-in-tray>	Paper input tray
Subelements	
<paper-in-tray-set-condition>, <paper-in-tray-set-name>	

Element	Description
<paper-in-tray-set-condition>	Condition for paper input tray
Subelement	
<script>	

Element	Description
<paper-in-tray-set-name>	Name of paper input tray
Subelement	
<script>	

Example: Element <paper-in-tray> and subelements

```
<paper-in-tray>
   <paper-in-tray-set-condition>
      <script>
      <!-- anything not empty will trigger the tray setting -->
         "yes"
      </script>
   </paper-in-tray-set-condition>

   <paper-in-tray-set-name>
      <script>
         "mytray_"+TRAYNAME
      </script>
   </paper-in-tray-set-name>
</paper-in-tray>
```

Element	Description
<paper-out-tray>	Paper output tray
Subelements	
<paper-out-tray-set-condition>, <paper-out-tray-set-name>	

Element	Description
<paper-out-tray-set-condition>	Condition for paper output tray
Subelement	
<script>	

Element	Description
<paper-out-tray-set-name>	Name of paper output tray
Subelement	
<script>	

Example: Element <paper-out-tray> and subelements

```
<paper-out-tray>
   <paper-out-tray-set-condition>
      <script>
         <!-- anything not empty will trigger the tray setting -->
         "yes"
      </script>
   </paper-out-tray-set-condition>

   <paper-out-tray-set-name>
      <script>
         "mytray_"+TRAYNAME
      </script>
   </paper-out-tray-set-name>
</paper-out-tray>
```

Element	Description
<papersize>	Specification of predefined paper sizes, i.e. A4, letter, etc. **Note:** If you use <papersize>, it overrides data specified with <height> and <width>.

Example: Element <papersize> and subelements

```
<papersize>
   <script>
      "letter"
   </script>
</papersize>
```

Element	Description
<parameterlist>	List with <parameter> elements
Subelement	
<script>	

Element	Description
<parameter>	For more detailed information on the element <parameter>, see *chapter 5.2.1. cpmill Profile Element <parameter> on page 211*.

Example: Element <parameterlist> and subelements

```
<parameterlist>
   <parameter name="in.search.mask"                  value="TIFFIN\*.ddf"/>
   <parameter name="in.rotationmode"                 value="none"/>
   <parameter name="out.rotationmode"                value="none"/>
   <parameter name="out.put.error.dir"               value="BADTYPE"/>
   <parameter name="out.put.success.dir"             value="TIFFOUT"/>
   <parameter name="out.put.completion.file"         value="TIFFOUT\done.dat"/>
   <parameter name="out.process.format.filetype"     value="PDF"/>
   <parameter name="out.process.stamp.form.file"     value="tlcstamp2.pdf"/>
   <parameter name="out.modify.replacevars.implicit" value="false"/>
</parameterlist>
```

Element	Description
<parameters>	Element specifies command line input used to call executable program. In general, command consists of variables. Only applicable if driver type="UXT".
Subelement	
<script>	

Element	Description
<ppml>	Support of PPML ((Personalized Print Markup Language)
Subelement	
<ppml-file>	

Element	Description
<ppml-file>	Subelement specifies PPML file.
Subelement	
<script>	

Example: Element <ppml> and subelements

```
<ppml>
   <ppml-file>
      <script>
         "myppml.dat"
      </script>
   </ppml-file>
</ppml>
```

Element	Description	
<script>	Defined formula for computing. Result of formula computing is integer with defined unit of measurement, if not otherwise specified.	
Attributes		
Name	**Value**	**Description**
name	e.g. test	Script name
type	e.g. text/javascript	Script type

Element	Description
<script-repository>	Element contains subelements with code objects that are valid for several units.
Subelement	
<script-repository-item>	

Element	Description
<script-repository-item>	Element contains code object.
Subelement	
<script>	

Element	Description
<stamp>	Stamps or watermarks
Subelements	
<stampcondition>, <stampposition>, <stamppositionx>, <stamppositiony>, <stamptemplatename>	

Element	Description
<stampcondition>	Conditions for stamp
Subelement	
<script>	

Element	Description
<stampposition>	Stamp position
Subelement	
<stamppositionx>, <stamppositiony>	

Element	Description
<stamppositionx>	Stamp position in x-direction (horizontal)
Subelement	
<script>	

Element	Description
<stamppositiony>	Stamp position in y-direction (vertical)
Subelement	
<script>	

Element	Description
<stamptemplatename>	Name of stamp template file
Subelement	
<script>	

Example: Element <stamp> and subelements

```
<stamp>
   <stamppositionx>
      <script>
         IF(TRUE, "260", "0")
      </script>
   </stamppositionx>
   <stamppositiony>
       <script>
        IF(TRUE, "580", "0")
       </script>
   </stamppositiony>
   <stampcondition>
      <script>
         IF(FALSE, "1", "")
      </script>
   </stampcondition>
   <stamptemplatename>
      <script>
         IF((($PAGE % 2) == 0), "xdvfm_var.xif", "xdvfm_var2.xif")
      </script>
   </stamptemplatename>
</stamp>
```

Element	Description	
<system>	Processing of Compart DLLs	
Attributes		
Name	**Value**	**Description**
sharedbinaries	e.g. mff*.dll	Directory for Compart DLLs
smartfileopera-tions	false	Default. No processing
	true	The application does a delete/try exit cycle instead of lock/unlock retry as a workaround. The timeout for this operation is 60 sec (currently hard coded).
	On some systems, depending on underlying network management software and also depending on the driver software (APPREND); there might be problems with erasing files that have been locked/unlocked by cpmill.	

Element	Description
<tasklist>	List of tasks
Subelement	
<task>	

Element	Description	
<task>	Task corresponds to (processing) Unit. But it can contain JavaScript code.	
Attributes		
Name	**Value**	**Description**
disabled	false	Default. Processing of task
	true	No processing.
key	e.g. unit01	Configurable short key uniquely identifying this task for reference purposes (i.e. for example as a command line parameter or future use within cascading / referencing tasks). If not specified, 1-based index of the task within the profile will be used, i.e. "1", "2", etc.
name	e.g. mytask	Name of task
rootdir	e.g. \...\abc\task02	Optional drive/directory specification. Whenever a file or directory value specification within this unit is encountered, this rootdir value is used as prefix, if the file or directory name is not fully specified.
title	e.g. afp2pdf	Configurable title text for log files

Element	Description
<tray>	Element with subelements deprecated, see elements <paper-in-tray> and <paper-out-tray>.
Subelements	
<tray-set-condition>, <tray-set-name>	

Element	Description
<tray-set-condition>	Element deprecated.
Subelement	
<script>	

Element	Description
<tray-set-name>	Element deprecated.
Subelement	
<script>	

Element	Description
<unitlist>	Element contains subelements specifying one or more units.
Subelement	
<unit>	

Element	Description	
<unit>	Processing unit	
Subelements		
<erroractionlist>, <indexlist>, <logfile>, <logging>, <processlist>, <successactionlist>, <tempdir>, <triggerlist>		
Attributes		
Name	**Value**	**Description**
disabled	false	Default. Processing of unit
	true	No processing
key	e.g. unit01	Configurable short key uniquely identifying this unit for reference purposes (i.e. for example as a command line parameter or future use within cascading / referencing units). If not specified, 1-based index of the unit within the profile will be used, i.e. "1", "2", etc.
name	e.g. mytask	Name of unit
rootdir	e.g. \...\tlc\unit02	Optional drive/directory specification. Whenever a file or directory value specification within this unit is encountered, the value rootdir is used as prefix, if the file or directory name is not fully specified.
title	e.g. afp2pdf	Configurable title text for log files

Element	Description
<erroractionlist>	Configuration generic error handling. One or more error actions can be specified. Error ignoring options are set per unit.
Subelement	
<erroraction>	

Element	Description
<erroraction>	For all available units information is issued about the actions if an error occurs. If no value is specified for the attribute `type`, the error is moved up one level. Notes: • Processing order is established by the cpmill not by the order presented above (order is: `ignore copy move erase`). • Only one `type` value will be evaluated - so from two `copy` types, for example only one (the first them) will be really executed. This behavior is subject to improvements/changes. • No consistence check, i.e. `copy` after `move` will fail. • Errors on error handling undo a possible `ignore`.

Attributes

Name	Value	Description
`directory`	e.g. `afp2afp\errormove`	Target directory `type="move"`or `type="copy"`
`level`	`file`	Currently valid value
`type`	`copy`	Copy
	`erase`	Delete
	`ignore`	Terminate command is ignored, process runs
	`move`	Copy and delete
	`pause`	Waiting for operator action (debugging)

Example: Element <erroractionlist> and Subelements

```
<erroractionlist>
   <erroraction  level="file" type="ignore"/>
   <erroraction  level="file" type="pause"/>
   <erroraction  level="file" type="move"  directory="errormove"/>
   <erroraction  level="file" type="copy"  directory="errorcopy"/>
   <erroraction  level="file" type="erase" />
</erroractionlist>
```

Element	Description
<indexlist>	List of one or more subelements that specify index entry.
Subelement	
`<indexentry>`	
Attributes	

Name	Value	Description
`name`	e.g. `indexlst01`	Name of index list
`type`	`nop-document-end`	NOP position: document end
	`nop-page-inside`	NOP position: inside page
	`nop-page-outside`	NOP position: outside page
	`tle`	TLEs (Tag Logical Element)

Element	Description
<indexentry>	Specification of index entry. Subelement specifies contents of index entry.
Subelement	
`<script>`	
Attributes	

Name	Value	Description
`name`	e.g. `something01`	Name of index entry
`type`	`remove`	Remove NOPs/TLEs
	`set`	Default. Set NOPs/TLEs

Example: Element <indexlist> and subelements

```
<indexlist type="nop-page-outside">
   <indexentry name="Objectdescription">
      <script>
         "Objectdescription:=" + substring($DOCUMENT_NAME_MAIN,51,20);
      </script>
   </indexentry>
   <indexentry name="Folder">
      <script>
         "Folder:=" + substring($DOCUMENT_NAME_MAIN,39,12) + "00"
                    + substring($DOCUMENT_NAME_MAIN,33,6);
      </script>
   </indexentry>
</indexlist>
```

Element	Description	
<logfile>	Log data generation and handling of further processing.	
Attribute		
Name	**Value**	**Description**
name	e.g. convert.log	It specifies the name of the file where unit specific processing elements are written to. The value specified here will be prefixed with unit\rootdir, if not fully specified. The directory will be created automatically.

Element	Description
<logging>	Log data generation and handling of further processing.
Subelements	
<logchannel-file>, <logchannel-smtp>	

Element	Description	
<logchannel-file>	Element generates log file with specified name and contents according to defined severity level.	
Attributes		
Name	**Value**	**Description**
append	false	Default: No appending of messages.
	true	No new log file is written. The messages are added to an existing log file.
name	e.g. my.log	Log file name
severity-level-tags	F:	FATAL activated
	E:	ERROR activated
	I:	INFO activated
	V:	VERBOSE activated
	W:	WARNING activated
	*:	ALL activated
	f:	FATAL deactivated (useful only after turned all on with *)
	e:	ERROR deactivated (useful only after turned all on with *)
	i:	INFO deactivated (useful only after turned all on with *)
	v:	VERBOSE deactivated (useful only after turned all on with *)
	w:	deactivated (useful only after turned all on with *)
	Note: If you do not specify the attribute severity-level-tags, all severity levels are activated except VERBOSE.	

Element	Description
<logchannel-smtp>	Generation of log data to be send as e-mail.
Attribute	

Name	Value	Description
mail-from	e.g. log-ging@company.net	Sender
mail-subject	e.g. generated message	Subject
mail-to	e.g. abc@company.net	Addressee
server-login-password	e.g. topsecret	Server password
server-login-user	e.g. guest	Server user name
server-name	e.g. johndoe	Server name
severity-level-tags	e.g. FE	See element <logchannel-file> for severity levels.

Example: Element <logging> and subelements

```
<logging>
   <logchannel-file name="log/something.log" severity-level-tags = "EF"/>
   <logchannel-smtp
      severity-level-tags   = "EF"
      server-name           = "marvin"
      server-login-user     = ""
      server-login-password = "password"
      mail-from             = "logging@test.net"
      mail-to               = "user@test.net"
      mail-subject          = "msg-Compart DocBridge Mill/cpMill-VBLRTF2AFP:"
    />
</logging>
```

Note: When you do not specify the attribute severity-level-tags, all severity levels are activated with the exception of VERBOSE.

Element	Description
<processlist>	Element contains processing chain information within a <unit>. Currently valid: <driver>.
Subelement	
<script>	

Element	Description
<successaction-list>	List of actions when processing is successful.
Subelement	
<successaction>	

Element	Description
<successaction>	At file level information for all available units is issued that specify actions that can be executed when processing is successful.
Attributes	

Name	Value	Description
directory	e.g. afp2pdf\okcopy	Target directory type="move"or type="copy"
level	file	Currently valid value
type	copy	Copy
	erase	Delete
	ignore	Terminate command is ignored, process runs
	move	Copy and delete
	nothing	
	pause	Waiting for operator action

Element	Description	
`<tempdir>`	Directory with temporary files that must be available during processing	
Attribute		
Name	**Value**	**Description**
`name`	e.g. `temp`	Value will be prefixed with `unit\rootdir`, if not fully specified. The directory will be created automatically.

5.2.1. cpmill Profile Element <parameter>

The values of the attributes name and value that you can specify for the cpmill profile element <parameter> are described in the following tables.

Attribute: fontmanager.database		
name	**fontmanager.database**	
	Description: Specification of path and file name to use for the fontmanager's database file. The default name is ftmgr.xml.	
value	Path and file name	See example.
	Type	String.
	Default	File ftmgr.xml in the current directory.
Example	<parameter name="fontmanager.database" value="/your/path/to/ftmgr.xml"/>	

Attribute: general.binarycopymode		
name	**general.binarycopymode**	
	Description: Binary copy, i.e. if input files are ‚converted' into the same format as the input format, it is possible to copy identically any single page.	
value	always	Binary copy. If binary copy is not possible, the processing is terminated.
	auto	Default. cpmill detects, if binary copy is possible or not.
	never	No binary copy.
	Type	String.
Example	<parameter name="general.binarycopymode" value="never"/>	
See also *chapter 4.13.4.4. Special Settings and Methods on page 117*.		

Attribute: general.duplex		
name	**general.duplex**	
	Description: Output datastream is preprocessed for duplex printing.	
value	false	Default. No duplex printing.
	true	Duplex printing.
	Type	Boolean.
Example	<parameter name="general.duplex" value="false"/>	
See also *chapter 4.13.5.2. OutputDocumentAttributes on page 121*.		

Attribute: general.indexprocessing.auto		
name	**general.indexprocessing.auto**	
	Description: Contents of index fields (NOPs, TLEs) of input file are processed automatically.	
value	false	Default. No processing of index fields.
	true	Processing of index fields.
	Type	Boolean.
Example	<parameter name="general.indexprocessing.auto" value="false"/>	
See also *chapter 4.4. Index on page 89*.		

Attribute: general.inputfiles.keepopencount		
name	**general.inputfiles.keepopencount**	
	Description: When the attribute general.inputfiles.keepopenstrategy is specified with the value mru for the attribute value, it defines the number of files to keep open (it is ignored otherwise). For best performance, Compart suggests to use a count not too low (unless you are having memory constraints). Closing and reopening files can often have significant impact on the performance of the processing.	
value	e.g. 128	Open (active) files.
	Type	Integer.
	Default	128
Example	<parameter name="general.inputfiles.keepopencount" value="128"/>	
See also attribute general.inputfiles.keepopenstrategy.		

Attribute: general.inputfiles.keepopenstrategy		
Note: The term "keep open strategy" refers to how and how many input files will be kept open during a cpmcopy or cpmill.		
name	**general.inputfiles.keepopenstrategy**	
	Description: Specification if and how cpmill will keep open input files during processing. Three modes of operation are available: all, mru, and none. Effectively, the parameters only make sense for cpmill driver CLU (Cluster, CSV) processing. Therefore, the parameter is ignored for all cpmill drivers other than CLU. For non-CLU drivers, the internal strategy is now mru with value of 1 for the parameter general.inputfiles.keepopencount.	
value	all	Default. Keep all the input files open. It provides the best performance, but it may run out of file handles, if you have more input files than the operating system will let you keep open at once.
	Example: <parameter name="general.inputfiles.keepopenstrategy" value="all"/>	
	mru	Keep the most-recently used files open. This is often the best compromise. Note: Do not choose the value of keepopencount too low unless you are having memory constraints. Having to close and re-open files may have significant performance impacts.
	Example: <parameter name="general.inputfiles.keepopenstrategy" value="mru"/> <parameter name="general.inputfiles.keepopencount" value="50"/>	
	none	Close every input file again as soon as processing of that files has finished. The memory consumption is low, but it may be slow, if files have to be re-opened.
	Example: <parameter name="general.inputfiles.keepopenstrategy" value="none"/>	
	Type	String.
See also attribute general.inputfiles.keepopencount.		

Attribute: general.mcopy.profiledir		
name	**general.mcopy.profiledir**	
	Description: Directory for MFF profiles.	
value	Directory path	See example.
	Type	String.
	Default	Current directory.
Example	<parameter name="general.mcopy.profiledir" value="."/>	

Attribute: general.shellcommand.file.begin		
name	**general.shellcommand.file.begin**	
	Description: At file level specification and execution of shell command at a certain point of time.	
value	dynamic	Contents of element <script> is evaluated and processed as command.
	Command	Depending on operating system.
	Type	String.
	Default	Empty.
Example	<parameter name="general.shellcommand.file.begin" value="dynamic">	

Attribute: general.shellcommand.file.end		
name	**general.shellcommand.file.end**	
	Description: At file level specification and execution of shell command at a certain point of time.	
value	dynamic	Contents of element <script> is evaluated and processed as command.
	Command	Depending on operating system.
	Type	String.
	Default	Empty.
Example	<parameter name="general.shellcommand.file.end" value="dynamic">	

Attribute: general.shellcommand.file.unit.wakeup.begin		
name	**general.shellcommand.unit.wakeup.begin**	
	Description: At unit level specification and execution of shell command at a certain point of time.	
value	dynamic	Contents of element <script> is evaluated and processed as command.
	Command	Depending on operating system, see example.
	Type	String.
	Default	Empty.
Example	<parameter name="general.shellcommand.unit.wakeup.begin" value="cmd /c @echo system exit:begin unit wakeup"/>	

Attribute: general.shellcommand.file.unit.wakeup.end		
name	**general.shellcommand.unit.wakeup.end**	
	Description: At unit level specification and execution of shell command at a certain point of time.	
value	dynamic	Content of element <script> is evaluated and processed as command.
	Command	Depending on operating system, see example.
	Type	String.
	Default	Empty.
Example	<parameter name="general.shellcommand.unit.wakeup.end" value="cmd /c @echo system exit:end unit wakeup"/>	

Attribute: general.warningmode		
name	**general.warningmode**	
	Description: Level specification of error tolerant processing. The warning mode can be changed on a per-unit basis (overriding the mode as set from the command line like –R (Relaxed) or –S (Strict)). Since the unit parameter can override the global setting for the warning mode, you may see two messages referring to the warning mode in a cpmill logfile, e.g. 2008-09-24 10:17:00 DMI4016I 'Normal' warning mode enabled - will abort on certain warning messages (...) 2008-09-24 10:17:01 VAR0074I warning mode for current unit is 'relaxed' [general.warningmode] In the above case, the selected unit would be executed in Relaxed mode despite the earlier reference to Normal mode.	
value	normal	Normal mode. cpmill processing starts in Normal mode (while all previous versions started in Relaxed mode). For error messages related to a processing in Strict and Normal mode, see *Appendix F: Monitored Messages on page 286*.
	relaxed	Relaxed mode. For error messages related to a processing in Strict and Normal mode, see *Appendix F: Monitored Messages on page 286*.
	strict	Strict mode. For error messages related to a processing in Strict and Normal mode, see *Appendix F: Monitored Messages on page 286*.
	Type	String.
	Default	normal
Example	<parameter name="general.warningmode" value="relaxed"/>	

Attribute: in.afp.formdef.name		
name	**in.afp.formdef.name**	
	Description: Name of form definition.	
value	e.g. F1INFORM	Form definition name.
	Type	String. Maximum length: 8 characters.
	Default	Empty.
Example	`<parameter name="in.afp.formdef.name" value="F1INFORM"/>`	
See also *chapter 4.13.5.1. InputDocumentAttributes on page 119*.		

Attribute: in.afp.pgm.ianacodepagename		
name	**in.afp.pgm.ianacodepagename**	
	Description: IANA code page name for application program (cpmill).	
value	e.g. ibm850	See also *Appendix B: Code Page Names on page 277*.
	Type	String.
	Default	Empty.
Example	`<parameter name="in.afp.pgm.ianacodepagename" value="IBM850"/>`	

Attribute: in.afp.tle.ianacodepagename		
name	**in.afp.tle.ianacodepagename**	
	Description: IANA code page name for contents of NOP and TLE fields.	
value	e.g. ibm850	See also *Appendix B: Code Page Names on page 277*.
	Type	String.
	Default	Empty.
Example	`<parameter name="in.afp.tle.ianacodepagename" value="IBM850"/>`	

Attribute: in.cluster.csv.ianacodepagename		
name	**in.cluster.csv.ianacodepagename**	
	Description: IANA code page name for contents of CSV files.	
value	e.g. ibm850	See also *Appendix B: Code Page Names on page 277*.
	Type	String.
	Default	Empty.
Example	`<parameter name="in.cluster.csv.ianacodepagename" value="IBM850"/>`	

Attribute: in.extractresources.enabled		
name	**in.extractresources.enabled**	
	Description: Resources are extracted from input file.	
value	no	No resource extraction.
	yes	Resource extraction.
	Type	Boolean.
	Default	Empty.
Example	`<parameter name="in.extractresources.enabled" value="yes"/>`	
See also attribute in.extractresources.librarynametarget.		

Attribute: in.extractresources.librarynametarget		
name	**in.extractresources.librarynametarget**	
	Description: Storage position for extracted resources.	
value	dynamic	Content of element <script> is evaluated and path and file name are generated.
	Type	String.
	Default	Empty.
Example	`<parameter name="in.extractresources.librarynametarget" value="dynamic">`	
See also attribute in.extractresources.enabled.		

Attribute: in.process.format.filetype		
name	**in.process.format.filetype**	
	Description: Explicitly indicated type of input file.	
value	e.g. `iff`	Format type.
	Type	String.
	Default	Empty.
Example	`<parameter name="in.process.format.filetype" value="iff"/>`	

Attribute: in.process.format.subfiletype		
name	**in.process.format.subfiletype**	
	Description: Explicitly indicated subtype for type.	
value	e.g. `tif`	Format subtype.
	Type	String.
	Default	Empty.
Example	`<parameter name="in.process.format.subfiletype" value="tif"/>`	

Attribute: in.process.mode.sequential		
name	**in.process.mode.sequential**	
	Description: Some MFF filters are provided with Sequential Mode. Using this mode input files can be read faster in respect on resources. However, it is only valid for forward reading direction. Future cpmill versions will be able to identify the adequate mode for the optimal performance.	
value	`off`	Default. cpmill operates the MFF filters in Non-Sequential Mode which may result in a suboptimal performance.
	`on`	The performance can be improved but only for forward reading direction. In case of other direction, the processing can be aborted. Before using this parameter, Compart recommends to contact the company's support department.
	Type	Boolean
Example	`<parameter name="in.process.mode.sequential" value="off"/>`	

Attribute: in.rotation.angle		
name	**in.rotation.angle**	
	Description: Rotation of input document page. If `in.rotation.angle` is specified, `in.rotationmode` cannot be used and vice versa.	
value	`0`, `90`, `180`, `270`	Angle values.
	Default	`0`
Example	`<parameter name="in.rotation.angle" value="90"/>`	
See also see *chapter 4.2.4. Page Rotation on page 85*.		

Attribute: in.rotationmode		
name	**in.rotationmode**	
	Description: Rotation of input document page. If `in.rotationmode` is specified, `in.rotation.angle` cannot be used and vice versa.	
value	`auto`	Rotation according to main text alignment.
	`landscape`	Landscape format.
	`portrait`	Default. Portrait format.
	Type	String.
Example	`<parameter name="in.rotationmode" value="portrait"/>`	
See also see *chapter 4.2.4. Page Rotation on page 85*.		

Attribute: in.search.mask		
name	**in.search.mask**	
	Description: Definition of path and filemask to select input files.	
value	Path and filemask.	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="in.search.mask" value="in/*.csv"/>`	

Attribute: out.addresourcelibraryin.list		
name	**out.addresourcelibraryin.list**	
	Description: Comma separated list of resource names, e.g. fonts, form definitions, page definitions etc.	
value	Resource name	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.addresourcelibraryin.list" value="C0A48410,C0A58550,P1B26P37">`	

Attribute: out.afp.copygroup.name		
name	**out.afp.copygroup.name**	
	Description: Copy group name can be specified, if it is not inherited by the input file.	
value	Copy group name	See example.
	Type	String. Maximum length: 8 characters.
	Default	Empty.
Example	`<parameter name="out.afp.copygroup.name" value="STANDARD"/>`	
See also attribute `out.afp.noinheritcopygroup`.		

Attribute: out.afp.formdef.name		
name	**out.afp.formdef.name**	
	Description: Form definition name.	
value	Form definition name	See example.
	Type	String. Maximum length: 8 characters.
	Default	Empty
Example	`<parameter name="out.afp.formdef.name" value="F1OUTFRM"/>`	

Attribute: out.afp.noinheritcopygroup		
name	**out.afp.noinheritcopygroup**	
	Description: Inherit copy groups from input files.	
value	`false`	Default. Copy groups are forwarded.
	`true`	Copy groups are not forwarded.
	Type	Boolean
Example	`<parameter name="out.afp.noinheritcopygroup" value="true"/>`	
See also attribute `out.afp.copygroup.name`.		

Attribute: out.afp.nowriteDSbyte		
name	**out.afp.nowriteDSbyte**	
	Description: Write AFPDS byte `0x5A` in AFP output file.	
value	`false`	Default. Byte `0x5A` in output file.
	`true`	No byte in output file.
	Type	Boolean.
Example	`<parameter name="out.afp.nowriteDSbyte" value="true"/>`	

Attribute: out.afp.pagegroup.mask		
name	**out.afp.pagegroup.mask**	
	Description: Page group names of input files can be renamed that is numbered consecutively.	
value	Mask definition	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.afp.pagegroup.mask" value="G%07d"/>`	

Attribute: out.afp.pgm.ianacodepagename		
name	**out.afp.pgm.ianacodepagename**	
	Description: IANA code page name for application program (cpmill).	
value	e.g. `ibm850`	See also *Appendix B: Code Page Names on page 277*.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.afp.pgm.ianacodepagename" value="IBM850"/>`	

Attribute: out.afp.tle.ianacodepagename		
name	**out.afp.tle.ianacodepagename**	
	Description: IANA code page name for contents of NOP and TLE fields.	
value	e.g. `ibm850`	See also *Appendix B: Code Page Names on page 277*.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.afp.tle.ianacodepagename" value="IBM850"/>`	
See also JavaScript in DocBridge Mill		

Attribute: out.applicationrenderer.printerdriver.format		
name	**out.applicationrenderer.printerdriver.format**	
	Description: Application determines the output format. Keep in mind the format settings of the Compart printer driver.	
value	Output format	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.applicationrenderer.printerdriver.format" value="pdf"/>`	

Attribute: out.applicationrenderer.printerdriver.useforallformats		
name	**out.applicationrenderer.printerdriver.useforallformats**	
	Description: Compart printer driver is used for all output formats.	
value	`false`	
	`true`	
	Type	Boolean.
Example		
Note: The parameter and attributes are deprecated. The usage will have no effect.		

Attribute: out.applicationrenderer.printerdriver.usestandardprinter		
name	**out.applicationrenderer.printerdriver.usestandardprinter**	
	Description: Use of defined default printer.	
value	`false`	No switch of printer at all. No printing.
	`true`	Default. Printing with default printer.
	Type	Boolean.
Example	`<parameter name="out.applicationrenderer.printerdriver.usestandardprinter" value="true"/>`	

Attribute: out.applicationrenderer.printername		
name	**out.applicationrenderer.printername**	
	Description: Name of Compart printer driver.	
value	Printer driver name	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.applicationrenderer.printername" value="Compart DocBridge Print V001"/>`	

Attribute: out.convert.dither.clustered.size		
name	**out.convert.dither.clustered.size**	
	Description: Converting images with clustered order dithering.	
value	e.g. `3`	Values: 2 to 8.
	Type	Integer.
	Default	`3`
Example	`<parameter name="out.convert.dither.clustered.size" value="3"/>`	
See also *chapter 4.2.1.11. Dithering and Threshold on page 77*.		

Attribute: out.convert.dither.dispersed.size		
name	**out.convert.dither.dispersed.size**	
	Description: Converting images with dispersed order dithering.	
value	e.g. `4`	Values: 2 to 8.
	Type	Integer.
	Default	`4`
Example	`<parameter name="out.convert.dither.dispersed.size" value="4"/>`	
See also *chapter 4.2.1.11. Dithering and Threshold on page 77*.		

Attribute: out.convert.treshold.percent		
name	**out.convert.treshold.percent**	
	Description: Converting images with specification of threshold percent value.	
value	e.g. `50`	Percentage values: 0 - 100.
	Type	Integer.
	Default	`50`
Example	`<parameter name="out.convert.treshold.percent" value="50"/>`	
See also *chapter 4.2.1.11. Dithering and Threshold on page 77*.		

Attribute: out.file.addpagesforduplex		
name	**out.file.addpagesforduplex**	
	Description: Add empty page at the end of the output document, if the page number is odd.	
value	`false`	Default. No insert of empty page.
	`true`	Insert of empty page.
	Type	Boolean.
Example	`<parameter name="out.file.addpagesforduplex" value="false"/>`	

Attribute: out.file.meta.author		
name	**out.file.meta.author**	
	Description: Author information for output file (XMP file), currently PDF and PDF/A.	
value	Author information	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.file.meta.author" value="John Doe"/>`	
See also *chapter 4.13.4.3. XMP Data on page 117*.		

Attribute: out.file.meta.keywords		
name	**out.file.meta.keywords**	
	Description: Keyword information for output file (XMP file), currently PDF and PDF/A.	
value	Keyword information	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.file.meta.keywords" value="data extraction"/>`	
See also *chapter 4.13.4.3. XMP Data on page 117*.		

Attribute: out.file.meta.subject		
name	**out.file.meta.subject**	
	Description: Subject information for output file (XMP file), currently PDF and PDF/A.	
value	Subject information	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.file.meta.subject" value="Output Management"/>`	
See also *chapter 4.13.4.3. XMP Data on page 117*.		

Attribute: out.file.meta.title		
name	**out.file.meta.title**	
	Description: Title information for output file (XMP file), currently PDF and PDF/A.	
value	Title information	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.file.meta.title" value="DocBridge Mill"/>`	
See also *chapter 4.13.4.3. XMP Data on page 117*.		

Attribute: out.file.raster		
name	**out.file.raster**	
	Description: Generate document page as image.	
value	`false`	Default. No image generation.
	`true`	Image generation.
	Type	Boolean.
Example	`<parameter name="out.file.raster" value="true"/>`	
See also *chapter 4.2.1.10. Converting to Image on page 76.*		

Attribute: out.file.resources.selfcontained		
name	**out.file.resources.selfcontained**	
	Description: Write self-contained AFP files. Required (inline) resources are included into output file.	
value	`false`	Resources not in output file.
	`true`	Default. Resources in output file.
	Type	Boolean.
Example	`<parameter name="out.file.resources.selfcontained" value="false"/>`	
See also *chapter 4.13.4.4. Special Settings and Methods on page 117*.		

Attribute: out.file.separate.overlays		
name	**out.file.separate.overlays**	
	Description: Object in an overlay (currently limited to images) that is on more than one document page or in several input documents is available only once in datastream. For additional occurrences, image references are embedded in overlays.	
value	`false`	Positioning of objects as in input files.
	`true`	Default. One-time positioning.
	Type	Boolean.
Example	`<parameter name="out.file.separate.overlays" value="false"/>`	
See also *chapter 4.13.4.4. Special Settings and Methods on page 117*.		

Attribute: out.ipds.printer.name		
name	**out.ipds.printer.name**	
	Description: Printer name for IPDS printer.	
value	Printer name	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.ipds.printer.name" value="InfoPrint1585"/>`	

Attribute: out.modify.replacevars.implicit		
name	**out.modify.replacevars.implicit**	
	Description: Replacement of variables in CSV files, only valid for cpmill driver CLU (Cluster or CSV). cpmill detects a variable name in the presentation area (PA) and locates the name in the CSV file to replace it by the current value. The replacement can only be executed, if the variable name exists as a continuous string in the PA.	
value	`false`	Default. No variable replacement.
	`true`	Variable replacement.
	Type	Boolean.
Example	`<parameter name="out.modify.replacevars.implicit" value="false"/>`	

Attribute: out.process.format.filetype		
name	**out.process.format.filetype**	
	Description: Explicitly specified type of output file.	
value	e.g. `pdf`	Format type. Note: The format type specified with the Application Renderer cannot be overridden.
	Type	String.
	Default	TIFF, if the parameter is not specified.
Example	`<parameter name="out.process.format.filetype" value="PDF"/>`	

Attribute: out.put.dir		
name	**out.put.dir**	
	Description: Directory for output files. Default directory is `out`. If the output folder does not exist, it is created automatically.	
value	e.g. `pdf_io`	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.put.dir" value="pdf_io"/>`	

Attribute: out.put.filename.mask		
name	**out.put.filename.mask**	
	Description: Input file name can be overwritten for output. The structure of default file names depends on the cpmill driver.	
value	File name mask	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.put.filename.mask" value="X_X_BO&amp;$COUNTER_0000;_D&amp; $SDATE_YYMMDD;_T&amp;$STIME_HHMM;B00C00.&amp; $FILEEXTENSION;"/>`	

Attribute: out.put.font.id		
name	**out.put.font.id**	
	Description: The last two characters of the 8-character font ID can be set for the AFP output file.	
value	ID characters	See example.
	Type	String.
	Default	Empty.
Example	`<parameter name="out.put.font.id" value="XY"/>`	

Attribute: out.resolution.x.dpi		
name	**out.resolution.x.dpi**	
	Description: Resolution for output document in x-direction.	
value	e.g. 240	Unit: dpi.
	Type	Integer.
	Default	0
Example	<parameter name="out.resolution.x.dpi" value="240"/>	
See also *chapter 4.2.1.20. Resolution on page 83*.		

Attribute: out.resolution.y.dpi		
name	**out.resolution.y.dpi**	
	Description: Resolution for output document in y-direction.	
value	e.g. 240	Unit: dpi.
	Type	Integer.
	Default	0
Example	<parameter name="out.resolution.y.dpi" value="240"/>	
See also *chapter 4.2.1.20. Resolution on page 83*.		

Attribute: out.rotation.angle		
name	**out.rotation.angle**	
	Description: Rotation of output document page. If out.rotation.angle is specified, out.rotationmode cannot be used and vice versa.	
value	0, 90, 180, 270	Angle values.
	Type	Integer.
	Default	0
Example	<parameter name="out.rotation.angle" value="180"/>	

Attribute: out.rotationmode		
name	**out.rotationmode**	
	Description: Rotation of output document page. If out.rotationmode is specified, out.rotation.angle cannot be used and vice versa.	
value	auto	Rotation according to main text alignment.
	landscape	Landscape format.
	none	Default. No rotation.
	portrait	Portrait format.
	Type	String.
Example	<parameter name="out.rotationmode" value="none"/>	

5.2.2. cpmill Profile Drivers

The driver configuration with element `<driver>` and subelements is one of the core elements of the cpmill profile. The following drivers are available:

- `CLU`: Cluster or CSV
- `121`: one-to-one
- `APPREND`: Application Renderer
- `N21`: cumulate a lot of files into one single file
- `NOTES`: Compart DocBridge Notes is required
- `UXT`: user exit
- `BATCH`: Directory containing raster images converted to Compart Batch Format (XML, scan batches)

5.2.2.1. cpmill NULL Driver

With the introduction of JavaScript, users came up with new use cases for cpmill that did not quite fit into the original approach of always having a set of input files to be converted into one or more output files. All the original drivers (121, N21, CLU, etc.) follow that approach and they all require at least one input file to be present to be executed.

The NULL driver provides a simple way to execute JavaScript in cpmill without having to provide any input files. It also does not produce any output files (unless you create them yourself). So, you do not have to use the workaround of creating “files” with the MFF filter NUL.

```
<driver type="NULL">
```

The following attributes of the parameter `<parameter>` are ignored:

- `in.search.mask`
- `in.process.format.filetype`
- `out.process.format.filetype`
- `out.put.dir`

All the other parameters work as before, although some of them may not make a lot of sense, e.g. `general.inputfiles.keepopenstrategy`, see *chapter 5.2.1. cpmill Profile Element <parameter> on page 211.*

The NULL driver only executes the following events:

- `begin`
- `end`

5.3. MFF Filter Profiles

The MFF input and output filter are part of the MFF architecture and are used to read and write files in different formats. They can be customized be means of profiles in XML format to the environment in which the application is running.

In the profiles can be specified, where certain resources like fonts should be searched, which code page should be used, or the assignment of font names to font files. Although the profiles of different MFF filters are similar in some aspects, they are described separately to allow a better overview about all relevant sections. None of the MFF filters requires a profile; profiles are only used when the conversion results shall be improved.

If there is nothing else specified in the profile during reading of a page with an MFF filter, the font name is written into the Presentation Area, as it is specified in the input file. During writing a page, a certain font in the Presentation Area can be assigned to any font. If e.g. during reading of an AFP file the font *C0OCB11* is referenced, this font can be replaced by the font *OCR-B* for the output in PDF.

In the most cases, the automatic font matching of the writing MFF filter is sufficient to get practical conversion results. For that, the writing filter searches a suitable font corresponding to the font attributes *Serif/Sans Serif, Proportional/Monospaced*, and *Italic/not Italic*.

5.3.1. MFFAFP

5.3.1.1. Global Settings

In the following, global settings are described, which overwrite the predefined filter settings.

Example: Global settings in the MFFAFP profile

```
<globals>
  <defaultchar            unc="003F"    />
  <multidocument          value="true"  />
  <generateoverlays       value="false" />
  <autoformdefgeneration  value="false" />
  <treatrectstransparent  value="false" />
  <rasterunevenpointsizes value="false" />
  <rastersettype          value="screen"/>
  <overlayasreference     value="false" />
  <pagesegmentasreference value="false" />
  <mappagesegments        value="false" />
  <optimizepa             value="always"/>
  <aliasmap  devicename="SD150300" map="SD150000" condition="ifnotfoundinencodings"/>
             <!-- U+02C6: Circumflex Accent -->
  <aliasmap devicename="SD190300"  map="SD190000" condition="ifnotfoundinencodings"/>
             <!-- U+007E: Tilde -->
  <aliasmap  devicename="SD310000" map="SM150000" condition="ifnotfoundinencodings"/>
             <!-- U+00AF: Macron -->
  <aliasmap  devicename="GF010002" map="GF010000" condition="ifnotfoundinencodings"/>
             <!-- U+03D5: Phi Small (open form) -->
  <aliasmap  devicename="SS680000" map="SP100000" condition="ifnotfoundinencodings"/>
             <!-- U+2013: En Dash -->
  <aliasmap  devicename="ND051001" map="ND051000" condition="ifnotfoundinencodings"/>
             <!-- U+2075: Five Superscript -->
```

`<defaultchar>`

Element specifies the character, which is written, if the appropriate character is not available, i.e. if it was not found in a code page. The value of the attribute `unc` is a hexadecimal number that represents the character.

`<multidocument>`

Element specifies, whether only one or several documents, which are generated in one file, should be read, i.e. it applies to files, in which many AFP overlays or page segments are stored. The values of the attribute `value` are `true` or `false`.

`<generateoverlays>`

While writing is executed, element controls whether references to “external pages” in form of an AFP overlay will be generated in the output file (inline resource). These “external pages” can be AFP overlays that were not resolved during reading (see also element `<overlayasreference>`), or overlays (“stamp”) with other formats that were contributed during the conversion. The values of the attribute `value` are `true` or `false`.

`<autoformdefgeneration>`

Element controls the automatic generation of AFP copy groups (= Medium Maps) with data like tray, number of copies, and simplex/duplex. This information can come from the reading filter (e.g. from PCL) during conversion or can be set by the user.

For any new arising combination of the information mentioned above a new copy group will be created. All automatically generated copy groups will be numbered and stored in the formdef "F1CPAUTO". As in the case of automatic overlay generation, inline resources will be generated. The values of the attribute `value` are `true` or `false`.

`<treatrectstransparent>`

Element controls AFP generation, especially when converting PDF files. During tables generation in PDF files, the following elements are generated: At first (mostly black) lines and subsequently table cells with non-transparent rectangular areas (sometimes with filling patterns) that overlap the lines partially. Normally this causes the rectangular areas to be automatically combined with the lines to a bitmap. The AFP architecture pixels can only be black and never white (the area overlapping the line must not be black, but must have the filling pattern of the cell).

Since the output file will increase because of the conversion into an image and it will not have a better quality, it is possible to specify that no rasterization process will be executed. The values of the attribute `value` are `true` or `false`.

`<treatpathstransparent>`

Element controls whether paths are treated as transparent paths. The values of the attribute `value` are `true` or `false` (default). For more information, see also the description of element `<treatrectstransparent>`.

`<rasterunevenpointsizes>`

Note: In documents with formats that support vector fonts, e.g. TrueType or Type 1 (PDF or PCL) the following problem comes up when converting to a format that does not support vector font like AFP: Non-integer font sizes (e.g. 10.25) cannot be reproduced exactly.

To eliminate differences of visual integrity, element controls converting text with non-integer font sizes into bitmaps. **Note:** Visual integrity improves at the expense of the file size. The values of the attribute `value` are `true` or `false`.

`<rastersettype>`

If areas are written with filling patterns, different types of patterns are available. Element is used to provide a specified area with a defined pattern type. The values of the attribute `value` are `screen`, `screenmodified`, `afp`, and `afpmodified`.

`<overlayasreference>` / **`<pagesegmentasreference>`**

Note: For the reading process of an AFP file, the user can be faced with the following question: Should referenced overlays be maintained as they are or should the information from the overlays be resolved into the page? There are several options to control this behavior during reading as well as during writing into AFP.

During reading process element controls whether overlays that are referenced on an AFP page are placed directly in the page or whether the references to the overlays are treated as they are. If the references stay unchanged, the content can be written once (although several references do exist) when writing in appropriate output formats. Currently, AFP filter and PDF filter are able to generate information from overlays only once. The values of the attribute `value` are `true` or `false`.

<mappagesements>

Element controls, whether Map Page Segment structured field (MPS) is processed. The values of the attribute `value` are `true` or `false`. If `true` is specified, page segments are loaded into the printer and stored until the print job is completed (hard object). If `false` is specified, page segments are sent to the printer directly as often as requested but not stored (soft object).

<optimizepa>

Element controls the generation of the page before it is written to AFP. If necessary, items are removed or rasterized to prevent transperancy problems. The values of the attribute `value` are `ifdifferenttypes`, `always` or `never`. If the input filter is not the AFP filter, `ifdifferenttypes` must be specified. Then, the page is generated for AFP output.

<aliasmap>

Element controls whether a character described by a character identifier is replaced by another character. If a character is not found in a specified code page during writing, a defined replacement character is used, see also *chapter 5.3.1.4. Code Pages on page 227*.

The values of the attribute `condition` are `ifnotfoundinencodings` and `always`. The values specified for the attributes `devicename` and `map` is an IBM graphic character identifier.

5.3.1.2. User Defined Encodings

Normally in the text of AFP files, a byte represents a particular character. However to be able to display more than 256 characters, each text section is assigned to a certain code page. This code page assigns each byte in the text (code point) to a character. There is a set of default character names, but of course, they do not cover all characters. For example, not all IBM specified names are assigned to a Unicode character and vice versa not all Unicode characters have a representation with the default character names. In addition, many AFP users create their own characters. Because the MFF architecture stores text always in Unicode and only certain concatenations between IBM name and Unicode are “hard-wired” built in, any name, defined by attribute `devname`, can be assigned to the attribute `unc` in the profile.

Example: User defined encodings in the MFFAFP profile

```
<encodinglist>
  <encoding name="T1ANSI">
    <entry devname="TRANS032" unc="0020"/> <!-- space               -->
    <entry devname="TRANS033" unc="270F"/> <!-- pencil              -->
    <entry devname="TRANS034" unc="2702"/> <!-- black scissors      -->
    <entry devname="TRANS035" unc="2701"/> <!-- upper blade scissors -->
    <entry devname="TRANS040" unc="260E"/> <!-- black telephone     -->
    <!-- [...]                          -->
  </encoding>
  <encoding name = "elixirsymbol">
    <entry devname="XC065410"  unc="F041"/>  <!—s of a saving bank-->
  </encoding>
<encodinglist>
```

5.3.1.3. Fonts

Normally, AFP fonts are available as raster fonts. You must specified which font file should be used for which font size.

Example: Font definition in the MFFAFP profile

```
<fontlist>
   <font family="Arial" serifstyle="SANSSERIF" spacing="PROPORTIONAL" >
     <face   weight="MEDIUM"
             width="NORMAL"
             style="UPRIGHT"
             baselineincrement="1150">
       <raster devname="C0A48410" size= "1" />
       <raster devname="C0A48420" size= "2" />
       <raster devname="C0A48430" size= "3" />
       <raster devname="C0A48440" size= "4" />
       <raster devname="C0A48450" size= "5" />
     </face>
     <face   weight="BOLD"
             width="NORMAL"
             style="UPRIGHT"
             baselineincrement="1150">
       <raster devname="C0A58510" size= "1" />
       <raster devname="C0A58520" size= "2" />
       <raster devname="C0A58530" size= "3" />
       <raster devname="C0A58540" size= "4" />
       <raster devname="C0A58550" size= "5" />
     </face>
   </font>
</fontlist>
```

5.3.1.4. Code Pages

Example: Code page definition in the MFFAFP profile

```
<codepagelist>
   <codepage name="T1V10273" iana="IBM273" default="yes"/>
   <codepage name="T1V10500" iana="IBM500" />
</codepagelist>
```

`<codepagelist>`

The child element `<codepage>` of `<codepagelist>` defines several code pages. One of them can be assigned by the attribute `default` with the value `yes` as the default code page. When writing to AFP the code page will be switched automatically, as long as characters are not found in the currently used code page.

5.3.1.5. Resources

AFP resources can either be bundled in one file (as resource library) or exist as single resources.

Example: Resource definition in the MFFAFP profile

```
<resourcelist>
  <library filename="resources1" />
  <library tle="reslib"          />
  <files path=\\fileserv\drive\resoure\reslibs type="resourcelib"    extension="" />
  <files path="\\fileserv\drive\resourc\codedfonts" type="codedfont" extension="300" />
  <files path="\\fileserv\drive\resourc\charsets"   type="charset"   extension="" />
  <files path=\\fileserv\drive\resourc                              extension="" />
</resourcelist>
```

<resourcelist>

Within the element all resource definitions are specified. To find resource libraries only file name and path must be known. The file name can be fully qualified,
e.g. `<library filename="/home/user/afpres1.lib"/>`
or without path which can be specified for several resource libraries, e.g.
`<files path="/home/user" type="resourcelib" extension="lib"/>`.
It is also possible to pick up a name of a Reslib from a TLE during reading an AFP file, e.g.
`<library tle="reslib"/>`.

To specify the location, which should be scanned for AFP resources, it is sufficient to specify the directory, in which the resources are `<files path="/usr/local/afpres"/>`.

Additionally by means of the attribute `type`, it can be specified that in the defined directory a scan process shall be executed for resources. The attribute `extension` limits the search to files with a special extension. For the attribute `type` the values `overlay`, `pagesegment`, `codedfont`, `codepage`, `charset`, `formdef`, `resourcelib`, and `pagedef` can be specified.

5.3.1.6. Trays

Input trays are assigned to the input and output format via unique tray names. In AFP, tray names are specified inside a copy group via a number (device-ID). Because the numbers specified in different data streams are not related to each other, these numbers have to be assigned to a tray name on the input side; on the output side they again will be reconverted to a number. Example: If in a PCL input data stream a tray is assigned to number 3, this tray number can be assigned to the name “PAGE1” (in the profile of the PCL module). For the output in AFP this name will be assigned again to the tray number 0. The release of a tray number in AFP can only be arranged, if the element `<autoformdefgeneration>` is specified. In this case, a copy group is created for the required tray number (as well as all other specifications like simplex/duplex, etc.).

Example: Tray definition in the MFFAFP profile

```
<traylist>
<!-- = inputtray    ========================== -->
  <inputtray deviceid='1' name="UpperTray"       />
  <inputtray deviceid='2' name="ManualFeed"      />
  <inputtray deviceid='3' name="ManualEnvelope"  />
  <inputtray deviceid='4' name="LowerTray"       />
  <inputtray deviceid='5' name="LargeCapacity"   />
  <inputtray deviceid='6' name="Envelope"        />
  <inputtray deviceid='7' name="Automatic"       />
</traylist>
```

<inputtray>

Element specifies with the attributes `deviceid` (tray number) and `name` (tray name) the tray definition for input and output.

5.3.1.7. Color Assignment

Any RGB color can be assigned to an OCA color index (AFP input).

Example: Color Index List in the MFFAFP profile

```
<colorindexlist>
   <!--                          a r g b                          -->
   <colorindex index='0'  color="#00000000" /> <!-- DEVICE DEFAULT -->
   <colorindex index='1'  color="#000000FF" /> <!-- BLUE           -->
   <colorindex index='2'  color="#00FF0000" /> <!-- RED            -->
   <colorindex index='3'  color="#00FF00FF" /> <!-- MAGENTA        -->
   <colorindex index='4'  color="#0000F800" /> <!-- GREEN          -->
   <colorindex index='5'  color="#0000B0B0" /> <!-- CYAN           -->
   <colorindex index='6'  color="#00FFFF00" /> <!-- YELLOW         -->
   <colorindex index='7'  color="#00FFFFFF" /> <!-- WHITE          -->
   <colorindex index='8'  color="#00000000" /> <!-- BLACK          -->
     .
     .
     .
   <colorindex index='16' color="#00800000" /> <!-- BROWN          -->
</colorindexlist>
```

`<colorindex>`

The child element `<colorindex>` of `<colorindexlist>` with its attributes `index` and `color` is used to assign a RGB color to OCA color index (AFP).

Example: Highlight Color (Spot Color) List in the MFFAFP Profile

```
<highlightcolorlist>
   <highlightcolor name='default' type='oca' number='0x0000'
                        coverage='0'  shading='0'/> <!-- default color -->
   <highlightcolor name='color 1' type='oca' number='0x0001'
                        coverage='10' shading='11'/> <!—- 1st hilight color -->
   <highlightcolor name='color X' type='oca' number='0x0002'/>
</highlightcolorlist>
```

`<highlightcolorlist>`

The child element `<highlightcolor>` of `<highlightcolorlist>` with its attributes `name`, `type`, `number`, `coverage`, and `shading` defines a output highlight color.

If the attribute `name` is specified with the value `default`, the default color is defined. Any additional color can specified with a user-defined name. The values of the attribute `type` are `oca` and `highlightcolor`.

If the value `oca` is specified, the value of the attribute `number` specifies a defined set (PT1 Subset) of colors. The value of the default printer color is `X'0000'`.

Note: The attributes `coverage` and `shading` are not supported in combination with the value `oca` of the attribute `type`.

For information about the set of colors, see profile file mffafp.pro and *IBM Data Stream and Object Architectures Presentation Text Object Content Architecture Reference*.

If the value `highlightcolor` is specified for the attribute type, the value of the attribute number defines a printer dependent highlight color. The value of the default printer color is `X'0000'`.

If the attribute `coverage` is specified, a percentage value (0 - 100) defines the color coverage. If the attribute `shading` is specified, a percentage value (0 - 100) defines the color shading. The sum of both values must be less or equal 100.

5.3.1.8. Input AFP Files Settings

Note: The chapter provides information for “advanced” user.

The following elements can be used in the element `<input>`.

`<defaultformdef>`

Element specifies with the value of the attribute `name` the formdef name, if it is not passed by the application or unknown.

`<usefontspacecharacter>`

Element controls the use of the space character increment. The values of the attribute `value` are `false` and `true`. If `false` is specified, the space character increment parameter is used. If `true` is specified the font character metrics information of a character is used, see also *IBM Font Object Architecture (FOCA) Reference*.

`<usegridwidth>`

Element defines whether the width value specified in the global resource identifier (GRID) is used. The values of the attribute `value` are `false` and `true`. If `true` is specified, the GRID defined width is used. If `false` is specified, the width value is taken from a built-in table.

`<inputinterpretationmode>`

Explanation: (1) Logical page. It is a presentation space that consists of one or more object areas. The logical page has specifiable characteristics such as size, shape (rectangle), orientation, and offset. (2) Medium. It is a two-dimensional space with a base coordinate system from which all other coordinate systems are either directly or indirectly derived. It can be mapped on a physical medium such as paper depending on the hardware, e.g. printer.

Element controls the page size and the position of the page. The values of the attribute `value` are `page`, `pagewithoffset`, and `medium`.

If you specify the default value `page`, the resulting page size and the position are defined by the page.

If you specify the value `pagewithoffset`, the page to medium offset is taken into account.

If you specify the value `medium`, the size and the position are defined in consideration of the specified medium formdef. If can be used for n-up printing. The page to medium offset is taken into account.

`<readarchivefields>`

Element controls whether archive fields are read. The values of the attribute `value` are `true` and `false`. The default value is `true`, i.e. archive fields (TLEs) are read. If `false` is specified, archive fields are ignored.

<trimarchivefields>
Element controls trailing spaces. A TLE consists of a key and a value. The value can be provided with trailing spaces. The values of the attribute `value` are `true` und `false`. The default value is `false`, i.e. TLE values are unchanged. If you specify the value `true`, trailing spaces are truncated.

<readcomments>
Element `controls` whether information in NOPs is read as comments. **Note:** NOP is a container for meta data. The values of the attribute `value` are `true` and `false`. The default value is `true`, i.e. data content of NOPs is read. If `false` is specified, content reading is ignored.

<addcommentstopage>
Element controls whether NOP `data` content is imported. The values of the attribute `value` are `true` and `false`. The default value is `false`, i.e. import of data content is ignored. If `true` is specified, data content is imported. **Note:** Element is used is such a way as to import and provide NOP data content when non-binary files are copied.

5.3.1.9. Output AFP Files Settings

Note: The chapter provides information for “advanced” users.

The following elements can be used in the element `<output>`.

<measurementunits>
Element defines with the value of the attribute `value` a printer specific resolution, e.g. `240,1440`. One or more values can be specified. The values are separated with a comma. **Note:** The resolution depends on the conversion software (CpCOLD, cpmcopy) and the input data format.

<goca>
Element controls whether the output is generated with GOCA (Graphics Object Content Architecture). The values of the attribute value are false and true. The default value of the attribute value is false, i.e. GOCA generation is not performed.

If the value of the attribute value is true, preselected objects are written as GOCA objects. Objects are circles, segments of circles, lines, rectangles, and text. For further definitions and settings, see also CPAIR.H file.

<colorspaces>
Element controls how color is applied to GOCA objects. The values of the attribute `type` (pattern) are `FILL` and `STROKE`. The values of the attribute `value` (color model) are `CIELAB`, `CMYK`, `HIGHLIGHT`, `OCA`, `OCAEXT`, and `RGB`. One, several or all models can be specified. If a model is selected as ‘best case’ but not allowed to be used, an error message is issued.

<positioningmethod>
Element controls the positioning method for GOCA characters. The values of the attribute `value` are `font` and `cell`. If `font` is specified, the character cell is ignored and the size of the font is used to define the size of the character. If `cell` is specified, the size of the character cell defines the size of the character, see also *IBM Graphics Object Content Architecture (GOCA) Reference*.

<writecrlfafterstructuredfield>

Element controls whether each structured field is followed by a CRLF. The values of the attribute `value` are `false` and `true`.

<avoidcrlfinsidestructuredfields>

Element has the attribute `value` which can be `false` (Standard) and `true`. The element prevents byte sequences of `X'0D'` and `X'0A'` to be written inside of structured fields in AFP output. The major way of avoiding the byte sequences `X'0D'` and `X'0A'` is to split structured fields containing these byte sequences between `X'0D'` and `X'0A'`. Normally structured fields are delimited by the byte value `X'5A'`. However, some software products delimit structured fields both with `X'5A'` and with byte sequences of `X'0D'` and `X'0A'` in writing mode and expect AFP datastreams to be delimited with `X'0A'` and `X'0D'` in reading mode, too. These software products cannot read AFP data streams with `X'0A'` and `X'0D'` byte sequences within structured fields. Activation of this element ensures that AFP output is generated which can be processed by software products that expect `X'0A'` and `X'0D'` structured field delimiters.

<usehighlightcolor>

Element controls the processing of the elements `<highlightcolorlist>` and `<highlightcolor>`. The values of the attribute `value` are `false` (default) and `true`. If you specify the value `false` for the attribute `value`, the processing is not enabled. If you specify the value `true` for the attribute `value`, the processing of the elements is enabled.

<ignoreclipping>

Element controls the processing of the clipping rect(angle)s that may be part of a PDF input file. The values of the attribute `value` are `false` (default) and `true`. If you specify the value `false` for the attribute `value`, clipping rects are ignored. The accurate conversion of the PDF input file contains rasterized characters. Therefore, you may note a reduced processing speed, increased file and bad presentation results. If you specify the value `true` for the attribute `value`, ignoring of clipping rects is not activated.

<usetrayasmediummap>

Element controls the use of tray names. The values of the attribute `value` are `false` and `true`. If `true` is specified, the application defined tray name is converted to a reference to a copy group. The IMM (Invoke Medium Map) structured field is generated. If `false` is specified, the tray is used for formdef generation.

<writeorder>

Element defines the writing order of `TLE`s, `NOP`s, and `IMM`s in the attribute `value`.

<usegridwidth>

Element defines whether the width value specified in the global resource identifier (GRID) is used. The values of the attribute `value` are `false` and `true`. If `true` is specified, the GRID defined width is used. If `true` is specified, the width value is taken form a built-in table.

<maxrecordsize>

Element defines the maximum AFP record size. The default value for the attribute `value` is `0`, i.e. 32 KB.

<writeonlyasolutemoves>
Element defines whether absolute positioning commands are used exclusively or relative positioning commands are used, as well. The values of the attribute `value` are `false` for absolute and for relative positioning and `true` for absolute positioning only.

<characteraccuracy>
Element controls that while output text is written the character spacing of the input font is compared with spacing of the output font. If the spacing is different to the specified value of the attribute value, subsequent text is repositioned. If the value `0` is specified, default values of the driver are used.

<combinexaccuracy>
Modifications of output font insert positioning commands into the data stream, see also element `<characteraccuracy>`. Depending on the application element controls the suppression of negligible positioning commands. The value of the attribute `value` is the threshold value for the horizontal (x) direction (character spacing).

<combineyaccuracy>
Modifications of output font insert positioning commands into the data stream, see also element `<characteraccuracy>`. Depending on the application element controls the suppression of negligible positioning commands. The value of the attribute `value` is the threshold value for the horizontal (y) direction (baseline).

<suppressaeginpagesegment>
Element controls the activation of Active Environment Group (AEG) for page segments. The values of the attribute `value` are `false` (default) and `true`. If you specify the value `false`, the AEG is enabled. If you specify the value `true`, the AEG is suppressed.

<jpegquality>
Element controls visual integrity. The value of the attribute `value` contains a percentage number. The default value is `75`. **Note:** Visual integrity improves at the expense of the file size.

<monochromecompression>
With the attribute `value` element specifies the compression method for bi-level (two color) images. The values of the attribute are `MMR` (CCITT Modified Modified Read – IBM MMR) and `FAXG4` (CCITT T.6 Facsimile Coding – G4MMR).

<fonthandling>
Element controls the font usage. The values of the attribute `default` are `auto` and `converttoafp`. If you specify `auto` for the attribute `default`, an automated and optimized conversion is executed based on the available fonts. If you specify the value `converttoafp` for the attribute `value`, the fonts are generated automatically.

Example: Gamma value/Tone correction list in the MFFAFP profile

```
<gammalist>
    <gammavalue from=  '0' to=  '0'/>
    <gammavalue from=  '7' to= '48'/>
    <gammavalue from= '15' to= '72'/>
    <gammavalue from= '23' to= '88'/>
      .
      .
      .
    <gammavalue from='199' to='224'/>
    <gammavalue from='207' to='232'/>
    <gammavalue from='215' to='232'/>
    <gammavalue from='223' to='240'/>
    <gammavalue from='231' to='240'/>
    <gammavalue from='239' to='248'/>
    <gammavalue from='247' to='248'/>
    <gammavalue from='255' to='255'/>
 </gammalist>
```

`<gammalist/>`

Element controls the gamma value correction/tone correction. The element `<gammavalue>` defined as child element of `<gammalist>` contains the tone correction values for rasterized areas. The attribute `from` specifies the input value and the attribute `to` the output value.

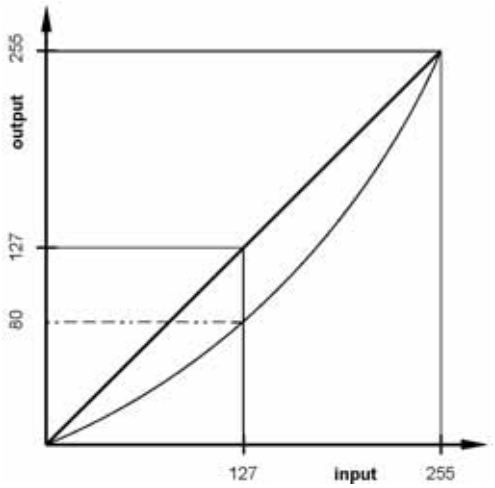


Figure 6: Sample Chart – Tone correction in MFFAFP Profile

5.3.2. MFFGOF

The filter MFFGOF is only designed as input filter and is reading two SAP formats, which are combined in the name *Generic Output Format (GOF)*:

- Output Text Format (OTF)
- ABAP List Format (ALF)

The GOF input filter identifies the format by the `FORMAT` parameter of the data stream header: OTF, if `*FORMAT=OTF`, and ALF, if `*FORMAT=LIST`.

OTF is a page-describing format. Therefore, it does not need any additional layout rules, the settings of the GOF profile do not influence the processing of OTF data streams. During reading and converting into the internal meta format, they are ignored. The settings of the MFFGOF profile are only effective to ALF lists and influence exclusively the presentation of their layout. An exception is the element `<stringappendmode>`.

`<stringappendmode>`
: Element affects the method how two strings in sequential order are displayed, if the second string is not repositioned.

  In the OTF data stream, two positioning commands are used: MT (Move-To) and ST (STring). The values of the attribute `value` are `absolute` and `relative`. If `absolute` is specified, the positioning of the second string is absolute, i.e. the position of the second string is calculated from the position and the length of the first string. If relative is specified, the positioning is relative to the second string, i.e. it is appended.

Compart recommends the relative positioning. In the output file formatted text strings can have a different length because of the output format fonts. Thus, if absolute positioning is used, text strings may overlay each other.

5.3.2.1. User Defined Encodings

For detailed list of encodings, see MFFGOF profile file and *chapter 5.3.1.2. User Defined Encodings on page 226*.

5.3.2.2. Medium Definitions

The layout presentation is specified in the ALF data stream by the parameters `*PJFORM` or `*PJPAPER` in the header depending of the ALF version. The keywords assigned to these parameters are of descriptive nature like *X_PAPER*, *Z_PA01* or *LETTER* or they contain a line and column assignment in its keywords like *Z_65_132*, i.e. 65 lines and 132 columns.

The GOF filter reads this header entry and looks in the profile file mffgof.pro for an entry in the element `<mediumdefinition>` with the same value of the attribute `name`.

Example: Medium Definition in the MFFGOF profile

```
<mediumdefinition
   name="X_65_132"
   formdef="F1A10111"
   pagedef="A4_portrait"
   pagedefs="A4_portrait, A4_landscape"
   copygroup="1"
   fontnormid="12" />
```

The medium definition uses the following attributes according to specifications of an AFP data stream: `copygroup`, `formdef`, `fontnormid`, `name`, `pagedef` und `pagedefs`.

If you use the attribute `pagedefs`, you can specify two page definitions (pagedef). The first pagedef is used, if 'P' (portrait) is specified in the OP command 'P', the second pagedef is used, if 'L' (Landscape) is specified. If the attribute `pagedefs` is not specifie, the attribute `pagedef` is default.

If the MFFGOF filter does not find a medium definition with the name that is specified the header, it tries to read a name form the line and column definition and tries to find a corresponding page definition in the GOF profile, see *chapter 5.3.2.4. Page Definitions on page 237*. If it finds an adequate definition, it uses the parameter values and Courier font with 10pt size. Otherwise, an error message is issued.

5.3.2.3. Font Assignments

Example: Font definitions in the MFFGOF profile

```
<fontlist>
  <font family="COURIER" serifstyle="SERIF" spacing="MONOSPACED" id="1"  pitch="12000">
    <face weight="MEDIUM" width="NORMAL" style="UPRIGHT" size="10" />
  </font>
  <font family="COURIER" serifstyle="SERIF" spacing="MONOSPACED" id="12" pitch="15000">
    <face weight="MEDIUM" width="NORMAL" style="UPRIGHT" size="8" />
  </font>
  <font family="COURIER" serifstyle="SERIF" spacing="MONOSPACED" id="13" pitch="12000">
    <face weight="BOLD" width="NORMAL" style="UPRIGHT" size="10" />
  </font>
</fontlist>
```

`<fontlist>`

Element defines fonts in a list. The element `<font>` specifies which fonts are used by the GOF filter. The assignment is based on the value of the attribute `fontnormid` of the related element `<mediumdefinition>` that has to match the value of the attribute `id` of an element `<font>` inside the font list. If no corresponding `id` value is found, the font with `id` value 1 inside the font list is used. The font is defined by the element `<font>` in combination with the element `<face>` and its attributes.

5.3.2.4. Page Definitions

The page definitions in the `<input>` element of the GOF profile are specified by the elements:

`<pagesize>`,
`<margins>` and, where applicable
`<linesperinch>`,
`<resolution>`, and
`<rotation>`.

Chapter Example: Page definitions in the MFFGOF profile on page 237 lists a set of page definitions that can be specified in the MFFGOF profile.

Example: Page definitions in the MFFGOF profile

```
<pagedefinition name="A4_portrait">
  <papersize format = "A4" orientation = "portrait" />
  <margins left = "2 cm" right = "2 cm" top = "1 cm" bottom = "1 cm" />
  <linesperinch value  = "8" />
  <rotation value = "0" />
</pagedefinition>

<pagedefinition name="A4_landscape">
  <papersize format = "A4" orientation = "landscape" />
  <margins left = "2 cm" right = "2 cm" top = "1 cm" bottom = "1 cm" />
  <linesperinch value = "8" />
  <rotation value = "0" />
</pagedefinition>

<pagedefinition name="LETTER_portrait">
  <papersize format = "letter" orientation = "landscape" />
  <margins left = "1 in" right = "1 in" top = "2 in" bottom = "2 in" />
  <linesperinch value = "8" />
  <rotation value = "0" />
</pagedefinition>

<pagedefinition name="LETTER_landscape">
  <papersize format = "letter" orientation = "landscape" />
  <margins left = "2 cm" right = "2 cm" top = "1 cm" bottom "1 cm" />
  <linesperinch value = "8" />
  <rotation value = "0" />
</pagedefinition>

<pagedefinition name="mypage">
  <papersize width = "297 mm" height = "21 cm" />
  <margins left = "2 cm" top = "15 mm" />
  <linesperinch value = "8" />
  <resolution rx="1440" ry="1440" />
  <rotation value = "270" />
</pagedefinition>
```

`<papersize>`

Element defines the paper size by the attributes `format` and `orientation`, as shown in the upper definitions of this example, or by the attributes `width` and `height`, as shown in the last definition of the example. As value for the attribute `format` all names of page formats can be used that are listed in *Appendix C: Paper Formats on page 280*. The values for the attribute `orientation` are `landscape` and `portrait`. For the values of the attributes `width` and `height`, unit abbreviations are listed in *Appendix E: Units of Measurements on page 285*.

<margins>
Element defines page margins. The following attributes can be specified: right margin (attribute `right`), left margin (attribute `left`), top margin (attribute `top`), and bottom margin (attribute `bottom`). The same unit abbreviations as for the attributes `width` and `height` in the element `<papersize>` can be used for the margin attributes.

<linesperinch>
Element defines in the attribute `value` the number of lines per inch.

<resolution>
Element defines the resolution. The resolution in x-direction is defined by the value in the attribute `rx`, the resolution in y-direction is defined by the value in the attribute `ry`. Default unit is of the attribute value is a Twentieth of Inch Point, i.e. 1/1440 inch.

<rotation>
Element specifies the rotation of a document page. Valid values for the attribute `value` are `0`, `90`, `180`, and `270` specifying the orientation.

If a medium definition for the data stream was found, the GOF filter identifies the page definition to be used by means of the `name` attribute in the element `<pagedefinition>` that has to correspond to the value of the `pagedef` attribute in the element `<mediumdefinition>`.

5.3.2.5. Form Definitions

The form definition (attribute `formdef`) in the `<mediumdefinition>` element allows the use of formdefs like in the case of AFP. With formdefs forms or overlays can be defined for ALF lists. The value of the attribute `formdef` specifies the name of a formdef object. This object has to be accessible directly for the filter. The drive and path of the object will be read by the filter in the resource list of the GOF profile – as long as it is not specified fully qualified (see the following subchapter).

5.3.2.6. Copy Group Definitions

The formdef consists of one or more copy groups (medium maps). Each of these copy groups can contain definitions for overlays (also different overlays for different pages), tray assignment, and simplex/duplex printing. The value of the attribute `copygroup` specifies which of the copy groups inside the formdef shall be used for the ALF list. Currently, the GOF filter does not interpret this value: It uses the first copy group anyway.

5.3.2.7. Page Orientation

<autorotate>

Element controls page orientation from landscape to portrait. **Note:** <autorotate> is used, if no tray name is specified as medium map name in elements like <pagedefinitions>, <mediumdefintions>, with the SAPGOF command OP (OPEN PAGE), and if the printer cannot print landscape jobs.

Document pages are generated with landscape orientation, i.e. in the data stream pages have the landscape orientation. Using <autorotate> pages can be rotated to portrait on above mentioned conditions, i.e. they are rotated 90 degrees counterclockwise.

The values of the attribute value are false and true. If the value false is specified, the page rotation is not active. If the value true is specified, pages are rotated.

5.3.2.8. Resource Definitions

For more details, see *chapter 5.3.1.5. Resources on page 227.*

5.3.2.9. Trays

<tray>

Element controls the definition of trays. The value of the attribute name is the name of the paper tray. The name must correspond with the value that is specified with the attribute name of the child element <inputtray> of the element <inputtray>. For more information, see *chapter 5.3.1.6. Trays on page 228.*

5.3.2.10. Barcode Definitions

For each barcode name used in OTF a corresponding barcode type can be specified as well as the fact whether a check digit has to be generated or not. Furthermore it can be specified whether the content of the barcode has to be written as text under the symbol and the font to be used for it (the so-called „Human Readable Interpretation“, HRI). Additionally the width can be defined that has to be used for the symbol depending of the information contained in the barcode, as well as the decision whether the maximum width predetermined in the OTF file has to be considered or not.

Example: Barcode definition in the MFFGOF profile

```
<barcodelist>
  <barcode name="BC_CD39C" type="Code 39"
                           checkdigits='1'
                           displayhri="FALSE"
                           modulewidth="18 tw"
                           respectmaxwidth="FALSE"
                           fontid='13' />
  <barcode name="ARTNR"    type="Code 128"
                           checkdigits='0'
                           displayhri="TRUE"
                           modulewidth="20 tw"
                           respectmaxwidth="FALSE"
                           fontid='13' />
  <barcode name="MBBARC1"  type="EAN 8"
                           checkdigits='0'
                           displayhri="TRUE"
                           modulewidth="20 tw"
                           respectmaxwidth="FALSE"
                           fontid='14' />
</barcodelist>
```

<barcode>

Within the element <barcodelist> the element defines the barcodes that are to be used. The barcode name has to match with the barcode name in the OTF file (attribute name). The barcode type (attribute type) can be one of the types listed in *Appendix D: Barcodes on page 284*. The number of the check digits to be generated is identified by the attribute checkdigits.

The attribute displayhri specifies whether a text with the content of the barcode shall be generated in addition to the barcode symbol.

The width of the barcode symbol can be controlled by the attribute modulewidth. It specifies how wide a module (= a thin bar) has to be presented. With this attribute, the same units can be used as in the page size attributes.

The attribute respectmaxwidth specifies whether the value of the maximum barcode width specified in the OTF file has to be used or not.

The attribute fontid specifies the font to be used in the case displayhri is set to TRUE, see also element <fontlist>.

5.3.2.11. Colors

Example: Highlight Color (Spot Color) List in the MFFGOF Profile

```
<highlightcolorlist>
   <highlightcolor name="COL0N" foreground-rgb="#00000000"
                                background-rgb="#00808080" />
   <highlightcolor name="COL0H" foreground-rgb="#000000FF"
                                background-rgb="#00808080" />
   <highlightcolor name="COL0V" foreground-rgb="#00808080"
                                background-rgb="#000000FF" />
   <highlightcolor name="COL1N" foreground-rgb="#00000000"
                                background-rgb="#008585A6" />
   <highlightcolor name="COL1H" foreground-rgb="#00000000"
                                background-rgb="#000000FF" />
   <highlightcolor name="COL1V" foreground-rgb="#000000FF"
                                background-rgb="#00808080" />
      .
      .
      .
   <highlightcolor name="COL7N" foreground-rgb="#00000000"
                                background-rgb="#00A185A6" />
   <highlightcolor name="COL7H" foreground-rgb="#00000000"
                                background-rgb="#00E06090" />
   <highlightcolor name="COL7V" foreground-rgb="#00E06090"
                                background-rgb="#00808080" />
```

`<highlightcolorlist>`

The child element `<highlightcolor>` of `<highlightcolorlist>` with its attributes `name`, `foreground-rgb`, and `background-rgb` defines a highlight color. The value of the attribute name specifies a predefined SAPGOF color. For a set of colors, see profile file *MFFGOF.PRO*. **Note:** The SAPGOF command PC (PRINTCONTROL) and the name of printcontrol, e.g. `PCCOL0N` for a color, defines that a color is assigned to the data following the command.

If the attribute `foreground-rgb` is specified, the foreground color, i.e. line or font color, is defined. The default font color is black.

If the attribute `background-rgb` is specified, the background color is defined. If no color list is defined in the profile, the default is black and white.

5.3.2.12. Printing

`<duplexcontrol>`

Element controls simplex and duplex printing. The values of the attribute `value` are `simplex`, `duplexfront`, and `duplexback`.

If the attribute `simplex` is specified, data is printed on one side of a page. If the attribute `duplexfront` is specified, printing starts on the front side of a page and is continued on the backside.

If the attribute `duplexback` is specified, printing starts on the backside of a page.

5.3.2.13. Conversion of OTF to AFP with cpmcopy

If the attribute `PJFORM` in the header of the OTF file is specified, the name of a formdef is found in the medium definition, see also *chapter 5.3.2.2. Medium Definitions on page 235* and *chapter 5.3.2.5. Form Definitions on page 238*. The command line option `-selfcontained` must be specified in order to include the formdef with the specified name in the AFP output profile.

In the MFFAFP profile file the attribute value of `<usetrayasmediummap>` must be `true` to read the name of the copy group that is to be referenced from the tray name of the OP (Open Page) command in the OTF file. If the tray name is omitted, the element `<autorotate>` in the MFFGOF profile file can be used to place the page correctly without a copy group on a medium in portrait orientation, see also *chapter 5.3.2.7. Page Orientation on page 239*.

5.3.3. MFFIJP

Note: It is recommended to generate separate profiles for spot color and full color mode.

Sample profile file names:

- mffijp_spotcolor.pro (spot color)
- mffijp_cmyk.pro (full color)

5.3.3.1. Global Settings (Spot Color)

In the following global settings are described, which overwrite the predefined filter settings.

Example: Global settings in the MFFAFP profile

```
<warninglevel   value="2"      />
<rastersettype  value="screen" />
```

<warninglevel>

Element specifies the warning type for warning messages issued during the process run. The value of the attribute `value` specifies the warning level. The default value is `2`. Levels 0 to 3 are available.

<rastersettype>

Element is used to provide a specified area with a defined pattern type. The values of the attribute `value` are `screen`, `screenmodified`, `afp`, `afpmodified`, `roughw0`, `roughw15`, `roughw45`, `roughw75`, `fine0`, `fine15`, `fine45`, and `fine75`.

5.3.3.2. General Settings (Spot Color)

Example: General Settings in the MFFIJP profile

```
<fullcolor           value="false"  />
<cutout              value="false"  />
<cuemark             value="none"   />
<OptimizationMethod  value="font"   />
<fontfile            value="none"   />
<RespectRefPoint     value="true"   />
<overlayasreference  value="false"  />
```

<fullcolor>

Element controls whether a job is printed in spot color or full color mode. The values of the attribute `value` are `false` (spot color mode) and `true` (full color mode).

<cutout>

If a spot color object (e.g. yellow rasterized area) is overlaid by a black object (e.g. text), the silhouette of the black object is cut out of the spot color object. The procedure prevents that the inks intermingle while printing on certain types of paper. **Note:** Element is valid for spot color mode only.

<cuemark>

Element controls generation of a cue mark that is printed with the black print head on the front page. For duplex printing, the cue mark controls the synchronization of front and back page for printing on the back page. The values of the attribute `value` are `left`, `right`, and `none`, i.e. marks are printed at the left or right margin or they are preprinted already on the paper.

<OptimizationMethod>

Element controls the methods used to optimize and minimize the IJPDS data volume. The values of the attribute `value` are `font`, `fixedfile`, and `none`. The default value is `font`. The value `font` controls the optimization with macro fonts, the value `fixedfile` with fixed file macros. If the value `none` is specified, optimization is ignored.

<fontfile>

Element controls the font file processing. The font files have the extension .ffi. The values of the attributes `value` are `import`, `export`, and `none`. If the value `export` is specified, all raster fonts are exported to font files. If the value `import` is specified, all files with the extension .iff are imported. If the value `none` is specified, fonts are generated by the font rasterizer.

<RespectRefPoint>

Element controls the occurrence of a thin line in a rasterized area that actually consists of two areas, but the pixels do not have an offset, i.e. a black pixel of one area contacts a black pixel of the other.

The values of the attribute `value` are `false` and `true`, i.e. line formation is suppressed.

<overlayasreference>

See *chapter 5.3.1.1. Global Settings on page 224*.

5.3.3.3. RIP and Print Head Configuration (Spot Color)

Note: For detailed information about raster image processors (RIP) and print heads, see *Scitex Printing Systems IJPDS Formats.*

Example: RIP and Print Head Configuration in the MFFIJP profile

```
<!-- === RIPs === Do not modify RIP sequence ============================== -->
    <RIP id='0' type="front">
        <printheads ResX="300" ResY="300">
            <PH Jets="2688" Drops="1" RelPos="0" />
        </printheads>
    </RIP>
    <RIP id='1' type="front2">
        <printheads ResX="300" ResY="300">
            <PH Jets="2688" Drops="1" RelPos="0" />
        </printheads>
    </RIP>
    <RIP id='2' type="frontcolor" hsbcolor="#00355555" hueangle="30">
        <printheads ResX="300" ResY="300">
            <PH Jets="2688" Drops="1" RelPos="0" />
        </printheads>
    </RIP>
    <RIP id='3' type="backcolor" hsbcolor="#00355555" hueangle="30">
         <printheads ResX="300" ResY="300">
            <PH Jets="2688" Drops="1" RelPos="0" />
        </printheads>
    </RIP>
    <RIP id='4' type="back">
        <printheads ResX="300" ResY="300">
            <PH Jets="2688" Drops="1" RelPos="0" />
        </printheads>
    </RIP>
    <RIP id='5' type="back2">
        <printheads ResX="300" ResY="300">
            <PH Jets="2688" Drops="1" RelPos="0" />
        </printheads>
    </RIP>
```

A RIP uses input character data and fonts from the IJPDS job file to build a bitmap in a page buffer that represents the image to be printed. Each bit in a page buffer represents a dot in the image to be printed. Each bitmap represents one page of the document to be printed. In the IJPDS job file the physical RIPs are numbered consecutively starting with 0.

Note: Although a RIP can control up to four print heads, Compart does not recommend a configuration with more than one print head per RIP for a print job.

`<Rip>`

Element defines the physical RIP. The attributes are `id`, `type`, `hsbcolor`, and `hueangle`. The value of the attribute `id` is the RIP number. The value of the attribute `type` is the RIP identifier name:

`front`	front page
`front2`	front page 2-up
`back`	back page
`back2`	back page 2-up
`frontcolor`	spot color front page
`front2col`	spot color front page 2-up
`backcolor`	spot color back page
`back2col`	spot color back page 2-up

The hexadecimal value of the attribute `hsbcolor` (hsb color orientation) specifies the position in the color wheel, i.e. colors of the visible spectrum arranged into a circle. `hsbcolor` has the following format: `'#HHHHSSBB'`. Each color can be defined and described with three characteristics: hue (H), saturation (S), and Brightness (B).

The value of the attribute `hueangle` is the angle specified by the degree. The color located in the angle segment is interpreted as spot color.

`<printheads>`

Element is a child element of `<Rip>` and specifies the print head definition. The value of the attribute `ResX` is a fixed value of `300` dpi. The value of the attribute `ResY` (direction of paper transport) can be `300` dpi or `600` dpi.

`<PH>`

Element is a child element of `<printheads>` and specifies the nozzle device of the print head. The value of the attribute `Jets` specifies the number of nozzles. In the above example 2688 nozzles are specified which results in width of 9 inches with 300 dpi.

The value of the attribute `Drops` specifies the number of ink drops ejected by one nozzle dot a single dot. The default value is one drop.

The value of the attribute `RelPos` is a binary number that specifies the print head offset to the start position of the first left most print head. If the value `0` is specified for all print heads, the positioning of print heads is ignored.

`<gammalist>`

See *chapter 5.3.1.9. Output AFP Files Settings on page 231*.

Note: In below description for the sample profile file mffijp_cmyk.pro (full color) only elements are listed that differ from elements of the profile file mffijp_spotcolor.pro (spot color).

5.3.3.4. General Settings (Full Color)

Example: General Settings in the MFFIJP profile

```
<fullcolor           value="true"   />
<cutout              value="false"  />
<cuemark             value="none"   />
<OptimizationMethod  value="font"   />
<fontfile            value="none"   />
<RespectRefPoint     value="true"   />
<overlayasreference  value="false"  />
```

`<fullcolor>`

Element controls whether a job is printed in spot color or full color mode. The values of the attribute `value` are `false` for spot color and `true` for full color mode. The default value for full color is `true`.

5.3.3.5. RIP and Print Head Configuration (Full Color)

Note: For detailed information about raster image processors (RIP) and print heads, see *Scitex Printing Systems IJPDS Formats.*

Example: RIP and Print Head Configuration in the MFFIJP profile

```
<!-- === RIPs ==== RIP/PRINTHEAD CONFIGURATION ============================== -->
    <RIP id='0' type="frontcyan">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
      </printheads>
      </RIP>
      <RIP id='1' type="backcyan">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
      </printheads>
      </RIP>
      <RIP id='2' type="frontmagenta">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
      </printheads>
      </RIP>
      <RIP id='3' type="backmagenta">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
      </printheads>
      </RIP>
      <RIP id='4' type="frontblack">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
      </printheads>
      </RIP>
      <RIP id='5' type="backblack">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
              </printheads>
      </RIP>
      <RIP id='6' type="frontyellow">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
              </printheads>
      </RIP>
      <RIP id='7' type="backyellow">
              <printheads ResX="300" ResY="300">
                      <PH Jets="2688" Drops="1" RelPos="0" />
      </printheads>
      </RIP>
```

`<Rip>`

Element defines the physical RIP. The attributes are `id`, `type`, and `type`. The value of the attribute `id` is the RIP number. The value of the attribute `type` is the RIP identifier name made up of position and color:

`frontcyan`	front page cyan
`backcyan`	back page cyan
`frontmagenta`	front page magenta
`backmagenta`	back page magenta
`frontblack`	front page black
`backblack`	back page black
`frontyellow`	front page yellow
`backyellow`	back page yellow

Example: Definition of a Color Profile List in the MFFIJP profile

The ICC profile is a standardized dataset that described the color space of a hardware device (here printers, also color monitors or scanners). Windows accepts profile files with the extension .ICC (Mac) or .ICM (Windows).

Note: Some application programs identify just one of the file extensions, i.e. if necessary files must be renamed.

```
<colorprofilelist>
   <!--<colorprofile name="JapanWebCoated.icc" />                    -->
       <colorprofile name="ScitexVM_Profil01_Draft_TR_310304.icm" />
   <!--<colorprofile name="ScitexVM_Profil02_Draft_TR_310304.icm" /> -->
   <!--<colorprofile name="ScitexVM_Profil03_Draft_TR_310304.icm" /> -->
   <!--<colorprofile name="ScitexVM_Profil04_Draft_TR_310304.icm" /> -->
</colorprofilelist>
```

`<colorprofile>`

Element specifies the name of the color profile. The value of the attribute `name` contains the name of the color profile. **Note:** The list can contain any number of profiles. Just the current profile should be uncommented.

5.3.3.6. Resources (Full Color)

Example: Resource Definition in the MFFIJP profile

```
<resourcelist>
      <files   path="C:\PROJECTS\ICC" ext="icm" />
      <files   path="C:\PROJECTS\ICC" ext="icc" />
      <files   path="."               />
</resourcelist>
```

`<resourcelist>`

With the child element `<files>` all resources, i.e. the color profile files are specified. The value of the attribute `path` contains the path for the color profile. **Note:** Windows accepts profile files with the extension .ICC (Mac) or .ICM (Windows).

5.3.4. MFFIPD

IPDS (Intelligent Printer Data Stream) is a printer protocol of IBM. Although IPDS and the well know AFPDS (Advanced Function Presentation Data Stream) are internally quite similar, you may not mix them up. For further IPDS information, see *IBM Intelligent Printer Data Stream Reference*.

5.3.4.1. Output File Settings

Following child elements of `<output>` are available:

`<rastersettype>`

If areas are written with filling patterns, different types of patterns are available. Element is used to provide a specified area with a defined pattern type. The values of the attribute `value` are `screen`, `screenmodified`, `afp`, `afpmodified`, `roughw0`, `roughw15`, `roughw45`, `roughw75`, `fine0`, `fine15`, `fine45`, and `fine75`.

Example: Printer Definition in the MFFIPD profile

Note: The example displays just an extract of the elements and attribute defined in `<ipdsprinter>`.

```
<printerlist              default                 = "Printer1">
     <ipdsprinter         name                    = "Printer1"
                          vendor                  = "IBM"
                          connection              = "tcpip"
                          userelativefontmetrics  = "false"
                          resolution              = "1440"
                          datapacketsize          = "8"
                          maxpagesegments         = "127"
                          maxoverlays             = "254"
                          maxfonts                = "128"
                          pagesperacknowledge     = "1"
                          reportundefinedchars    = "true"
                          reportprintoutsidepage  = "true"
                          positioncheckhighlight  = "true"
                          reportotherexceptions   = "true"
                          usedoublebytefonts      = "false"
                          enablecompression       = "false"
                          useioimagecommand       = "false"
                          stackreceivedpages      = "false"
                          >

      </ipdsprinter>
</printerlist>
```

`<ipdsprinter>`

Element controls the print output with the following attributes:

`name`: The value of the attribute is a freely defined printer name. From an application the printer is accessible with this name. The printer is called with type and subtype.
Example: of command line program call where *ipd* is the type and *Printer1* is the subtype:
`cpmcopy –i input.afp –o –type ipd.Printer1`

`vendor` (required): The value of the attribute specifies the name of the printer manufacturer. The printers may differ in the connection set-up and establishment. But the standard IPDS communication applies to all printers. The values are `IBM`, `NIPSON`, `OCE`, and `XEROX`.

`connection`: The value of the attribute specifies the name of the connection definition, i.e. the connection between system and printer. The values are `tcpip`, `file`, and `scsi`. See also element `<connectionlist>` for the connection definitions.

`userelativefontmetrics` (required): The value of the attribute specifies loading of relative fonts to the printer. So, big fonts can be loaded that cannot be loaded otherwise as fixed fonts. The values are `true` and `false` (default). Recommendation: If the printer does not start a print job with small fonts and the value `false`, use the value `true`.

`resolution`: The value of the attribute specifies the print resolution. If you do not use the attribute, the printer's internal defined resolution is used when a job is printed. Recommendation: For good print results do not use the attribute.

`datapacketsize`: The value of the attribute specifies the data packet size, when transferring data the printer. Recommendation: A value of 8 KB can be handled by all printers. If you specified a higher value, a higher data transfer rate is used. **Note:** The printer manufacturers' datasheets inform you about the maximum transfer speed. It may happen that with certain speed values IPDS errors occur, because they are not supported. Recommended values are: IBM (old): `8`, IBM (new): `16`, Nipson: `16`, Océ (old): `12`, Océ (new): `16`.

`maxpagesegments`: The value of the attribute specifies the maximum number of page segments that can be processed. The number depends on the installed printer. If the value `0` is specified, no page segments are processed.

`maxoverlays`: The value of the attribute specifies the maximum number of overlays that can be processed. If the value `0` is specified, no overlays are processed. Newer IBM or Océ printers support up to 2047 overlays.

`maxfonts`: The value of the attribute specifies the maximum number of fonts that can be loaded to the printer.

`pagesperacknowledge`: The value of the attribute specifies the number of pages sent to the printer without generating acknowledgement messages for received pages.

`reportundefinedchars`: The value of the attribute specifies that the printer gives informational messages about characters to be printed that are not part of the defined character set. The values are `true` (default) and `false`.

`reportprintoutsidepage`: The value of the attribute specifies that the printer gives informational messages before data is printed outside of the logical page. The values are `true` (default) and `false`.

`positioncheckhighlight`: The value of the attribute controls the printing of PEMs (Print-Error Marker). A PEM is a small, rectangular mark that indicates incorrectly placed data in the valid printable area. The values are `true` (default) and `false`.

`reportotherexceptions`: The value of the attribute specifies that the printer gives informational messages about exceptions that are not covered by the two above mentioned attributes. The values are `true` (default) and `false`.

`usedoublebytefonts`: The value of the attribute specifies whether the printer is able to print double-byte fonts. The values are `true` and `false`. The default value is `false`. If the value `false` is specified, a maximum number of 256 x 255 characters can be used.

`enablecompression`: The value of the attribute specifies that images with Fax G4 compression can be sent to the printer. The values are `true` and `false` (default).

`useioimagecommand` (Nipson): The value of the attribute specifies that the IPDS IO Image Command Set is to be used. The values are `true` and `false` (default).

`stackreceivedpages` (IBM/Océ): The value of the attribute specifies that the printed pages of print job are to be moved out of the print engine area (to the stacker), when the print job is completed. Examples: Océ: For each print job, the Twin System units stay synchronized. IBM: The print job is moved completely through a finisher (sorting, punching, stapling), if installed. The values are `true` and `false` (default).

`<renderoverlays>`

Element controls with the values of the attribute `value`, if overlays are rasterized before they are sent dot the printer. The values are `true` (default) and `false`.

Example: Tray Definition in the MFFIPD profile

```
<traylist>
<!-- = paper source trays mapping ================================
       ATTENTION: The IPDS trays are unlike the AFP trays are 0-based!
      ========================================================== -->

 <inputtray deviceid='0' name="Tray1" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>

 <inputtray deviceid='1' name="Tray2" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>

 <inputtray deviceid='2' name="Tray3" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>

 <inputtray deviceid='3' name="Tray4" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>

 <inputtray deviceid='4' name="Tray5" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>

 <inputtray deviceid='5' name="Tray6" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>

 <inputtray deviceid='6' name="Tray7" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>

 <inputtray deviceid='7' name="Tray8" >
  <!-- <papersize format = "A4" orientation = "portrait" /> -->
 </inputtray>
</traylist>
```

`<inputtray>`

The child element `<inputtray>` of `<traylist>` specifies the tray definition with the values of the attributes `deviceid` and `name`. **Note:** The tray definition in the MFFIPD profile is 0-based, whereas in the MFFAFP profile it is 1-based.

`<papersize>`

Element defines with the values of the attribute `format` the paper size and with `orientation` the page orientation, the values are `landscape` and `portrait`. For values of the attribute format, see *Appendix C: Paper Formats on page 280*.

Example: Gamma value/Tone correction list in the MFFAFP profile

```
<gammalist>
     <gammavalue from=  '0' to=  '0'/>
     <gammavalue from=  '7' to= '48'/>
     <gammavalue from= '15' to= '72'/>
     <gammavalue from= '23' to= '88'/>
       .
       .
       .
     <gammavalue from='199' to='224'/>
     <gammavalue from='207' to='232'/>
     <gammavalue from='215' to='232'/>
     <gammavalue from='223' to='240'/>
     <gammavalue from='231' to='240'/>
     <gammavalue from='239' to='248'/>
     <gammavalue from='247' to='248'/>
     <gammavalue from='255' to='255'/>
</gammalist>
```

<gammalist>

Element controls the gamma value correction/tone correction. The element <gammavalue> defined as child element of <gammalist> contains the tone correction values for rasterized areas. The attribute from specifies the input value and the attribute to the output value.

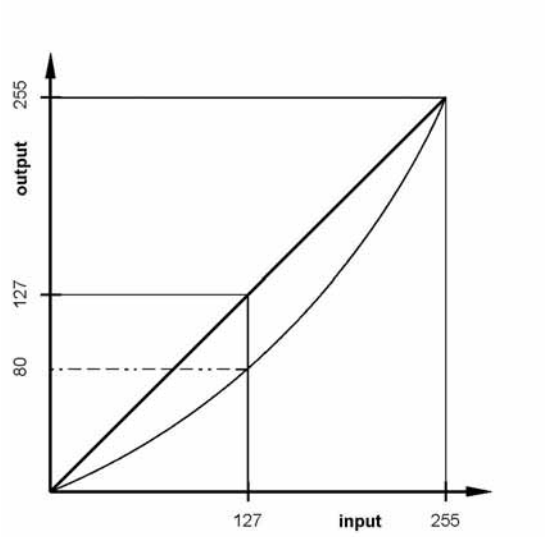


Figure 7: Sample Chart – Tone correction in MFFIPD Profile

Example: Printer connection types in the MFFIPD profile

```
<connectionlist>
   <tcpipconnection name                = "tcpip"
                    host                = "192.168.11.12"
                    port                = "5001"
                    bufferpages         = "10"
                    startpurge          = "false"
                    twin                = "false"
                    discarddataafterstop = "false"
                    tracefile           = "ipds_tcpip.trace" />

   <fileconnection  name                = "file"
                    filename            = "output.ipds"
                    splitfile           = "false"
                    tracefile           = "ipds_file.trace" />

   <scsiconnection  name                = "scsi"
                    adapter             = "0"
                    id                  = "5"
                    lun                 = "0"
                    bufferpages         = "10"
                    twin                = "false"
                    discarddataafterstop = "false"
                    tracefile           = "ipds_scsi.trace" />
</connectionlist>
```

<connectionlist>

Element specifies in the child elements <tcpipconnection>, <fileconnection> and <scsiconnection> the printer definitions with the following attributes:

name: name that is used to reference the connection specified in the attribute connection of the element <ipdsprinter>.

host: IP address of the printer.

port: port number of the printer.

bufferpages: number of pages that can be buffered in the printer.

startpurge: value specifies, whether a PURGE command is sent to the printer after a connection was established. The values are true and false. The default value is false.

twin: value specifies, whether the defined printer is an Océ printer provided with a Twin System. The values are true and false (default).

discarddataafterstop: no function.

tracefile: name of trace file.

filename: name of output file.

splitfile: value specifies, the output shall be split into two or more files. The values are true and false (default).

adapter: name of SCSI adapter.

`id`: SCSI device number.

`lun`: Logical Unit Number.

5.3.4.2. Fonts

See also *chapter 5.3.6.1. Fonts on page 261* and *Example: Font definitions in the MFFPDF profile on page 261* for the font definitions in the MFFIPD profile.

5.3.4.3. Resources

See also *chapter 5.3.6.2. Resources on page 261* and *5.3.6.2. Resources on page 261* for the resource definitions in the MFFIPD profile.

5.3.5. MFFPCL

Note: The element <optimizerastertext> is obsolete. Do not use it anymore.

5.3.5.1. Global Settings

In the following global settings are described, which overwrite the predefined filter settings.

Example: Global Settings in the MFFPCL profile

```
<globals>
   <warninglevel   value="2"      />
<globals/>
```

<warninglevel>

Element <warninglevel> as child element of <globals> specifies the warning type for messages issued during the process run. The value of the attribute value specifies the warning level. The default value is 2. Levels 0 to 3 are available.

5.3.5.2. Code Pages

Example: Code Page Definition in the MFFPCL profile

```
<codepagelist>
  <codepage name="12U" iana="IBM850"/>
</codepagelist>
```

<codepagelist>

The child element <codepage> of <codepagelist> defines several code pages.

5.3.5.3. Fonts

Example: Font Definition in the MFFPCL profile

```
<fontlist>
  <font family="Courier">
     <face weight="MEDIUM" style="UPRIGHT">
        <raster fontfile="DATAM4" fontfiletype="PCL" size="1.599999"
                                              fixedsymbolset="TRUE"/>
        <raster fontfile="cour" fontfiletype="TrueType"/>
     </face>
<!-- <face weight="BOLD" style="UPRIGHT" fontselector="0p0s3b4102T"  embed="NEVER" /> -->
  </font>
</fontlist>
```

5.3.5.4. Trays

For additional information, see also *chapter 5.3.1.6. Trays on page 228.*

Example: Tray Definition in the MFFPCL profile

```
<traylist>
<!-- = inputtray   ================================================ -->
  <inputtray deviceid='1' name="UpperTray"  mediatypeid='2'/>
  <inputtray deviceid='2' name="ManualFeed"      />
  <inputtray deviceid='3' name="ManualEnvelope"  />
  <inputtray deviceid='4' name="LowerTray"       />
  <inputtray deviceid='5' name="LargeCapacity"   />
  <inputtray deviceid='6' name="Envelope"        />
  <inputtray deviceid='7' name="Automatic"       />
</traylist>
```

`<inputtray>`

Element `<traylist>` as child element of `<inputtray>` specifies the tray definition for input and output. The attributes are `deviceid`, `name`, and `mediatypeid`. The value of the attribute `mediatypeid` specifies the printing medium:

`0`	plain paper
`1`	bond paper
`2`	special paper
`3`	glossy film
`4`	transparency film

5.3.5.5. Color Profile

Example: Definition of a Color Profile List in the MFFPCL profile

```
<colorprofilelist>
   <!-- <colorprofile name="GenericRGB"  /> -->
   <!-- <colorprofile name="GenericCMYK" /> -->
   <!-- <colorprofile name="GenericGray" /> -->
</colorprofilelist>
```

`<colorprofile>`

Element specifies the name of the color profile. The values of the attribute `name` are `GenericRGB`, `GenericCMYK`, and `GenericGray`. **Note:** The list can contain any number of profiles. The current profile should be without comment characters.

5.3.5.6. Input

In the following default settings are described, which overwrite the predefined filter settings. All elements are child elements of `<input>`.

Example: Default Settings in the MFFPCL profile

```
<defaultsetting>
   <pagesize value="26"/>
   <pagelength value="68"/>
   <imageresolution value="300"/>
   <symbolset value="341"/>
</defaultsetting>
```

The following child elements of <`defaultsetting`> are available:

`<pagesize>`

Element defines the paper size. The value of the attribute `value` specifies the paper size. The supported page sizes and corresponding values are listed in *Table 13: Paper* Sizes *on page 258*.

`<pagelength>`

Element controls the form length. The value of the attribute `value` specifies the number of lines.

`<imageresolution>`

Element controls the resolution. The value of the attribute `value` specifies the resolution.

`<symbolset>`

Element controls the symbol set. The value of the attribute `value` specifies a symbol set. The default symbol set is PC-8.

The various symbol sets include symbols and characters as well as language dependent special characters, graphic characters, scientific symbols, etc. The supported symbol sets and related values are listed in *Table 14: Symbol* Sets *on page 259*.

Table 13: Paper Sizes

Format Name	Value	Format Name	Value
A3	27	**Executive**	1
A4	26	**Hagaki Postcard**	71
A5	25	**JIS B4 Pager**	46
Envelope #10	81	**JIS B5 Paper**	45
Envelope B5	100	**Ledger**	6
Envelope C5	91	**Legal**	3
Envelope DL	90	**Letter**	2
Envelope Monarch	80	**Oufuku-Hagaki Postcard**	72

Table 14: Symbol Sets

Name	Value	Name	Value
Desktop	234	**MC-Text**	394
Iso L1	14	**PC-1004**	298
Iso L2	78	**PC-775**	853
Iso L5	174	**PC-8**	341
Iso L6	206	**PC-8 DN**	373
Iso L9	302	**PC-8 TK**	308
Iso-11	19	**PC-850**	405
Iso-15	9	**PC-852**	565
Iso-17	83	**PS Text**	330
Iso-21	39	**Roman-8**	277
Iso-4	37	**Win 3.0**	309
Iso-6	21	**Win Balt**	620
Iso-60	4	**Win L1**	629
Iso-69	38	**Win L2**	293
Legal	53	**Win L5**	180

Example: Overlay Definition in the MFFPCL profile

```
<macrolist>
    <file name="EDI_4035M.pcl" >
       <macro name="Overlay"/>
    </file>
</macrolist>
```

`<macrolist>`

Element controls the access of a PCL file to an external macro file to integrate an overlay. The child element `<file>` of `<macrolist>` specifies the path of the macro file.

5.3.5.7. Output

In the following default settings are described, which overwrite the predefined filter settings. All elements are child elements of `<output>`.

`<nohpgl>`

Element controls the generation of HPGL (Hewlett Packard Graphics Language). The values `true` and `false` (default) of the attribute `value` specify the HPCL support of the PCL printer.

`<color>`

Element defines whether the specified printer device supports color or not, or whether the output written to a file shall contain color or not. The values of the attribute `value` are `true` and `false` (default).

`<fullprintablearea>`

Element controls with the values `true` and `false` of the attribute `value`, whether the printer can use the complete physical page (edge-to-edge printing).

5.3.5.8. Resources

For additional information about resource definitions, see also the adequate chapters of other filters.

Example: Resource Definition in the MFFPCL profile

```
<resourcelist>
   <files   path="resource" type="PCL" extension="FLJ"/>
   <files   path="resource" type="PCL" extension="ovl"/>
   <files   path="\resource" type="ICC" extension="icc"/>
   <files   path="\resource" type="ICC" extension="icm"/>
</resourcelist>
```

`<resourcelist>`

With child element `<files>` all resources are specified. The value of the attribute `path` contains the path information for the resources. The values of the attribute `type` can be `PCL`, `TrueType`, `Type1Font`, or `Type1Metrics`.

5.3.6. MFFPDF

5.3.6.1. Fonts

Example: Font definitions in the MFFPDF profile

```
<fontlist>
  <font family="Gothic" serifstyle="SERIF" spacing="MONOSPACED" >
    <face weight="MEDIUM" width="NORMAL" style="UPRIGHT" devname="Gothic"/>
  </font>

  <font family="Univers CE" reftype="TrueType">
    <face weight="MEDIUM" style="UPRIGHT" devname="Univers" fontfile="Univer"
                          fontfiletype="TrueType" embed="always"/>
    <face weight="BOLD"   style="UPRIGHT" devname="Univers-Bold"
                          fontfile="Univerb" fontfiletype="TrueType"/>
  </font>
</fontlist>
```

`<fontlist>`

Element specifies for which font name in the Presentation Area which font should be used, when PDF files are written. In PDF files fonts can be referenced as well as completely included.

If a TrueType font is to be referenced or included, the attribute `reftype` with the value `TrueType` has to be specified in the element `<font>`. Otherwise, a reference to a Type 1 font is written.

If the attribute `fontfiletype with the value TrueType` is specified for the element `<face>`, it is searched for the font with extension `ttf` set by `fontfile`, otherwise `pfb` (for Type 1). The font name can be specified as absolute name (with path). Otherwise, it is searched in the current directory or in one of the paths, which are specified as described in chapter *5.3.6.2. Resources on page 261*.

5.3.6.2. Resources

`<resourcelist>`

Element defines the font location. If font files are not specified with an absolute path in the element `<fontlist>`, paths can be defined in the element `<resourcelist>`.

Attention: When search paths are specified, the current directory is not scanned!

Example: Resource definition in the MFFPDF profile

```
<resourcelist>
   <!-- = Files ================================================
        type can be "TrueType", "Type1Font", "Type1Metrics" if it is not
        defined, the path will be used for all types extension can be empty,
        <extension> or wildcard
      ========================================================== -->
    <files   path="."   type="TrueType"    extension="ttf"/>
    <files   path="\Programme\Adobe\Acrobat 5.0\Resource\Font"
             type="Type1Font" extension="pfb"/>
    <files   path="\Programme\Adobe\Acrobat 5.0\Resource\Font\PFM"
             type="Type1Metrics" extension="pfm"/>
    <files   path="\WINDOWS\Fonts" type="TrueType" extension="ttf"/>
    <files   path="c:\fonts"
</resourcelist>
```

5.3.6.3. Output

In the following default settings are described, which overwrite the predefined filter settings. All elements are child elements of `<output>`.

`<replacepattern>`

Element controls the use of fill patterns for rectangular areas. Fill patterns can be placed by the appropriate gray values. The values of the attribute `type` are `false` and `true`.

If `true` is specified, the fill pattern is replaced by the gray value. If `false` is specified, the fill pattern is used. The default value is `false`. **Note:** A replacement reduces the file size and increase the writing performance.

`<noimagemasks>`

Element controls the writing of image masks (transparent masking also know as stencil masking) to ‘normal images’. The values of the attribute value are `false` (default) and `true`. **Note:** The conversion increases the compatibility with (HP) printer drivers.

`<jpegquality>`

Element controls visual integrity. The value of the attribute `value` contains a percentage number. The default value is `75`. **Note:** Visual integrity improves at the expense of the file size.

`<fonthandling>`

Element controls the handling of fonts that are not defined in the element `<font>`, see all *chapter 5.3.6.1. Fonts on page 261*. The values of the attribute `value` are `auto`, `rasterize`, `converttotype3`, `usestandardfonts`, `usebestmatchingfonts`, and `useoriginalfonts`.

If you specify `auto` as value of the attribute `default`, an automated and optimized conversion is executed based on the available fonts.

If you specify `rasterize` as value of the attribute `default`, all characters are replaced by images. **Note:** There are no alignment problems. No characters are missing. The file size increases. The text is not searchable. There are known and Adobe-documented compatibility problems with (HP) printer drivers. You can use the element `<noimagemask>` to avoid problems. The attribute value is used automatically, if missing characters are detected in the specified fonts and raster information is available.

If you specify `converttotype3` as value of the attribute `default`, all fonts specified in the input filter are replaced by Adobe PostScript Type 3 fonts. **Note:** The characters and its appearance are the same as for the specified input fonts. The text is searchable. The file is smaller than a file generated with the attribute value `rasterize`. There are no alignment problems or missing characters. Possible compatibility problems cannot be solved with the element `<noimagemasks>`, because Acrobat does not accept non-transparent Type 3 glyphs. If files are displayed with Acrobat Reader 5, the representation may not be accurate. Therefore, it is recommended to upgrade to a higher version.

If you specify `usestandardfonts` as value of the attribute `default`, the 14 Adobe Type 1 standard fonts are used. **Note:** The file size is small. Alignment problems and missing characters may occur, if no raster information of the input file is available. There are no compatibility problems. The attribute value is used automatically, if the fonts specified in the input file do not have font metric information.

If you specify `usebestmatchingfonts` as value of the attribute `default`, the Multiple Master fonts are use in the output file with the same font metrics as in the input file fonts. **Note:** There are no alignment problems. Missing character may occur, if no raster information of the input file is available. If the input file fonts are not embedded, the output file size decreases. Compatibility problems may occur with (HP) printer drivers. The attribute value is used automatically, if the input file fonts are not available, but font metric information of input file fonts are known.

If you specify `useoriginalfonts` as value of the attribute `default`, the fonts of the input file are embedded if there are TrueType or Type 1 fonts. **Note:** There are no alignment or missing characters problems. Even if the attribute value `auto` is specified, the font handling works primarily under the terms of the value `useoriginalfonts`.

`<writearchivefields>`

Element controls whether archive fields are written as annotations. The values of the attribute `value` are `not` and `annotations`. If `not` is specified, writing of archive fields is ignored. If `annotations` is specified, archive fields (AFP TLEs and index values) are written as PDF annotations (comments).

Note: In the data stream, Acrobat comments cannot be assigned to a specified page. Whereas annotations generated with `<writearchivefields>` can be assigned to specific pages.

`<stampaswatermark>`

Element controls the positioning of a stamp. The values of the attribute `value` are `false` (default) and `true`. If you specify the value `false` for the attribute `value`, the stamp is treated like an overlay. The stamp overlays other elements on a page. It is placed in the foreground, see also elements `<generateoverlay` and `<overlayasreference>` in *chapter 5.3.1.1. Global Settings on page 224*. If you specify the value `true`, elements of a page overlay the stamp. The stamp is placed in the background like a watermark.

5.3.6.4. Font Optimization

Note: The description in this chapter applies to binary copy only.

In standard mode, binary copy does hardly interpret or parse the objects in a PDF file. This makes binary copy considerably faster than normal PDF parsing. However, when concatenating many PDF files, each of them using the same fonts and the lack of parsing results in a replication of the common fonts in the generated file. This may not only increase the size of the resulting file, but also slows down future processing.

To avoid increasing file size and reduced processing speed you can use the element `<font-optimization>`.

But be aware that font optimization may be an extremely time-consuming operation. The same is true for testing if two fonts are identical, because PDF allows two fonts with the same name to have different embedded font files. Even if font names and font files are the same, the fonts may have different encodings. To make things worse, PDF may declare different character widths tables for two otherwise identical fonts. And in the end, in order to test fonts identity, a lot of PDF objects which were not parsed in standard mode, need now parsing. However, if you are sure that the fonts you want to merge are identically (e.g. they are produced in the same way), you can to control the level of accuracy in font equality proving.

`<font-optimization>`

Element controls font optimization. If you specify `false` as value of the attribute `value`, no font optimization is done. If you specify `true` as value of the attribute `value`, you use it together with the values of the attribute `fontidentity` to define an accuracy level for the font optimization. The values are:

`exact`: Default value. All available font information is used for processing.

`low`: You use the level, if all the fonts are not embedded and fonts with identical names have also identical encoding and this encoding is not tagged “Custom”. At this level font file identity and encoding identity is not examined.

`medium`: You use this level, if some fonts fulfill the criteria listed in the `low` level and there are some embedded fonts so that any two fonts with the same name have the same encoding or the encoding is tagged “Built-in”. At this level font encoding identity is not tested, except for “Built-in” encodings. Proving font file equality implies proving “built-in” encodings identity.

`high`: With this level processing works in the same manner as with `exact` level in most of the cases, but faster. PDF character widths modification tables are not examined for identity. **Note:** Although it does not make sense, defining two fonts with identical font files and identical encodings with different character widths is tolerated by Acrobat applications. If two fonts with different character widths are considered equal, then you get an “optimized” file with bad aligned, bad positioned and overlapping text.

Conclusion: If you know what you are doing and you do not have any doubts, Compart recommends the `high` level. If you choose `low` or `medium` where you should not, you may get unwanted the results. Acrobat error messages can be issued such as: “Cannot extract the embedded font...”.

However the lower levels will save a lot of time and are indeed useful when you are sure that any two fonts are identical. Compart recommends the `medium` level, if fonts are embedded and the `low` level, if there are no embedded fonts.

5.3.7. MFFPOS

5.3.7.1. Fonts

<fontlist>

Element controls the assignment of font names to fonts in the presentation area when PostScript file is generated. In PostScript file fonts can be referenced as well as fully integrated.

A font name specified in the element <font family> must correspond to the value of the attribute fontfiletype of the element <face>.

The font name can be specified with an absolute path. Otherwise, the search is executed in the current directory or in one of the paths, which are specified as described in chapter *5.3.6.2. Resources on page 261*.

Example: Font Definitions in the MFFPOS profile

```
<fontlist>
  <font family="Arial,Bold">
    <face devname="Arial-BoldMT" fontfile="_AB_____" fontfiletype="Type1"/> </font>
  <font family="Arial,BoldItalic">
    <face devname="Arial-BoldItalicMT" fontfile="_ABI____" fontfiletype="Type1"/> </font>
  <font family="ArialMT">
    <face devname="ArialMT" fontfile="_A_____" fontfiletype="Type1"/>
  </font>
  <font family="Helvetica">
    <face devname="ArialMT" fontfile="_A_____" fontfiletype="Type1"/>
  </font>
  <font family="Courier">
    <face devname="Courier" fontfile="COM____" fontfiletype="Type1"/>
  </font>
  <font family="CourierNew">
    <face devname="Courier" fontfile="COM____" fontfiletype="Type1"/>
  </font>
    <font family="Courier,New">
    <face devname="Courier" fontfile="COM____" fontfiletype="Type1"/>
  </font>
  <font family="Courier New">
    <face devname="Courier" fontfile="COM____" fontfiletype="Type1"/>
  </font>
  <font family="Symbol">
    <face devname="Symbol" fontfile="SY______" fontfiletype="Type1"/>
  </font>
  <font family="Univers CE">
    <face devname="TimesNewRomanPSMT" fontfile="_ER_____" fontfiletype="Type1"/>
  </font>
</fontlist>
```

5.3.7.2. Output

Element <output> controls the output process with following child elements:

<fonthandling>

Element specifies the handling of fonts that are not defined with the element <font>, see also *chapter 5.3.6.1. Fonts on page 261*. The values of the attribute value are auto, none, and converttotype3.

If auto is specified, the font is not embedded, it is referred with the font metrics of the input font.

If no font is specified within the element `<fontlist>` and the attribute value `none` is specified, the font is just referred but not embedded.

If no font is specified within the element `<fontlist>` and the attribute value `convertto-type3` is specified, the font is embedded as Type-3 raster font.

5.3.7.3. Resources

`<resourcelist>`

Element defines the font location. If font files are not specified with an absolute path in the element `<fontlist>`, paths can be defined in the element `<resourcelist>`.

Attention: When search paths are specified, the current directory is not scanned, see also *Example: Resource Definition in the MFFPOS profile*.

Example: Resource Definition in the MFFPOS profile

```
<resourcelist>
   <files   path="." />
   <files   path="\Programme\Adobe\Acrobat 6.0\Resource\Font" />
   <files   path="\WINDOWS\Fonts" />
   <files   path="c:\fonts"  extension ="*"   />
</resourcelist>
```

5.3.7.4. Trays

For more information, see also *chapter 5.3.1.6. Trays on page 228*.

Example 1: Tray Definition in the MFFPOS profile

```
<traylist>
<!-- = inputtray    ================================================ -->
  <inputtray deviceid='1' name="UpperTray"       />
  <inputtray deviceid='2' name="ManualFeed"      />
  <inputtray deviceid='3' name="ManualEnvelope"  />
  <inputtray deviceid='4' name="LowerTray"       />
  <inputtray deviceid='5' name="LargeCapacity"   />
  <inputtray deviceid='6' name="Envelope"        />
  <inputtray deviceid='7' name="Automatic"       />
</traylist>
```

`<inputtray>`

Element specifies the tray definition for input an output with the attributes `deviceid` und `name`.

Example 2: Tray Definition in the MFFPOS profile

```
<xeroxdocumentmedialist>
  <xeroxdocumentmedia type="UpperTray" resx="612" resy="792" weight="75"
                                                    color="white" name="form1"/>
  <xeroxdocumentmedia type="LowerTray" resx="612" resy="792" weight="75"
                                                    color="yellow" name="form2" />
  <xeroxdocumentmedia type="Envelope"  resx="612" resy="792" weight="75"
                                                    color="white"  name="form3"  />
</xeroxdocumentmedialist>
```

Element specifies the tray definition with following attributes:

`type`	DocBridge Mill tray name
`resx/resy`	paper size in Inch * 72
`weight`	paper weight
`color`	paper color
`name`	printer tray name

5.3.8. MFFXFO

5.3.8.1. Global Settings

In the following global settings are described, which overwrite the predefined filter settings.

Example: Global Settings in the MFFXFO profile

```
<globals>
   <overlayasreference value="false"      />
<globals/>
```

`<overlayasreference>`

Element controls the one-time occurrence of images in the PA. For repeated use references can be specified. The values of the attributes `value` are `true` and `false`. If you specify the value `false` for the attribute `value`, the image is imbedded in the data stream. If you specify the value `true` for the attribute `value`, a reference is specified.

5.3.8.2. Fonts

In the following font settings are described, which overwrite the predefined filter settings.

Example: Font Definition in the MFFXFO profile

```
<fontlist>
   <font family="arial" reftype="TrueType">
      <face weight="MEDIUM" style="UPRIGHT" fontfile="arial"   fontfiletype="truetype"/>
      <face weight="BOLD"   style="UPRIGHT" fontfile="arialbd" fontfiletype="truetype"/>
      <face weight="MEDIUM" style="ITALIC"  fontfile="ariali"  fontfiletype="truetype"/>
   </font>
    <font family="times new roman" reftype="TrueType">
      <face weight="MEDIUM" style="UPRIGHT" fontfile="times"   fontfiletype="truetype"/>
      <face weight="BOLD"   style="UPRIGHT" fontfile="timesbd" fontfiletype="truetype"/>
      <face weight="MEDIUM" style="ITALIC"  fontfile="timesi"  fontfiletype="truetype"/>
   </font>
 </fontlist>
```

`<fontlist>`

Element specifies with the child element `<font>` which font corresponds to the font name in the Presentation Area, when the output file is generated. In output files fonts can be referenced as well as completely imbedded.

5.3.8.3. Resources

Example: Resource Definition in the MFFXFO profile

```
<resourcelist>
   <!-- = Files=================================================
           type can be "TrueType", "Type1Font", if it is not defined,
           the path will be used for all types extension can be empty,
           <extension> or wildcard
    ========================================================== -->
   <files path="c:\WINDOWS\Fonts" type="TrueType" extension="ttf"/>
</resourcelist>
```

`<resourcelist>`

Element defines the font location. If font files are not specified with an absolute path in the element `<fontlist>`, paths can be defined in the element `<resourcelist>`.
Attention: When search paths are specified, the current directory is not scanned!

5.4. MFF Profiles Files in OS/390 and z/OS

5.4.1. Characteristics and Methods

The chapter describes characteristics and methods working with MFF profiles in USS (UNIX System Services) under IBM operating systems OS/390 and z/OS.

The first line of the profile file mffafp.pro must contain the following information:

```
<?xml version="1.0" encoding="IBM500" standalone="yes"?>
```

The value of the attribute `encoding` means that the profile is coded with the code page IBM EBCDIC International.

Working under Windows and Linux the user can edit and modify the profile. It is transferred with FTP (File Transfer Protocol) to the host system.

Note: For USS Secure Shell (SSH) is not supported. Thus, a secure connection is not supported.

For FTP file transfer the user must consider the following:

- Transfer modus must be set to ASCII
- Use of the correct code page setting with the FTP command:
  `quote site SBD=(IBM-500,ISO8859-1)`

Note: If you use the standard editor Vi to work with a file under USS, the character '`!`' (exclamation mark) is displayed as '`|`' (vertical bar). The reason is the code page 1047 which is the default code page. Compart does not support this code page. Nevertheless, the conversion works error-free.

5.4.1.1. Transferring a Profile File

Note: Instead of `remotehost` an IP address can be specified. The example assumes that the profile file mffafp.pro is in the local folder `d:\projects\testprojekt`.

```
------------------------------------------------------------------------
d:\projects\testprojekt>ftp remotehost
Verbindung mit remotehost wurde hergestellt.
220-FTPD1 IBM FTP CS V1R4 at MVS1, 23:54:24 on 2004-03-30.
220 Connection will close if idle for more than 5 minutes.
Benutzer (remotehost:(none)): compart
331 Send password please.
Kennwort: xxxxxxxxx
230 COMPART is logged on.  Working directory is "COMPART.".
ftp> asc
200 Representation type is Ascii NonPrint
ftp> quote site SBD=(IBM-500,ISO8859-1)
200 SITE command was accepted
 cd /compart/compar1
HFS directory /compart/compar1 is the current working directory
ftp> put mffafp.pro
200 Port request OK.
125 Storing data set /compart/compar1/mffafp.pro
250 Transfer completed successfully.
FTP: 4781 Bytes gesendet in 0,00Sekunden 4781000,00KB/s
ftp>
------------------------------------------------------------------------
```

In general, the contents of the MFF profile files are identical with the entries used for UNIX (AIX, Solaris, etc.).

The exception is the profile file mffafp.pro. External resources (character sets, code pages, overlays, page segments etc.) can be coded according to the UNIX syntax, i.e. a path must be specified in the UNIX file system HFS, or the resource location can be specified directly in the OS/390 or z/OS file system.

Fonts, code pages, coded fonts, overlays or page segments must be located in a PDS (Partitioned Data Set). PDS address example:

```
<files path="MVS://'SYS1.FONTLIBB("     extension=""   />
```

If fonts and code pages are located in `PDS SYS1.FONTLIBB`, the name `MVS://` must be prefixed. The character "`'`" (apostrophe) turns the name into a fully qualified name for the system. Without the character "`'`" the system prefixes the name with the current user ID.

Example: If the user ID is Compart, the output of the name `//SYS1.FONTLIBB` results in `COMPART.SYS1.FONTLIBB`.

The resources can be located in different PDS. By default, fonts, code pages, and coded fonts are located in `PDS SYS1.FONTLIBB` and overlays in PDS `AFPPROJ.OVLS`.

In the element `<resourcelist>` of the profile file mffafp.pro statements must be coded as in the example below.

```
<resourcelist>
      <|-- = library ================================================ -->
      <|-- = files ================================================
           type can be "overlay", "pagesegment", "font", "codedfont",
           "codepage", "charset", "formdef" "pagedef" or "*" extension can be
           empty, <extension> or wildcard
        ============================================================ -->
     <files   path="MVS://'SYS1.FONTLIBB("         extension=""   />
     <files   path="MVS://'AFPPROJ.OVLS("         extension=""   />
</resourcelist>
```

So the code page T1GI0382 is loaded as member `SYS1.FONTLIBB(T1GI0382)`. Then, the MFF filter for AFP issued the following message:

```
AFP1059V External resource 'T1GI0382' found in file MVS://'SYS1.FONTLIBB(T1GI0382)'
```

The example below is a MFFAFP profile that runs under USS, if it is transferred correctly.

```
<?xml version="1.0" encoding="IBM500" standalone="yes"?>
<|--*********************************************************************
    Profile for Compart AFP I/O driver.
    Copyright (C) Compart Systemhaus GmbH, 2000
**********************************************************************-->
<|DOCTYPE mffafp>
<mffafp>
   <globals>
          <defaultchar   unc="003F"/>
          <multidocument value="true"/>
   </globals>
   <fontlist>
      <font family="Helvetica" serifstyle="SANSSERIF"
                               spacing="PROPORTIONAL" >
         <face weight="NORMAL" width="NORMAL" style="UPRIGHT">
            <raster size="6"  devname="C0H20060"/>
            <raster size="7"  devname="C0H20070"/>
             .
             .
            <raster size="25" devname="C0H200T0"/>
            <raster size="36" devname="C0H200Z0"/>
         </face>
         <face weight="NORMAL" width="NORMAL" style="ITALIC">
            <raster size="6"  devname="C0H30060"/>
            <raster size="7"  devname="C0H30070"/>
             .
             .
            <raster size="25" devname="C0H300T0"/>
            <raster size="36" devname="C0H300Z0"/>
         </face>
         <face weight="BOLD" width="NORMAL" style="UPRIGHT">
            <raster size="6"  devname="C0H40060"/>
            <raster size="7"  devname="C0H40070"/>
            <raster size="8"  devname="C0H40080"/>
             .
             .
            <raster size="25" devname="C0H400T0"/>
            <raster size="36" devname="C0H400Z0"/>
         </face>
         <face weight="BOLD" width="NORMAL" style="ITALIC">
            <raster size="6"  devname="C0H50060"/>
            <raster size="7"  devname="C0H50070"/>
             .
             .
            <raster size="25" devname="C0H500T0"/>
            <raster size="36" devname="C0H500Z0"/>
         </face>
      </font>
   </fontlist>
```

```
   <codepagelist>
      <codepage name="T1000382" iana="IBM1141" mapping="unc" default="yes"/>
      <codepage name="T1V10273" iana="IBM273"  mapping="unc"/>
      <codepage name="T1V10500" iana="IBM500"  mapping="unc"/>
   </codepagelist>
   <resourcelist>
      <|-- = library ================================================ -->
      <|-- = files  ================================================
             type can be "overlay", "pagesegment", "font", “codedfont",
             "codepage" "charset", "formdef" "pagedef" or "*" extension can be
             empty, <extension> or wildcard
            ============================================================ -->
        <files   path="MVS://'SYS1.FONTLIBB("         extension=""   />
   </resourcelist>
</mffafp>
```

5.4.2. Parameter Submission with DD Statements

Under OS/390 or z/OS input and output parameter can be submitted as DD statements.

DD statements are defined within a JCL (Job Control Language). Any kind of file can be defined: HFS or MVS file or MVS spool (for output).

Example of a file definition in a JCL:

```
//OUT      DD  SYSOUT=&SYSOUT,
//             DEST=&DEST,
//             DCB=(RECFM=VBM,LRECL=12288)
```

`&DEST` is the submission parameter in the procedure. The program to be called is submitted a file name `//DD:OUT`.

Under USS the C Runtime library identifies the file. The called program finds the file name `"//DD:OUT"`. Then, the command `fopen` is used with this name.

```
fopen("//DD:OUT", "wb");
```

A correct interpretation of the name is guaranteed, if the program is called from a JCL with the program `BPXBATSL`. The alternative program `BPXBATCH` does not interpret the file names for DD statements correctly.

JCL example for a program call of cpmcopy

```
//TIF2AFP PROC INPUT=&SYSUID.1,
//             DEST=LOCAL,
//             SYSOUT=C
//*
//*****************************************************************
//*        CONVERT HFS-FILE WITH CPMCOPY                           *
//*        PRINT MCOPY LOG DATASET                                 *
//*****************************************************************
//*
//*---------------------------------------------------------------
//MCOPY   EXEC PGM=BPXBATSL,
//             PARM='PGM /usr/lpp/mc/cpmcopy -i //DD:IN -type iff.tif
//              -o MVS://DD:OUT -type afp -logdir /tmp/other'
//*---------------------------------------------------------------
//IN       DD  DISP=SHR,
//             DSN=&INPUT
//OUT      DD  SYSOUT=&SYSOUT,
//             DEST=&DEST,
//             DCB=(RECFM=VBM,LRECL=12288)
//STDOUT   DD  PATH='/tmp/other/&SYSUID..&INPUT..stdout',
//             PATHOPTS=(OCREAT,OWRONLY),
//             PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//*---------------------------------------------------------------
//STDOUT  EXEC PGM=BPXBATSL,
//             PARM='PGM /bin/cp -T //DD:STDOUT  //DD:SYSPRINT'
//STDOUT   DD  PATH='/tmp/other/&SYSUID..&INPUT..stdout',
//             PATHDISP=(DELETE,DELETE)
//SYSPRINT DD  SYSOUT=*,
//             DCB=(RECFM=VB,LRECL=132)
//        PEND
```

Notes:

- For the output operation the parameter `//DD:OUT` is prefixed with `MVS:`. Writing AFP files you must consider that each structured field must be written in one record of the output file by one single `fwrite`.
- The impact of adding `MVS:` is that after the file is opened `fwrite` writes a complete record into the file. Thus, subsequent reblocking is not necessary.
- If an output file is specified as sequential of HFS file, `MVS:` must not be prefixed. Thus, subsequent reblocking is necessary. If `MVS:` is used for a sequential file, the system issues an error messages.

Appendix A: Deprecated JavaScript Constructs

The following JavaScript constructs have been deprecated, i.e. they should not be used any more. They do, however, continue to work for the time being.

- `FileInputRaw` and `FileOutputRaw` have been replaced with `FileInputText` and `FileOutputText` for text files and `FileInputBinary` and `FileOutputBinary` for binary files, see also *chapter 4.13. File Handling on page 112*.
- The method `sys.clone()`has been replaced with a new constructor call, `v = new Vars(sys)`, see also *chapter 4.12. Vars (Varpool) on page 111*.
- The OMR method `setWhiteOutRect()` should be replaced with a `Rect`, see also *chapter 4.10.1.6. Rect on page 105*.
- The method `setPatternType()` for the `Rect` object never worked as expected and has been deprecated as of release 200707. It is currently not possible to set a fill pattern for a rectangle.
- The method `obfuscateText` for the `Page` object has been deprecated. Alternatively, you can use the methods `setInTextObfuscation` and `setTextObfuscation` instead, depending on when you want the obfuscation to happen, see also *chapter 4.2.1.17. Obfuscating Text on page 81*.
- All the Docponent items whose names end in ...Item (e.g. `TextItem`, `RectItem`, etc.) have been deprecated in favor of Docponents without the “Item” suffix (e.g. `Text`, `Rect`; exception: `ExternalItem`). New functionality will only be added to the newer items. Also make sure not to mix old and new items (e.g. creating a Text with a FontItem, as that may cause unpredictable results.
- The method `setJog` has been deprecated in favor of the new method `toggleJog`, see also *chapter 4.2.1.16. Jogging on page 80*.
- Pushing output files into the output queue (as the second parameter for `output.push`) has been deprecated. Alternatively, you use the output file’s method `write` instead to write to the file directly, see also *chapter 4.13.2. Output Queue on page 113*.
- The variants of the `FileInputDocument` and `FileOutputDocument` constructors using `context.getFileMgr()` as the first parameter have been deprecated. Alternatively, you use the constructors with the filename as the first and a `InputDocumentAttributes` or `OutputDocumentAttributes` object as the second parameter instead, see also *chapter 4.13.5. Document Attributes on page 118*.

The use of one of these deprecated constructs will usually issue a warning in the log file (with the exception of the renamed `...Items`, where this has not been implemented yet).

Appendix B: Code Page Names

The following table describes the IANA names (Internet Assigned Numbers Authority) and the Compart defined names of the code pages supported by the Compart products.

Table 15: Code Page Names (IANA)

IANA Name	Compart Names	Description
IBM00858	858	OEM Multilingual Latin 1 + Euro symbol
IBM00924	924	IBM EBCDIC Latin 1/Open System (1047 + Euro symbol)
IBM01047	1047	IBM EBCDIC Latin 1/Open System
IBM01140	1140	IBM EBCDIC US-Canada (037 + Euro symbol)
IBM01141	1141	IBM EBCDIC Germany (273 + Euro symbol)
IBM01142	1142	IBM EBCDIC Denmark-Norway (277 + Euro symbol)
IBM01143	1143	IBM EBCDIC Finland-Sweden (278 + Euro symbol)
IBM01144	1144	IBM EBCDIC Italy (280 + Euro symbol)
IBM01145	1145	IBM EBCDIC Latin America-Spain (284 + Euro symbol)
IBM01146	1146	IBM EBCDIC United Kingdom (285 + Euro symbol)
IBM01147	1147	IBM EBCDIC France (297 + Euro symbol)
IBM01148	1148	IBM EBCDIC International (500 + Euro symbol)
IBM01149	1149	IBM EBCDIC Icelandic (871 + Euro symbol)
IBM037	37	IBM EBCDIC US-Canada
IBM038	38	International EBCDIC
IBM1026	1026	IBM EBCDIC Turkish (Latin 5)
IBM273	273	IBM EBCDIC Germany
IBM274	274	IBM EBCDIC Belgium
IBM275	275	IBM EBCDIC Brazil
IBM277	277	IBM EBCDIC Denmark-Norway
IBM278	278	IBM EBCDIC Finland-Sweden
IBM280	280	IBM EBCDIC Italy
IBM281	281	IBM EBCDIC Japan (Latin)
IBM284	284	IBM EBCDIC Latin America-Spain
IBM285	285	IBM EBCDIC United Kingdom
IBM290	290	IBM EBCDIC Japanese Katakana Extended
IBM297	297	IBM EBCDIC France
IBM363	363	Symbols, Set 8 (00363)
IBM420	420	IBM EBCDIC Arabic
IBM423	423	IBM EBCDIC Greek
IBM424	424	IBM EBCDIC Hebrew
IBM437	437	IBM PC / MS-DOS
IBM500	500	IBM EBCDIC International
IBM737	737	OEM Greek
IBM775	775	OEM Baltic
IBM819	819	Western European (ISO)
IBM850	850	OEM Multilingual Latin 1
IBM851	851	Greece - Personal Computer
IBM852	852	OEM Latin 2
IBM855	855	OEM Cyrillic (primarily Russian)
IBM857	857	OEM Turkish
IBM860	860	OEM Portuguese
IBM861	861	OEM Icelandic
IBM862	862	OEM Hebrew

Table 15: Code Page Names (IANA)

IANA Name	Compart Names	Description
IBM863	863	OEM French Canadian
IBM864	864	OEM Arabic
IBM865	865	OEM Nordic
IBM866	866	OEM Russian
IBM868	868	PC Urdu (Pakistan)
IBM869	869	OEM Modern Greek
IBM870	870	IBM EBCDIC Multilingual/ROECE (Latin 2)
IBM871	871	IBM EBCDIC Icelandic
IBM880	880	IBM EBCDIC Cyrillic Russian
IBM891	891	Korea - Personal Computer (00891)
IBM903	903	People's Republic of China (PRC)-PC
IBM904	904	Taiwan - Personal Computer
IBM905	905	IBM EBCDIC Turkish
IBM918	918	IBM Pakistan (Urdu)
IBMOEDIT	CPCODEPAGE_USER_BASE + 1	IBM Open Edition
ISO_10646-1:2000	CPCODEPAGE_WINGDINGS	Wingdings
ISO-646-11	CPCODEPAGE_ISO646_11	ISO-646-11 Swedish
ISO-646-15	CPCODEPAGE_ISO646_15	ISO-646-15 Italian
ISO-646-16	CPCODEPAGE_ISO646_16	ISO-646-16 Portuguese
ISO-646-17	CPCODEPAGE_ISO646_17	ISO-646-17 Spanish
ISO-646-21	CPCODEPAGE_ISO646_21	ISO-646-21 German
ISO-646-4	CPCODEPAGE_ISO646_4	ISO-646-4 United Kingdom
ISO-646-6	CPCODEPAGE_ISO646_6	ISO-646-6 ASCII
ISO-646-60	CPCODEPAGE_ISO646_60	ISO-646-60 Norwegian
ISO-646-69	CPCODEPAGE_ISO646_69	ISO-646-69 French
ISO-646-DE	CPCODEPAGE_ISO646_21	ISO-646-21 German
ISO-646-US	CPCODEPAGE_USASCII	ISO-646-US
ISO-8859-1	CP_ISO_1	ISO 8859-1 Latin 1 Western European
ISO-8859-10	CP_ISO_10	ISO-8859-10 Latin 6 Nordic
ISO-8859-11	CP_ISO_11	ISO-8859-11 Thai
ISO-8859-12	CP_ISO_12	ISO-8859-12 (Reserved)
ISO-8859-13	CP_ISO_13	ISO-8859-13 Latin 7 Estonian
ISO-8859-14	CP_ISO_14	ISO-8859-14 Latin 8 Celtic
ISO-8859-15	CP_ISO_15	ISO-8859-15 Latin 9 Western European
ISO-8859-2	CP_ISO_2	ISO-8859-2 Latin 2 Central European
ISO-8859-3	CP_ISO_3	ISO-8859-3 Latin 3 Southern European
ISO-8859-4	CP_ISO_4	ISO-8859-4 Latin4 Baltic
ISO-8859-5	CP_ISO_5	ISO-8859-5 Cyrillic
ISO-8859-6	CP_ISO_6	ISO-8859-6 Arabic
ISO-8859-7	CP_ISO_7	ISO-8859-7 Greek
ISO-8859-8	CP_ISO_8	ISO-8859-8 Hebrew (ISO-Visual)
ISO-8859-8-I	CP_ISO_8	ISO-8859-8-I Hebrew (ISO-Logical)
ISO-8859-9	CP_ISO_9	ISO-8859-9 Latin 5 Turkish
MacRoman	CP_MAC_ROMAN	MAC Roman Western European (Mac): not IANA
MS Window Symbols	CP_CODEPAGE_WINDOWS_SYMBOL	Windows Symbols Code Page
hp-roman8	CP_HP_ROMAN	HP Roman8
roman8	CP_HP_ROMAN8	HP Roman8
UTF-16BE	CP_UTF_16BE	UTF-16 Big Endian
UTF-16LE	CP_UTF_16LE	UTF-16 Little Endian
UTF-16	CP_UTF_16	UTF-16
UTF-8	CP_UTF_8	UTF-8
windows-1250	1250	ANSI Central European (Windows)
windows-1251	1251	ANSI Cyrillic (Windows)

Table 15: Code Page Names (IANA)

IANA Name	Compart Names	Description
windows-1252	1252	ANSI Latin 1 (Windows)
windows-1253	1253	ANSI Greek (Windows)
windows-1254	1254	ANSI Turkish (Windows)
windows-1255	1255	ANSI Hebrew (Windows)
windows-1256	1256	ANSI Arabic (Windows)
windows-1257	1257	ANSI Baltic (Windows)

Appendix C: Paper Formats

The paper sizes of a printer are closely connected to the respective national standards. In the German speaking areas DIN paper formats are used as standard sizes (DIN: Deutsches Institut für Normung e.V.) The DIN standard 476 defines the series A, B, and C, Series A covers the well-known paper formats for the documents like letters or engineering drawings. Series B is often used for letterpress printing and Series C for formats like envelopes. Additionally in this collection formats are listed other than that mentioned before. **Note:** The paper format keywords listed in tables are used in MFF filter profile attributes.

Table 16: Paper Size - DIN Formats A-Series (ISO 216)

Keyword	Width	Height
4A0	168.2 cm	237.8 cm
2A0	118.9 cm	168.2 cm
A0	84.1 cm	118.9 cm
A1	59.5 cm	84.1 cm
A2	42.0 cm	59.4 cm
A3	29.7 cm	42.0 cm
A4	21.0 cm	29.7 cm
A5	14.8 cm	21.0 cm
A6	10.5 cm	14.8 cm
A7	7.4 cm	10.5 cm
A8	5.2 cm	7.4 cm
A9	3.7 cm	5.2 cm
A10	2.6 cm	3.7 cm

Table 17: Paper Size - DIN Formats B-Series (ISO 216)

Keyword	Width	Height
B0	100.0 cm	141.4 cm
B1	70.7 cm	100.0 cm
B2	50.0 cm	70.7 cm
B3	35.3 cm	50.0 cm
B4	25.0 cm	35.3 cm
B5	17.6 cm	25.0 cm
B6	12.5 cm	17.6 cm
B7	8.8 cm	12.5 cm
B8	6.2 cm	8.8 cm
B9	4.4 cm	6.2 cm
B10	3.1 cm	4.4 cm

Table 18: Paper Size - DIN Formats C-Series (ISO 216)		
Keyword	**Width**	**Height**
C0	91.7 cm	129.7 cm
C1	64.8 cm	91.7 cm
C2	45.8 cm	64.8 cm
C3	32.4 cm	45.8 cm
C4	22.9 cm	32.4 cm
C5	16.2 cm	22.9 cm
C6	11.4 cm	16.2 cm
C65	11.4 cm	22.9 cm
C7	8.1 cm	11.4 cm
C8	5.7 cm	8.1 cm
C9	4.0 cm	5.7 cm
C10	2.8 cm	4.0 cm
DL	11.0 cm	22.0 cm

Table 19: Paper Size - Raw Format A		
Keyword	**Width**	**Height**
RA0	86.0 cm	122.0 cm
RA1	61.0 cm	86.0 cm
RA2	43.0 cm	61.0 cm
RA3	30.5 cm	43.0 cm
RA4	21.5 cm	30.5 cm

Table 20: Paper Size - Supplementary Raw Format A		
Keyword	**Width**	**Height**
SRA0	90.0 cm	128.0 cm
SRA1	64.0 cm	90.0 cm
SRA2	45.0 cm	64.0 cm
SRA3	32.0 cm	45.0 cm
SRA4	22.5 cm	32.0 cm

Table 21: Paper Size - Identification Cards		
Keyword	**Width**	**Height**
ID-1	8.560 cm	5.398 cm
ID-2	10.500 cm	7.400 cm
ID-3	12.500 cm	8.800 cm

Table 22: Paper Size - North America

Keyword	Width	Height
LETTER	8.500 in	11.000 in
LEGAL	8.500 in	14.000 in
CSHEET	17.000 in	22.000 in
DSHEET	22.000 in	34.000 in
ESHEET	34.000 in	44.000 in
TABLOID	11.000 in	17.000 in
LEDGER	17.000 in	11.000 in
STATEMENT	5.500 in	8.500 in
EXECUTIVE	7.250 in	10.500 in
FOLIO	8.500 in	13.000 in
QUARTO	21.500 in	27.500 in
10X14	10.000 in	14.000 in
11X17	11.000 in	17.000 in
NOTE	8.500 in	11.000 in
ENV9	3.085 in	8.075 in
ENV10	4.125 in	9.500 in
ENV11	4.500 in	10.375 in
ENV12	4.750 in	11.000 in
ENV14	5.000 in	11.500 in
MONARCH	3.875 in	7.500 in
PERSONAL	3.625 in	6.500 in
FANFOLDUS	14.875 in	11.000 in
FANFOLDSTDGERMAN	8.500 in	12.000 in
FANFOLDLGLGERMAN	8.500 in	13.000 in

Table 23: Paper Size - Imperial

Keyword	Width	Height
IMPERIALFOLIO	22.000 in	15.000 in
IMPERIALQUARTO	15.000 in	11.000 in
IMPERIALOCTAVO	11.000 in	7.500 in
ROYALFOLIO	20.000 in	12.500 in
ROYALQUARTO	12.500 in	10.000 in
ROYALOCTAVO	10.000 in	6.250 in
CROWNFOLIO	15.000 in	10.000 in
CROWNQUARTO	10.000 in	7.500 in
CROWNOCTAVO	7.500 in	5.000 in
FOOLSCAPFOLIO	13.500 in	8.500 in
FOOLSCAPQUARTO	8.500 in	6.750 in
FOOLSCAPOCTAVO	6.750 in	4.250 in
MEDIUMQUARTO	11.500 in	9.000 in
DEMYQUARTO	11.250 in	8.750 in
DEMYOCTAVO	8.750 in	5.625 in

Table 24: Paper Size - Japanese

Keyword	Width	Height
CHOU1	14.2 cm	33.2 cm
CHOU2	11.9 cm	27.7 cm
CHOU3	12.0 cm	23.5 cm
CHOU31	10.5 cm	23.5 cm
CHOU30	9.2 cm	23.5 cm
CHOU40	9.0 cm	22.5 cm
CHOU4	9.0 cm	20.5 cm
KAKUA3	32.0 cm	44.0 cm
KAKU0	28.7 cm	38.2 cm
KAKU1	27.0 cm	38.2 cm
KAKU2	24.0 cm	33.2 cm
KAKU3	21.6 cm	27.7 cm
KAKU4	19.7 cm	26.7 cm
KAKU5	19.0 cm	24.0 cm
KAKU6	16.2 cm	22.9 cm
KAKU7	14.2 cm	20.5 cm
KAKU8	11.9 cm	19.7 cm
FURUSU10	23.5 cm	12.0 cm
YOU0	19.7 cm	13.6 cm
YOU1	17.3 cm	11.8 cm
YOU2	16.2 cm	11.4 cm
YOU3	14.8 cm	9.8 cm
YOU4	23.5 cm	10.5 cm
YOU5	21.7 cm	9.5 cm
YOU6	19.0 cm	9.8 cm
YOU7	16.5 cm	9.2 cm

Appendix D: Barcodes

The table lists the barcode types for DocBridge Mill and DocBridge Mill Tool supported by Compart. The spelling in the column “Barcode Type” corresponds to the syntax that has to be used.

Note that the barcode generation and the barcode capture is not available for all barcode types.

Table 25: Barcodes

Barcode Type	Barcode Generation	Barcode Capture
Australia Post 4-state Barcode	✔	✘
Aztec	✔	✘
Code 128	✔	✔
Code 128A	✔	✔
Code 128B	✔	✔
Code 128C	✔	✔
Code 2/5 Industrial	✘	✔
Code 2/5 Interleaved	✔	✔
Code 2/5 Matrix	✔	✔
Code 39	✔	✔
Code 39 extended	✘	✔
Code 93	✔	✔
Code 93 extended	✔	✔
Data Matrix 200	✔	✔ (captureDataMatrixBarcode only)
DataBar Expanded	✔	✘
DataBar Stacked	✔	✘
EAN 128	✔	✔
EAN 13	✔	✔
EAN 8	✔	✔
Maxicode	✔	✘
MSI	✔	✘
MSI 10	✔	✘
MSI 11	✔	✘
Onecode	✔	✘
Patchcode I	✘	✔
Patchcode II	✘	✔
Patchcode III	✘	✔
Patchcode IV	✘	✔
Patchcode T	✘	✔
Patchcode VI	✘	✔
PDF 417	✔	✘
Pharmacode	✔	✘
POSTNET	✔	✘
POSTNET 11	✔	✘
POSTNET 5	✔	✘
POSTNET 9	✔	✘
QRcode	✔	✘
Royal Mail 4 State Customer Barcode	✔	✘
United States Postal Service 4 State Barcode	✔	✘
UPC A	✘	✔
UPC E	✘	✔

Appendix E: Units of Measurements

The table below lists of measurements and the related abbreviations.

Table 26: Unit Abbreviations

Unit Abbreviation	Description
cm	Centimeter
dm	Decimeter
in	Inch
inch	Inch
m	Meter
mm	Millimeter
pt	Point
Tcm	1/1000 cm
tw	1/1440

Note: Twip (twentieth of a point)

- 1/20 point
- 1 point = 1/72 inch
- 1/1440 inch
- 1440 twips = 1 inch
- 1/567 centimeter
- 567 twips = 1 centimeter

The following conversions are valid:

- 1 tw = 1/20 pt, i.e. 10 tw = 0.5 pt
- 1 pt = 1/72 in
- 1 in = 2,54 cm
- 1 cm= 10 mm

Appendix F: Monitored Messages

The Message Observer that is integrated in DocBridge Mill monitors messages with Normal mode listed in the table below.

Table 27: Monitored Messages - Normal Mode

Message Prefix	Message number	Severity Level	Message Text
AFP	1010	WARNING	The character set 'OBJECT' could not be found. Searched dirs: 'SEARCHED DIRS'
AFP	1021	WARNING	Character 'U+...' () not found in any code page, replaced with 'U+003f'
AFP	1111	WARNING	Error reading IOCA image
AFP	1139	ERROR	Character 'U+...' not found in font '...', not mapped by mffafp.pro, and cannot be rasterized
AFP	1153	ERROR	Character for Unicode U+xxxx not found in font xy
AFP	1155	**ERROR**	Illegal FNM index 0xe0 found in charset 'C0FNT002', valid range 0x00..0x5f
AFP	1156	**ERROR**	Raster Font with pattern map and without patterns
AFP	1157	**ERROR**	Image with invalid dimension or resolution
AFP	1162	WARNING	Active AutoFormdefGeneration requires SelfContained output (mffafp.pro auto-formdefgeneration)
AFP	1163	WARNING	Active resource manager requires mostly SelfContained output (mffafp.pro useresour-cemanager)
AFP	1170	FATAL	Impossible to adjust page size according to specification limits
AFP	1171	ERROR	For overlay or macro '...' data is not available, but option '...' was given, '...' will not be written as inline resource
AFP	1175	ERROR	Found illegal tray id 0x00 in profile for tray with name 'Tray 0'
AFP	1176	ERROR	Page '1' does not have a associated medium map, printer will use the first medium map found in the formdef
AFP	1177	ERROR	The formdef 'F11' does not contain any medium map and will not be written as inline resource
AFP	1178	ERROR	The container object 'OBJLOGO' with type 'PDF' will not be written as inline re-source
AFP	1179	ERROR	The formdef 'F11' is not available and will not be written as inline resource
AFP	1180	ERROR	Medium map 'MM01' is not available and will not be written as inline resource
AFP	1181	ERROR	Character set 'C0420000' is not available and will not be written as inline resource
AFP	1182	ERROR	Code page 'T1V10500' is not available and will not be written as inline resource
AFP	1183	ERROR	Coded font 'X0420000' is not available and will not be written as inline resource
AFP	1184	ERROR	Overlay 'OVL001' is not available and will not be written as inline resource
AFP	1185	ERROR	Page segment 'SEG001' is not available and will not be written as inline resource
AFP	1186	ERROR	Object container 'OBJLOGO' is not available and will not be written as inline re-source
AFP	1187	ERROR	Processing resource 'ADA' with type 'object' is not implemented
AFP	1188	ERROR	Internal error during processing resource 'ADA' with type 'object'
APR	2042	ERROR	Only single byte character sets supported in 'CFI', ignored
APR	2054	ERROR	Found BR without mandatory resource name
APR	2055	ERROR	Illegal number of repeating groups in PGP of medium map 'S100000L', found '2' re-peating groups but '1' expected
IJP	2210	ERROR	External item '...' is not available
IPD	1000	ERROR	Font '...' can't be rasterized - text will be missing
LCD	2001	ERROR	Font '...' not found in resource directory
LCD	2002	ERROR	Form '...' not found in resource directory
LCD	2003	ERROR	Error opening form ('...')
LCD	2004	ERROR	Image ('...') not found in resource directory, tried '...'
LCD	2005	ERROR	Error opening image ('...')
LCD	2013	ERROR	Line data in line number '...' is neither LCDS nor Metacode
MMD	3001	WARNING	Form definition '...' is not found from specified paths '...'
MPR	1002	ERROR	Attribute value 'CONVERTTOTYP3' not valid for attribute 'default'
PAR	3001	WARNING	External item '...' is not available
PCL	3033	WARNING	External item '...' is not available
PDI	2007	ERROR	Unicode char '...' is not in the code page of font '...'. Replaced with '...'
PDI	2100	WARNING	External item '...' is not available

Table 27: Monitored Messages - Normal Mode

Message Prefix	Message number	Severity Level	Message Text
PDI	2101	ERROR	Page not found. Empty A4 page generated.
PDI	2106	ERROR	Not enough data for an image
PDI	2112	ERROR	XFA-based PDF not supported. Some content might be lost
PDI	2116	ERROR	Annotations array referenced, but missing from file
PDL	3002	WARNING	The requested job '...' was not defined in the JSL
POS	1000	ERROR	Missing overlay named '...' - not provided by input filter
POS	2201	ERROR	Unable to rasterize font '...' Using core font name as reference
POS	2202	ERROR	Couldn't render item of type '...'
POS	2204	ERROR	Glyph '...' of font '...' not in 'ISO-Latin 1' and 'Adobe Standard' encoding
POS	3005	ERROR	Transparent images are not supported by PostScript level 2
PSI	2007	ERROR	Error decompressing image (cx='36', cy='39') using CCITTFaxDecode filter
PSI	2014	ERROR	Error querying string width from a font (default font is specified by the PSI)
PSI	2015	ERROR	Error querying string width from an unsupported font
PXL	2020	ERROR	Page definition with media size name '...' is not found in profile, use the default media size
RMG	2001	ERROR	The medium map '...' could not be found
RMG	2008	ERROR	Resource naming collisions for 'MMINPUT'
RMG	2009	ERROR	Code page 'T1000924' is already defined, definition at file offset '882076' will be ignored
RMG	2010	ERROR	Found illegal tray id 0x00 during writing medium map 'F2000001'
RMG	3002	WARNING	The character set 'C0H20050' could not be found. Searched dirs: 'C0H20050;./res\C0H20050'
XFF	2003	ERROR	External item '...' is not available
XRX	2016	ERROR	Image data for '...' could not be read
XRX	3000	WARNING	Font '...' not found (tried '...')

The Message Observer monitors messages with Strict mode listed in the table below and the message in Normal mode.

Table 28: Monitored Messages - Strict Mode

Message Prefix	Message number	Severity Level	Message Text
AFP	1008	WARNING	The coded font '...' could not be found. Searched dirs: '...'
AFP	1009	WARNING	The code page '...' could not be found. Searched dirs: '...'
AFP	1019	WARNING	Font '...', size='...', weight='...', style='...' not found, will be replaced with standard '...'
LCD	1002	FATAL	PDL File '...' not found in resource directory, tried '...' (to be replaced with 2012)
LCD	2012	ERROR	PDL File '...' not found in resource directory, tried '...'
MCF	1174	ERROR	Error copying comments of input page '...'
PCL	2074	ERROR	The char code '...' is mapped to Unicode U+... in the symbol set '...', use original char code as Unicode
PCL	3029	WARNING	Font file '...' specified in profile not found, tried ('...')
PDI	2005	WARNING	Font '...' (weight '...' style '...' replaced with '...')
PDI	2012	ERROR	Font file '...' not found
PDI	2033	WARNING	Font '...' has no fontfile. Using '...' ('...') instead
PDI	2099	ERROR	No glyph data for Unicode '...' in font '...'
PDL	2004	**ERROR**	**PDL command '...' with parameter(s) '...' could not be processed**
PDL	3002	WARNING	The requested job '...' was not defined in the JSL
XRX	2022	ERROR	Error using encoding for font '...' with character Unicode '...' - using new font '...' instead

Appendix G: Frequently Asked Questions – FAQ

G.1. FAQ 1: Empty Lines in CSV Files

Question: How do you treat empty lines in CSV files?

Initial situation: In the cpmill profile the Cluster (CLU) driver, see also *chapter 5.2.2. cpmill Profile Drivers on page 222*, typically uses a `lineextract` section like the one below.

```
<lineextract>
   <filename>
      <script>
         $ROOT_DIR + "/input/" + FILENAME
      </script>
   </filename>

   <pagefrom>
      <script>
          PAGEFROM
      </script>
   </pagefrom>

   <pageto>
      <script>
          PAGETO
      </script>
   </pageto>
</lineextract>
```

In the example above, `PAGEFROM`, `PAGETO`, and `FILENAME` are the names of columns in the CSV file that contain the start page, end page, and input file name, respectively.

Problem: This all works as expected as long as there are not any empty lines in the CSV file. For an empty line, you will encounter messages like these:

```
2009-06-17 17:04:58 CSV3001W /.../csv1.csv(2:1): Unexpected end of line, column 'CITY' truncated
2009-06-17 17:04:58 UNT1140E An error occurred when evaluating a script formula '
        PAGEFROM
    ...' - please have corresponding profile <script> section revised, base error code '8'
```

In this case, `CITY` was the second column in the CSV file. An empty line is interpreted as an empty value for the first column, and then the CSV parser will warn about missing values for the following columns.

When it comes to interpreting the formula for `PAGEFROM`, the formula parser will abort the processing since the variable `PAGEFROM` is not defined.

Answer: There are two steps to solve the problem:

- You need to introduce a `<skipcondition>` to tell cpmill to skip the empty line
- You need to work around the problem of the undefined variable, as the skip condition is evaluated after the other sections.

```
<lineextract>
    <filename>
        <script>
            IF(EXISTS(FILENAME), $ROOT_DIR + "/input/" + VARVALUE(FILENAME), "")
        </script>
    </filename>

    <pagefrom>
        <script>
            IF(EXISTS(PAGEFROM), "" + VARVALUE(PAGEFROM), "")
        </script>
    </pagefrom>

    <pageto>
        <script>
            IF(EXISTS(PAGETO), "" + VARVALUE(PAGETO), "")
        </script>
    </pageto>

     <skipcondition>
        <script>
            IF(ZIP=="", "TRUE", "")
        </script>
    </skipcondition>
</lineextract>
```

The Skip Condition

In our example, `ZIP` is the name of the first column. As explained above, it will be empty (but defined) for an empty line. So we use this for the skip condition: If `ZIP` is empty, return `"TRUE"` (any non-empty string will do), otherwise return an empty string.

Checking for Undefined Variables

The check for undefined variables requires two steps:

- You must use `EXISTS(variable)` to check that the variable is defined
- When it is defined, we still need to wrap it in `VARVALUE(variable)`, as the entire expression will be evaluated even when it's undefined, and so any further use of the variable name would throw an error again, unless it's wrapped in `VARVALUE`.

And finally, you will need to concatenate the result of `VARVALUE` with an empty string to prevent another error being thrown.

G.2. FAQ 2: Processing of Multiple cpmill Units

Question: How can cpmill handle the processing of two or more units?

Answer: The normal definition of a unit in the cpmill profile looks like this:

```
<unit key="MyUnit" ... disabled="true">
```

To process one unit you start cpmill as follows:

```
cpmill -uMyUnit myprofile.pro
```

You change the value of the attribute `disable` to `false`. Then, you can start cpmill with the parameter `-u`.

```
cpmill myprofile.pro
```

Then, cpmill will execute **any** units and tasks it finds that are not disabled.

So, if you want to run two or more units, you simply enable them all and they will be executed in the order in which they are defined in the profile.

Index