

Oracle® Documaker

Printers Reference

version 12.3

Part number: E51709-01

March 2014

Copyright © [2010], [2014], Oracle and/or its affiliates. All rights reserved. This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are “commercial computer software” or “commercial technical data” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Where an Oracle offering includes third party content or software, we may be required to include related notices. For information on third party notices and the software and related documentation in connection with which they need to be included, please contact the attorney from the Development and Strategic Initiatives Legal Group that supports the development team for the Oracle offering. Contact information can be found on the Attorney Contact Chart.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

THIRD PARTY SOFTWARE NOTICES

This product includes software developed by Apache Software Foundation (<http://www.apache.org/>).

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2000-2009 The Apache Software Foundation. All rights reserved.

This product includes software distributed via the Berkeley Software Distribution (BSD) and licensed for binary distribution under the Generic BSD license.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2009, Berkeley Software Distribution (BSD)

This product includes software developed by the JDOM Project (<http://www.jdom.org/>).

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved.

This product includes software developed by the Massachusetts Institute of Technology (MIT).

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright © 2009 MIT

This product includes software developed by Jean-loup Gailly and Mark Adler. This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Copyright (c) 1995-2005 Jean-loup Gailly and Mark Adler

This software is based in part on the work of the Independent JPEG Group (<http://www.ijg.org/>).

This product includes software developed by the Dojo Foundation (<http://dojotoolkit.org>).

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2005-2009, The Dojo Foundation. All rights reserved.

This product includes software developed by W3C.

Copyright © 2009 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. (<http://www.w3.org/Consortium/Legal/>)

This product includes software developed by Mathew R. Miller (<http://www.bluecreststudios.com>).

Copyright (c) 1999-2002 ComputerSmarts. All rights reserved.

This product includes software developed by Shaun Wilde and distributed via Code Project Open License (<http://www.codeproject.com>).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by Chris Maunder and distributed via Code Project Open License (<http://www.codeproject.com>).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by PJ Arends and distributed via Code Project Open License (<http://www.codeproject.com>).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by Erwin Tratar. This source code and all accompanying material is copyright (c) 1998-1999 Erwin Tratar. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY. USE IT AT YOUR OWN RISK! THE AUTHOR ACCEPTS NO LIABILITY FOR ANY DAMAGE/LOSS OF BUSINESS THAT THIS PRODUCT MAY CAUSE.

This product includes software developed by Sam Leffler of Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE

Copyright (c) 1988-1997 Sam Leffler

Copyright (c) 1991-1997 Silicon Graphics, Inc.

This product includes software developed by Guy Eric Schalnat, Andreas Dilger, Glenn Randers-Pehrson (current maintainer), and others. (<http://www.libpng.org>)

The PNG Reference Library is supplied "AS IS". The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

This product includes software components distributed by the Cryptix Foundation.

THIS SOFTWARE IS PROVIDED BY THE CRYPTIX FOUNDATION LIMITED AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE CRYPTIX FOUNDATION LIMITED OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

Copyright © 1995-2005 The Cryptix Foundation Limited. All rights reserved.

This product includes software components distributed by Sun Microsystems.

This software is provided "AS IS," without a warranty of any kind. ALLEXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Copyright (c) 1998 Sun Microsystems, Inc. All Rights Reserved.

This product includes software components distributed by Dennis M. Sosnoski.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2003-2007 Dennis M. Sosnoski. All Rights Reserved

It also includes materials licensed under Apache 1.1 and the following XPP3 license

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2002 Extreme! Lab, Indiana University. All Rights Reserved

This product includes software components distributed by CodeProject. This software contains material that is © 1994-2005 The Ultimate Toolbox, all rights reserved.

This product includes software components distributed by Geir Landro.

Copyright © 2001-2003 Geir Landro (drop@destroydrop.com) JavaScript Tree - [www.destroydrop.com/hjjavascripts/tree/version 0.96](http://www.destroydrop.com/hjjavascripts/tree/version0.96)

This product includes software components distributed by the Hypersonic SQL Group.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

Copyright © 1995-2000 by the Hypersonic SQL Group. All Rights Reserved

This product includes software components distributed by the International Business Machines Corporation and others.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright (c) 1995-2009 International Business Machines Corporation and others. All rights reserved.

This product includes software components distributed by the University of Coimbra.

University of Coimbra distributes this software in the hope that it will be useful but DISCLAIMS ALL WARRANTIES WITH REGARD TO IT, including all implied warranties of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. In no event shall University of Coimbra be liable for any special, indirect or consequential damages (or any damages whatsoever) resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

Copyright (c) 2000 University of Coimbra, Portugal. All Rights Reserved.

This product includes software components distributed by Steve Souza.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2002, Steve Souza (admin@jamonapi.com). All Rights Reserved.

This product includes software developed by the OpenSymphony Group (<http://www.opensymphony.com/>.)"

Copyright © 2001-2004 The OpenSymphony Group. All Rights Reserved.

PANTONE (R) Colors displayed in the software application or in the user documentation may not match PANTONE-identified standards. Consult current PANTONE Color Publications for accurate color. PANTONE(R) and other Pantone LLC trademarks are the property of Pantone LLC. (C) Pantone LLC, 2011.

Pantone LLC is the copyright owner of color data and/or software which are licensed to Oracle to distribute for use only in combination with Oracle Documaker. PANTONE Color Data and/or Software shall not be copied onto another disk or into memory unless part of the execution of Oracle Documaker.

CONTENTS

Preface

- Audience
- Documentation Accessibility
- Accessibility of Links to External Web Sites in Documentation
- TTY Access to Oracle Support Services
- Related Documents
- Conventions

Documaker Desktop Print Scenarios

Overview

- Task Flow

Configuring Print Operations

- Operator-Initiated Printing
- System-initiated Printing
- Customizing the Printing Process
- Batch Printing Options

Using the AFP Print Driver

Overview

Setting AFP INI Options

- Using Defaults for the Module and PrintFunc Options
- Using Documaker Shading Patterns Instead of Shaded Bitmaps
- Printing Highlight Colors
- Character Set and Code Page Font Information
- Outputting Character Set and Code Page Information
- Using Multiple Code Pages
- Using the CODED.FNT File
- Using the CPDEF.FNT File
- Using the CPGID.CP (Code Page Map) File

Using LLE Records to Link to External Documents
Adding Data to Begin Page and End Page Records

Defining AFP Printer Resources

FormDef
Fonts
System Fonts
Overlays
Page Segments
AFP 2-up Support

Including Documerge Form-level Comment Records

AFP Troubleshooting

Floating Section Limitations
Objects Extending Beyond the Edges
Conflicts Between Page and Form Orientation
Multipage FAP Limitation
Printing rotated variable Fields
AFP 240 DPI Print Problems

Using the Bitmap Print Driver

Overview

Prerequisites

Setting Up INI Files

Documaker Desktop and Documaker Studio INI Options
Documaker Server INI Options

Choosing Fonts

Working with Color

Forcing Color Output

Handling Multiple Page Form Sets

- For Documaker Studio and Documaker Desktop
 - Creating a TIFF File for Each Page of a Form Set
 - Creating a Single TIFF File for Each Form Set
- For Documaker Server
 - Generating Unique File Names

Selecting the Bitmap Print Driver

Additional Considerations

Using the EPT Print Driver

Overview

Creating EPT Print Files for Documaker Desktop

- Using the RecipMod and RecipFunc Options

Creating EPT Print Files for Documaker Server

Creating PDF Print Files

Sending Emails in Multipart MIME Format

- Examples

Including Attachments with MPM Files

- Including Attachments with MPM Files

Overriding Attached Files

Using Email Aliases

Using XML Print Driver

Using the GDI Print Driver

Overview

- How It Works

Setting GDI INI Options

- Using Defaults for the Module and PrintFunc Options

Avoiding Problems with FAX Drivers

Batch Printing to Files

Using the Metacode Print Driver

Overview

Setting the Required JSL INI Options

JDLName

JDEName

DJDEIden, DJDEOffset, and DJDESkip

JDLCode

JDLData

JDLHost

Additional Required INI Options

OutMode

ImageOpt

CompressMode

CompileInStream

Device

RelativeScan

Specifying Installable Functions

Using Defaults for the Module and PrintFunc Options

Optional INI Options

Setting the End of the Report

Starting New Pages

Adding an OFFSET Command

Jogging Pages

Specifying Spot Color

Chart Performance and Print Quality

Optimizing Metacode Print Streams

Using a Common Font List

Setting a Default Paper Size

Automatically Sizing Sections

Inline Graphic Performance and Print Quality

Adding Color to Charts

Using Named Paper Trays

Specifying the Printer Model

Specifying the Resolution

- Displaying Console Messages
- Stapling Forms
- Duplex Switching
- Using VSAM to Store Resources
- PrintViewOnly
- Caching Files to Improve Performance
- Using the Loader
- Using the Class Option
- Adding User-Defined DJDE Statements
- Using Third-Party Software to Read Metacode Files
- Specifying the Paper Stock

Using Mobius Metacode Print Streams

Metacode Printer Resources

- Fonts
- Forms
- Images
- Logos

Metacode Limitations

- Xerox Images
- HMI Support
- Changing the Paper Size on the 4235 Printer
- Xerox Forms

Metacode Troubleshooting

- Unexpected Color Output
- Unexpected Black and White Output
- Highlight Color Should Match the PrinterInk Option
- LOG File Orientation
- Output Catching Up with the Input
- Printing Rotated Variables
- Multipage Sections
- Operator Command, FEED, Causes Duplex Problems
- Line Density Errors
- Output Data Length Validation
- Using Xerox Forms (FRMs)

BARRWRAP

Transferring Files from Xerox Format Floppies

Using the MPM Print Driver

Setting MPM INI Options

Example MPM Document

Designing Forms

Using the PCL Print Driver

Overview

Setting PCL INI Options

Using Defaults for the Module and PrintFunc Options

Using PCL 6

Printing Under Windows

Using High-Capacity Trays 3 and 4 on HP 5SI Printers

Using a Staple Attachment

Overriding Paper Size Commands and Tray Selections

Using Simple Color Mode

Marking Objects to Print in Color

Specifying the Highlight Color to Use

Printing on Different Types of Printers

Creating Compressed PCL Files

Bitmap Compression

Adding Printer Job Level Comments

Adding Data for Imaging Systems

Limiting the Number of Embedded PCL Fonts

Defining PCL Printer Resources

Fonts

Overlays

Using the PDF Print Driver

Overview

- PDF Versions

Setting Up the PDF Print Driver

- Setting PDF Options

- Additional Feature Setup

 - Emailing PDF Files

 - Using the PDF Print Driver with GenPrint

 - Changing the GenPrint Program

 - Setting the CheckNextRecip Option

 - Using Overlays

 - Using the MultiFilePrint Callback function

 - Scaling and Cropping Graphics

 - Using the Log File

 - Caching Fonts

 - Generating Separate PDF Files

 - Generating a Single PDF File

- Limitations

- Paper Sizes

Customizing PDF Output

- Producing Optimal PDF Output

- Reducing PDF File Sizes

 - Setting PDF Compression Options

 - Using JPEG Compression

 - Using Compact Fonts

- Setting PDF Viewer Preferences

- Creating Linearized PDF Files

- Setting Up Bookmarks

 - Creating Custom Bookmarks

 - FAPGetExtraInfo

 - FAPPutExtraInfo

- Adding Hypertext Links

- Adding Digital Signature Placeholders

- Forcing the PDF Driver to Print Color Images

- Meeting the PDF for Archive Specification
- Emulating Duplex Printing from the PDF Print Driver
- Interfacing with Imaging Systems
- Creating Section 508-compliant PDF Files
- Examples

Working With Fonts

- Using the Base Fonts
- Using OpenType® Fonts
- Embedding Fonts
 - Not Embedding Fonts
 - Using Embedded Fonts
 - Scaling Embedded Fonts
- Subsetting Fonts
- Handling Fonts with Multiple Width Tables
- Using Font Cross Reference Files

Adding Security to PDF Files

- Configuring the Security Features
 - Configuring the INI Files
- Using the PDFKEY Utility
- Using the PDFKEYGEN Function
- Using AES Encryption
- Example Security Settings
- Tips

Using the PostScript Print Driver

Overview

Setting PostScript INI Options

- Using Defaults for the Module and PrintFunc Options
- Avoiding a White Outline Around Letters
- Printing Under Windows
- Generating PostScript Files on z/OS
- Creating Smaller PostScript Output

Bitmap Compression

Adding DSC Comments

Stapling Forms

Using PostScript Printer Resources

Fonts

Overlays

True Type fonts within a Postscript print stream

PostScript Printer Definition (PPD) Files

Using the RTF Print Driver

Overview

Setting RTF INI Options

Generating Separate Files

Adding or Removing Frames

Creating Form Fields

Setting Margins

Removing the Contents of Headers and Footers

Working with the Documaker Add-In

Using the VIPP Print Driver

Overview

Using VIPP Resource Files

Converting Bitmaps into VIPP Image Files

Converting FAP files into VIPP Segment Files

VIPP Fonts

VIPP Font Encoding Files

Managing VIPP Resources

Setting VIPP INI Options

Setting Up Folders and Projects

Overriding the List of Libraries for Projects

Setting up Paper Trays

Adding DSC Comments

VIPP Limitations

VIPP Troubleshooting

Scenario 1

Scenario 2

Scenario 3

Creating HTML Files

Setting Up the HTML Print Driver

Creating Separate HTML Files for each Transaction

Producing Table Information for TextMerge Paragraphs

Appendix A: Choosing a Paper Size

US Standard Sizes

ISO Sizes

ISO A Sizes

ISO B Sizes

ISO C Sizes

Japanese Standard Sizes

Printer Support for Paper Sizes

Paper Sizes for AFP Printers

Appendix B: Miscellaneous Print Topics

Using Pass-through Printing

Printing with Missing Graphics

Creating Print Streams for Docusave

Archiving AFP Print Streams

Archiving Metacode Print Streams

Archiving PCL Print Streams

Using DAL Functions

Adding TLE Records

Handling Multiple Paper Trays

For PCL Printers

For PostScript Printers

For GDI Printers

For AFP Printers

For Metacode Printers

Including Tray Selections in a Print Stream Batch

Spot Color Support

Index

Preface

Oracle Documaker is a powerful, adaptive enterprise document automation platform used worldwide to acquire, create, manage, and present structured, on-demand, and interactive customer communications. It is designed to put power in the hands of business users, giving them the flexibility to create interactive, dynamic documents on demand.

Spanning the entire business life cycle, Oracle Documaker helps you manage customer communications enterprise wide—including document production, correspondence, and cross-selling campaigns—across all locations and lines of business. The industry-leading platform offers a cost-effective way to address the design, production, and multichannel distribution of a broad spectrum of customer-facing documents. With robust functionality and cutting-edge technical capabilities, it maximizes efficiencies, ensures compliance, and enhances customer service.

Oracle Documaker is based on open standards and integrates easily into today's service-oriented architecture environments. It integrates with any type of system across the enterprise. It can even be integrated with your self-service Web portal so stakeholders can get immediate access to up-to-date information. Oracle Documaker provides the agility and flexibility you need to roll out new products quickly and remain competitive.

Oracle offers proven tools and migration methods, along with experienced, highly trained technical personnel to ease conversions while maintaining the intelligence of your data. Leveraging Oracle Documaker as a single system can dramatically reduce costs. One insurance customer recouped the full cost of an Oracle Documaker implementation within nine months.

Business users can easily author content in Oracle Documaker Studio using Microsoft Word through a plug-in that leverages the power of Documaker Studio in the background. For even more capability, Oracle Documaker's intuitive, easy-to-use design tool, Documaker Studio, empowers business users to create powerful, persuasive content minimizing their reliance on IT, so you can produce dynamic, *intelligent* transactional documents that transmit data and content.

AUDIENCE

This document is intended as a resource for the person who will set up Oracle Documaker and Documaker Desktop to generate output files.

DOCUMENTATION ACCESSIBILITY

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

RELATED DOCUMENTS

For more information, refer to the following Oracle resources:

- Documaker Installation Guide
- Documaker Administration Guide
- Documaker Studio User Guide
- Rules Reference
- DAL Reference
- Fonts Reference
- Unicode Reference
- Utilities Reference
- Documaker Desktop Installation Guide
- Documaker Desktop Administration Guide

CONVENTIONS

The following text conventions are used in this document:

Convention	Description
bold	Indicates information you enter.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands, URLs, code in examples, and text that appears on the screen.

Chapter 1

Documaker Desktop Print Scenarios

This chapter discusses print scenarios for Documaker Desktop and includes these topics:

- *Overview* on page 24
- *Configuring Print Operations* on page 26
- *Task Flow* on page 25
- *Customizing the Printing Process* on page 30
- *Batch Printing Options* on page 33

OVERVIEW

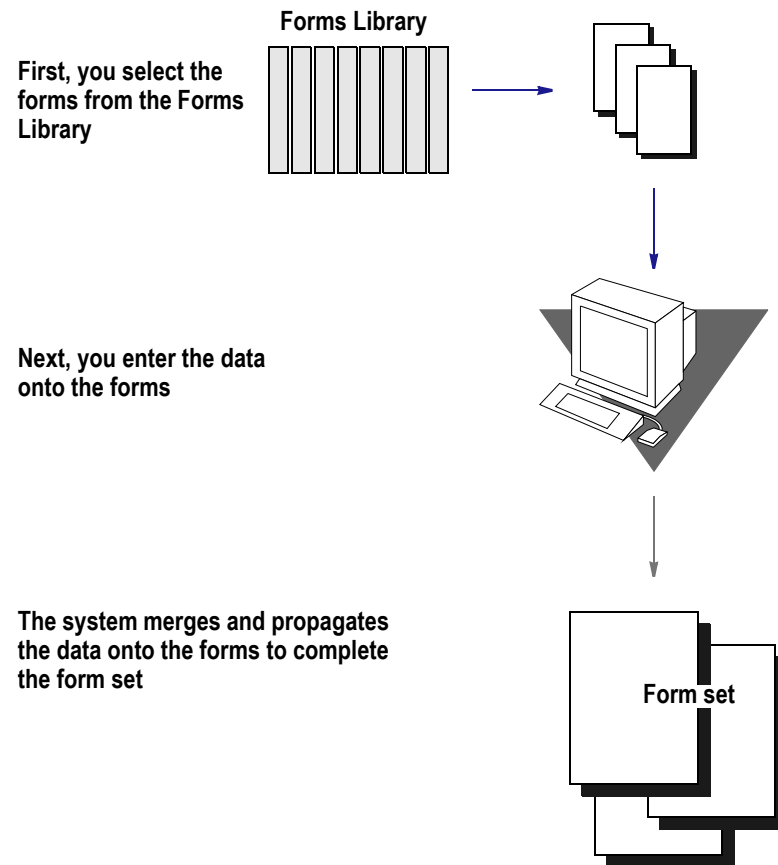
Documaker Desktop is the workstation-based form set entry and publishing piece of the Documaker system. Documaker (the rules-based publishing system) is a total form set automation system which enables forms-intensive industries such as insurance, finance, utilities, and government to automate enterprise-wide forms and forms processing.

You collect the information you need for your form sets from various sources including manually entered data, system default data, archived data, and data extracted from external application systems. Documaker Desktop lets you enter that information and print complete, collated form sets on laser printers.

The system's unique data import and export feature enhances the data entry process. The system lets you import data files that automatically fill or propagate the data onto a form's fields. Exporting lets you extract data from Documaker Desktop for use in other applications or for import back into the system.

The system's user interface makes data entry and forms processing easy for non-technical users. You enter basic information and Documaker Desktop displays a list of applicable forms. You then select specific form sets in which to enter data.

This illustration shows the process:

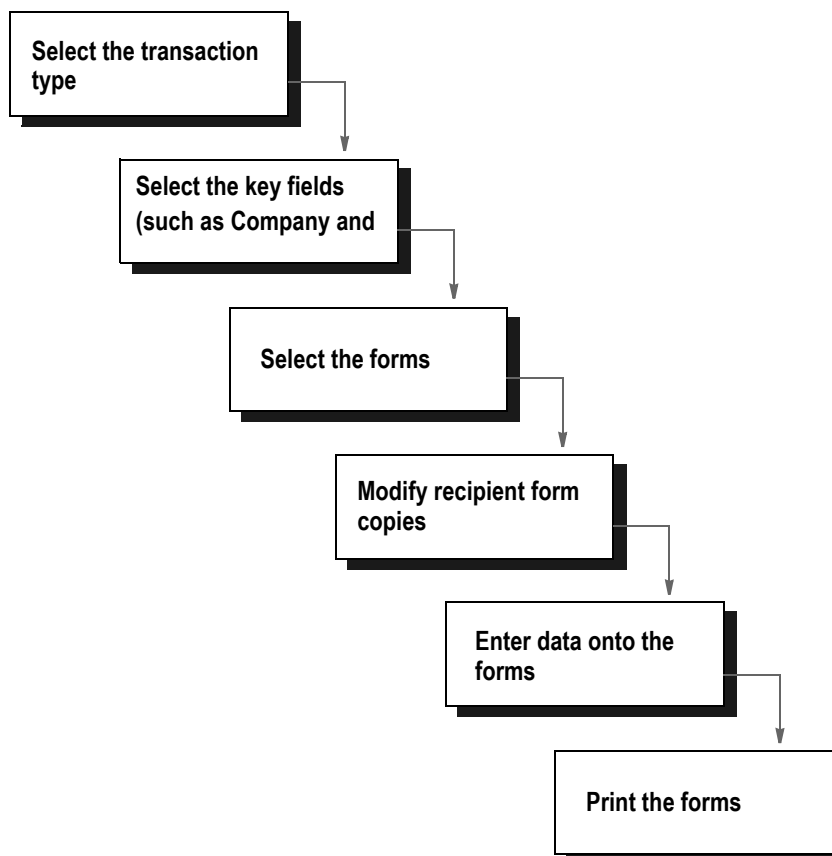


The system lets you import data from an internal or an external source and apply the data to specific forms. You can also export system data to use in other application systems or to import back into the system.

TASK FLOW

From the system's main window, you choose File, New and enter the key field information, such as selecting a company and corresponding lines of business, for a particular form set. The system displays the list of forms applicable for that key field data.

You can then change the number of recipient copies in the forms list. Next, you enter the required variable data into each form. If multiple forms require the same field data, the system automatically propagates the data into subsequent forms.



You can print a form set once you complete data entry, or send a form set to a batch queue for printing later. When you print form sets, the system merges the data with the appropriate form *template* and prints the form set. The system then archives the data and the form set template separately.

For detailed information on using Documaker Desktop's many features, please refer [Documaker Desktop User Guide](#).

CONFIGURING PRINT OPERATIONS

Whether your company operates using single user PC workstations, Local Area Network (LAN) or Wide Area Network (WAN), client-server systems, mainframe systems or a combination system, the system can handle the demands of your print workload. Users can send immediate print jobs, defer printing until a print batch is created, or distribute print jobs among various printers and/or operating systems.

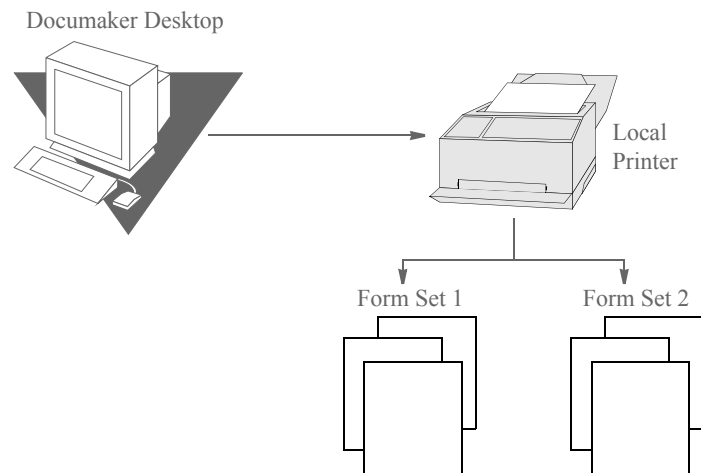
Workstation and mainframe systems process print transactions differently. If your company operates using a workstation system, the system may interact directly with the system print manager. When you send a print job from a workstation, the documents either print immediately or are redirected to a spool file, printing when the printer becomes available. If your company operates using a mainframe as a host, the print job writes directly to a spooler and prints when the printer becomes available.

Operator-Initiated Printing

Operator-initiated printing lets you submit print jobs to the system. Documaker Desktop lets you immediately print a form set or defer the print job to include multiple form sets printed in a batch. The system supports multiple printers, so you can distribute print jobs to different printers. The system also supports distributed operating environments; one user workstation may operate under Windows Vista, while another may operate under Windows XP.

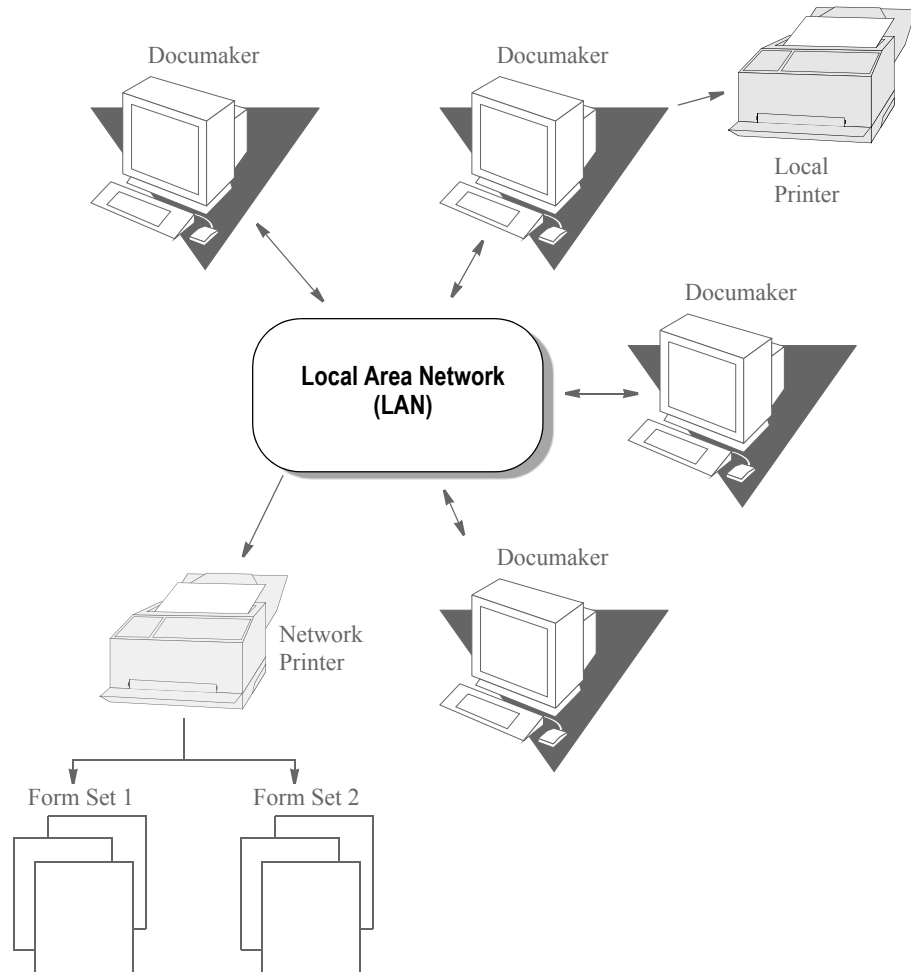
Local printing

Printing form sets locally lets users print while operating on single user workstations. For local printing, normally you send a print job to the specific printer connected to your workstation. Local printing produces immediate results; you do not have to wait for the print job to queue up. You can print work-in-process and archived form sets for review, or you can print a form set on which you have just completed data entry. When you print a form set locally, you print a copy of the form set for each predefined recipient of the form set.



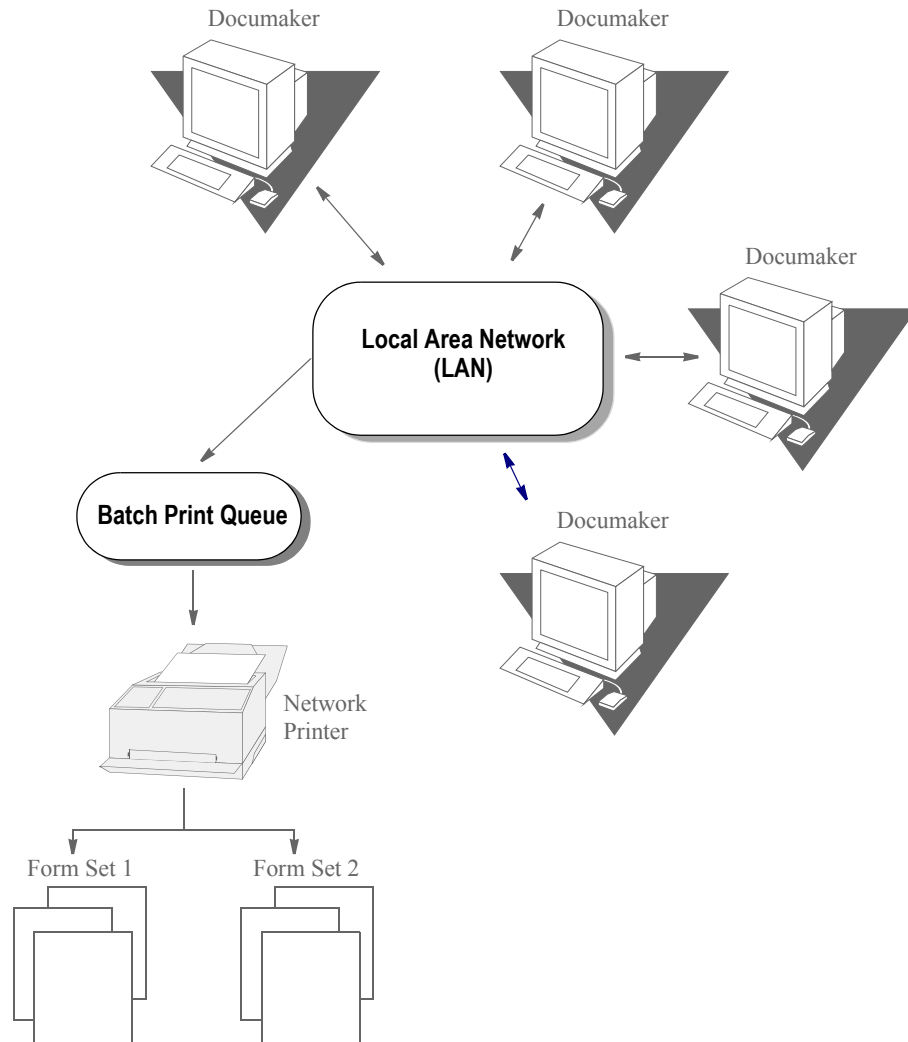
Network printing

Users can also send immediate print jobs via the network. Through a local network users select a printer on which to print form sets. The system lets you combine local and network printing options on your system.



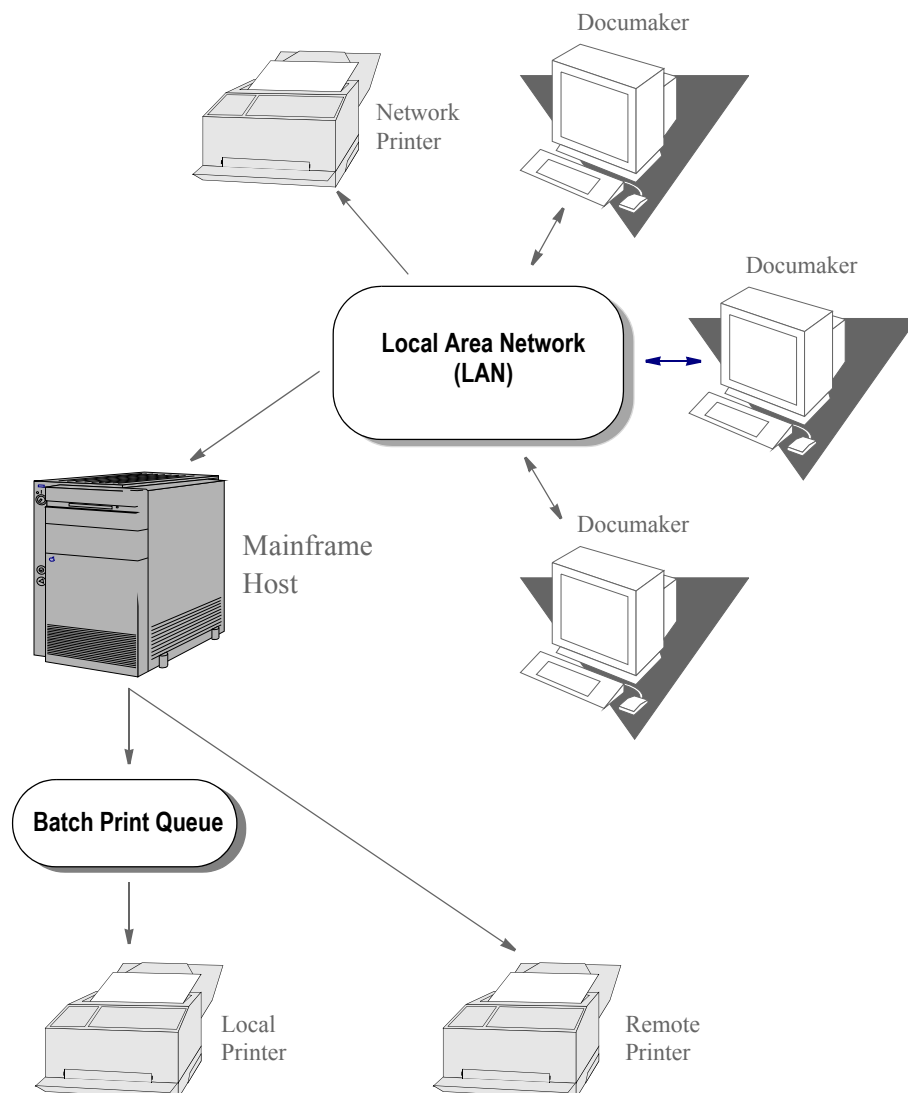
Deferred printing

Deferred printing lets you group multiple documents in a batch to print at one time. When you perform a batch print, you group documents and place them in print queues. All documents in this batch are queued in the order in which you add them to the queue. From a batch queue, you can print a single form set, print multiple form sets, or print the entire batch. The form sets remain in the batch queue until you print them.



Distributed printing

Distributed printing lets you select from various printers when you print form sets. For example, your company may use both a LAN system with workstations running various operating systems, as well as a mainframe host system. You can run your print job on the host and print on a local, a remote, or a network printer. When you send print jobs, either single form sets or batched form sets, you can distribute print jobs using several printers.



System-initiated Printing

System-initiated printing lets the system supervisor set up the system to automatically print large job batches. System-initiated printing is sometimes called *back end* batch printing because users do not work with the system's menus.

System-initiated automatic batch printing is handled exclusively by Documaker and other predefined print routines. Users do not manually enter data to process a system-initiated batch print. The data is gathered from an external database, then merged with the appropriate forms (sections). The form sets are collated, sorted, and printed in a particular sort order.

System-initiated print jobs are almost always batch print jobs. You can set up your system to distribute different print batches to different printers.

Local or network printing

Like operator-initiated local printing, system-initiated local printing lets you set up the system to process batch print jobs on local and/or networked printers.

Deferred printing

System-initiated print jobs let you choose to run a job during business hours or defer the print job until after hours. For companies producing large numbers of paper documents, running print jobs at night does not tie up the printers during working hours and maximizes printing efficiency.

Distributed batch printing

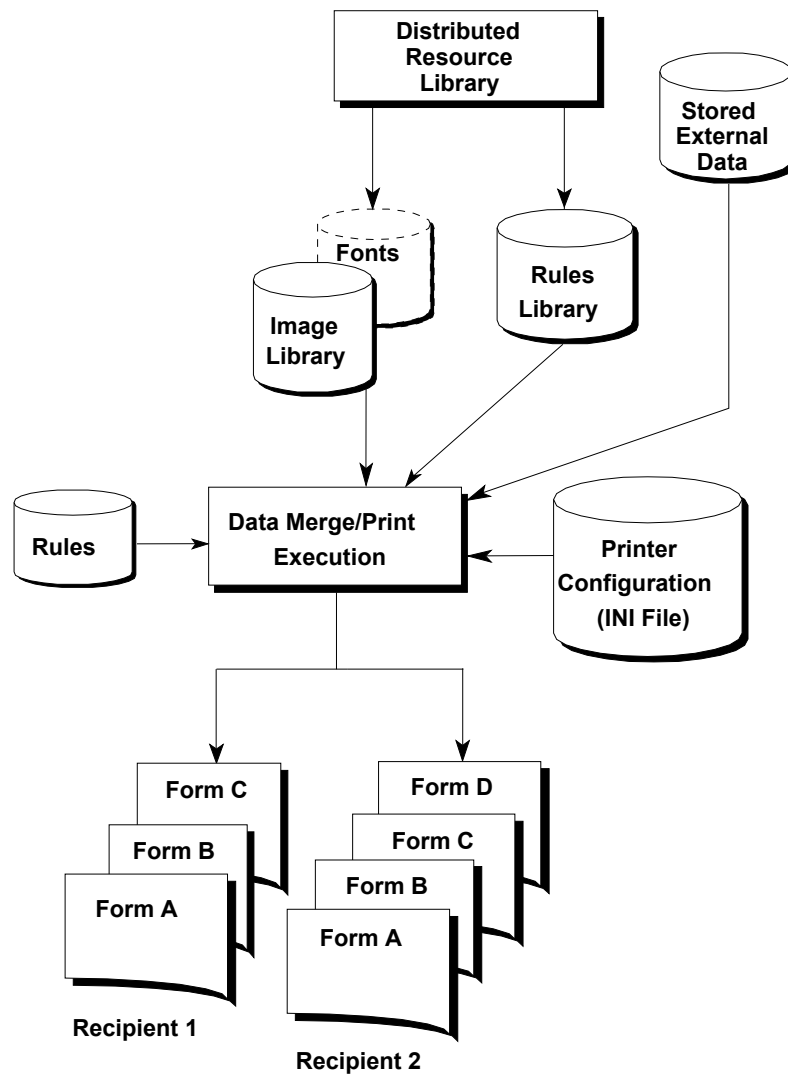
You can custom configure your system to print different batches on different printers. Just as you can distribute immediate print jobs among different printers and operating systems, you can distribute automatic batch print jobs.

CUSTOMIZING THE PRINTING PROCESS

You can customize your company's print configuration. The configuration determines how the system formats and processes the output (forms and sections) from Documaker. Your specific configuration also determines printing options and how you use printing features.

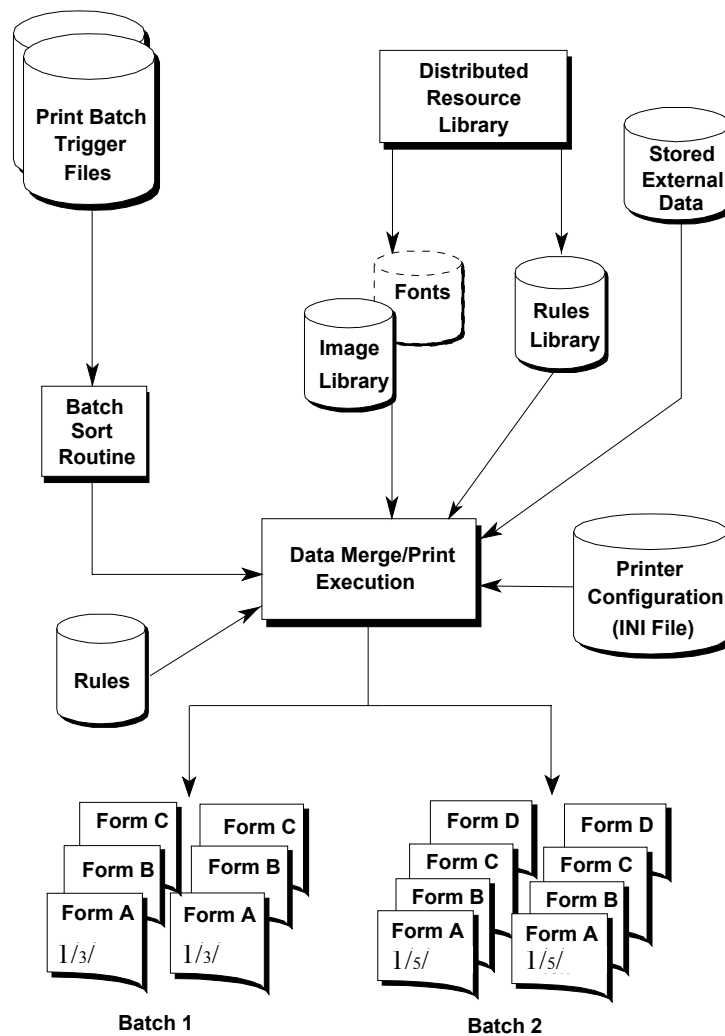
The following pages contain illustrations which show printing workflows and provide examples supporting the illustrations.

Immediate print transaction flow



When you select either single form sets or batched form sets for immediate print, your system extracts the appropriate information from three separate files: the section file, the recipient file, and the data file. The print programs merge the appropriate data into the appropriate sections, and combine the merged data with the printer configuration data.

The resulting output is the printed hard copy form set for each particular recipient. Form sets and recipients are predefined, so a recipient can receive a form set that differs both in form number and data type from another recipient's form set.

Deferred (batch) print workflow

When you perform a batch print, your sort order must be configured in the system INI file before you print your forms and documents. The print trigger batch files *mark* where the data for a specific batch begins and ends.

A system-initiated print configuration contains at least one print batch trigger file that corresponds to each printed document batch. The batch print function provides you the opportunity to configure the printer designations and sort routines to produce a variety of batch print options.

BATCH PRINTING OPTIONS

Your system supervisor can configure deferred batch print sort routines to print document batches by recipient. For example, suppose you are an administrator at a large insurance company. Once a year, the state audits all insurance policies generated during the past calendar year. You have implemented the printing features to quickly produce the required hard copy policies in a batch print. You have set up the appropriate files so that your print results in the following three batches:

- All branch office policy copies
- All agent policy copies
- All insured policy copies

The system collates each batch, so that all documents within a form set print in the sort order you specified. You defined your sort order by policy date, and your company generated 500 policies during the past year. Each batch begins with all forms for the policy dated 1/3/11, followed by all forms for the policy dated 1/4/11, and so on.

Batch print collation is a great benefit when dealing with a large number of documents. When you print a batch, you save countless manual resources and time spent sorting the copies by hand.

Chapter 2

Using the AFP Print Driver

This chapter provides information about using Documaker's AFP Print Driver. This print driver produces AFP data streams (AFPDS).

This chapter includes the following topics:

- *Overview* on page 36
- *Setting AFP INI Options* on page 37
- *Defining AFP Printer Resources* on page 49
- *Including Documerge Form-level Comment Records* on page 51
- *AFP Troubleshooting* on page 52

OVERVIEW

IBM created the Advanced Function Printing (AFP) language. The data streams produced by Documaker applications for AFP printers are called *Mixed Object Document Content Architecture* (MO:DCA) data streams. MO:DCA data streams are sometimes referred to as AFP data streams (AFPDS).

You must have a program such as IBM's Print Services Facility (PSF) to convert AFP data stream into the printer's native language. PSF is the umbrella software that brings the AFP resources (created by AFP or system utilities) together in one print job and sends it to the printer.

Note All system print drivers support 24-bit color graphics. If your printer does not support color, the print driver will automatically convert the color graphics into monochrome graphics. Keep in mind that for the best performance you should avoid color graphics.

SETTING AFP INI OPTIONS

You define the necessary printer options for the system to produce AFP data streams. These options specify how the system creates AFP output. Most of the AFP-related options are found in a `PrtType:XXX` control group, where *XXX* indicates the different printer types. *PrtType:AFP* is a common control group name used to contain AFP settings. The most common AFP printer options are shown below (default values are bold):

Option	Values	Description
Device	Any file or device name	The name of the file or device (LPT1) where the AFP data stream should be written. This setting is ignored by the GenPrint program but is used by Documaker Studio and other system programs.
Module	AFPPRT	The name of the program module which contains the system's AFP print driver. See also the discussion of the Class option. See also <i>Formdef = F1DOCUMK</i> : on page 40.
PrintFunc	AFPPrint	The name of the program function that is the main entry point into the system's AFP print driver. See also <i>Formdef = F1DOCUMK</i> : on page 40.
Resolution	240/300	The dots per inch (dpi) resolution of the printer which receives the AFP data stream.
SendOverlays	Yes/No	Set to Yes if you created AFP overlays for each FAP file.
ChartResolution	120/150/240/300	Used when printing charts as inline bitmap graphics on an AFP printer that does not have graphics (GOCA) support. Defaults to one-half of the Resolution option setting.
LandscapeSupport	Yes/ No	Although not required for printing, you can set this option to Yes if your printer supports landscape medium maps. Generally, AFP printers using cut-sheet paper <i>do not</i> support landscape medium maps.
SplitText	Yes/ No	Used to minimize the print differences between 240 and 300 dpi printing.
SplitPercent	0 to 100 (50)	Percentage of the width of the space character used to determine when the rounding error between 240 and 300 dpi printing has caused a significant difference and the text string should be split into smaller strings.
FudgeWidth	any number (0)	Can be used when building page overlays for sections smaller than a page.
GraphicSupport	0, 1, 2, 3	0 = no graphics (GOCA) support 1 = inline bitmap graphics support 2 = GOCA charts support 3 = inline bitmap graphics and GOCA charts support
PageNumbers	Yes/ No	Set to Yes to turn on form or form set page numbering

Option	Values	Description
PrintViewOnly	Yes/No	If set to Yes, the view only sections will print. This does not apply to entry only sections, which are never printed. Entry only sections are usually worksheets. If the section is marked as hidden and view only, it will not print.
PrePrintedPaper	Yes,Disabled	Determines if the check box which lets you print or not print pre-printed objects appears on the Print window. Also determines the default for this check box—checked or unchecked. You must add this option to the INI file if you want the check box to appear on the Print window. The default for this option includes the check box on the Print window and leaves it unchecked. All objects except fields can be designated as pre-printed on the object's Properties window.
Class	<i>(first three characters of the Module option)</i>	Specifies the printer classification, such as AFP, PCL, XER, PST, or GDI. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the Class option to make sure the correct printer is available before continuing.
OnDemandScript		Use this option to add comments to the print stream. This lets you handle archiving using OnDemand. Enter the name of the DAL script you want the system to run. This DAL script creates the On Demand records and adds them as comments. The AddComment function is also used in DAL scripts to add OnDemand command records. For more information about this and other functions, see the DAL Reference.
TLEScript		Enter the name of the DAL script to execute to add Tagged Language Element (TLE) records to the print stream. See <i>Adding TLE Records</i> on page 305 for more information.
TLESeparator		Enter the character you want to use to separate the key and value portions of the TLE comment string.
TLEEveryPage	Yes/No	(Optional) If you enter Yes, the TLE DAL script will be executed at the start of every page. If you enter No, the TLE DAL script is executed at the start of every form set. The default is No.
PaperSize	0, 1, 2, 3, 98	Use this option to set a default paper size when converting AFP print streams using the Internet Document Server or the MRG2FAP utility. Enter zero (0) for letter size (default) Enter 1 for legal size Enter 2 for A4 size Enter 3 for executive size Enter 98 for a custom size

Option	Values	Description
DocusaveScript		<p>Use this option to add comments to the print stream. This lets you handle archiving using Docusave.</p> <p>Enter the name of the DAL script you want the system to run. This DAL script creates the Docusave records and adds them as comments.</p>
SendColor	Yes/No	<p>Enter Yes to send color information to the printer. AFP highlight color printing on printers from Xerox and Oce is supported.</p> <p>Make sure the objects you want to print in color (text, lines, shades, and so on) are set to print in color. The Print in Color option is on the Color Selection window. You can display this window by clicking the Color button on the object's Properties window.</p>
NamedColors		<p>Use this option to tell the system to use only specific AFP named colors. For example, if you wanted all highlight (non-black) colors mapped to blue, you would set the NamedColors option to <i>blue</i>.</p> <p>To allow the mapping of the colors you assigned to the objects in the FAP file to multiple colors, separate each color with a semicolon (;). For example, to use red, blue, and magenta, set the NamedColors option as shown here:</p> <p><code>NamedColors = red;blue;magenta</code></p> <p>The order you list the colors does not matter.</p>
SkipChartColorChange	Yes/No	Enter Yes to suppress color changes normally done to enhance 3D bar charts.
SuppressLogoUnload	Yes/No	Enter Yes to suppress the unloading of graphics (LOG) files during a conversion of AFP files to FAP (or PDF) format. The default is No.
ReplaceBitmap	LIGHT, LIGHTER, LIGHTEST, MEDIUM, DARK, DARKER, DARKEST, NOSHADE, SOLID, HORIZONTAL, VERTICAL, DIAGRIGHT, DIAGLEFT, HATCH, or DIAGHATCH	<p>Enter the name of the bitmap you want to replace followed by one of the replacement patterns.</p> <p>The default is LIGHT.</p> <p>Keep in mind your entry must be in all caps.</p> <p>See <i>Using Documaker Shading Patterns Instead of Shaded Bitmaps</i> on page 41 for more information.</p>
DisplayCodedFont	Yes/No	<p>Enter No to include the character set/code page combinations in the AFP font list, instead of the coded fonts. The default is Yes, which tells the system to include the coded fonts.</p> <p>See <i>Outputting Character Set and Code Page Information</i> on page 42 for more information.</p>

There are some additional options you can use to print inline graphics (LOG files). Be aware that not all AFP printers support these settings. You'll find these options in the AFP control group.

AFP Options	Values	Description
OutputHalfRes	Yes/No	Scales the bitmap loaded from the graphic to half resolution in memory before writing the output.
DoubleOutputRes	Yes/No	Does not change the bitmap loaded from the graphic, but would tell the printer to double its resolution when printed. This lets the system load graphics that are half resolution already.
SuppressZeroData	Yes/No	Suppresses data containing a series of zeros (white space in the bitmap).
TrimWhiteSpace	Yes/No	Suppresses data containing zeros (white space) at the right edge of the bitmap.
MultiLinesPerCommand	Yes/No	Tries to combine AFP commands into fewer records when printing the bitmap. You cannot use this option with the SuppressZeroData option.

Adding Support upto Nine(9) Paper Trays

To enable support for paper trays 5 through 9, you will need to use a new AFP formdef file, F1DOCUMK.FDF for printing. You must add the following INI setting to your AFP printer INI group that you use when producing AFP output.

```
< PrtType:AFP >  
Formdef = F1DOCUMK:
```

Using Defaults for the Module and PrintFunc Options

Default values for the Module and PrintFunc options in the PrtType:xxx control group are provided when you use a standard print type name or print class, such as AFP, PCL, PDF, PST, VPP, XER, XMP, or GDI.

These defaults keep you from having to enter the Module and PrintFunc names in your INI file. For example, if you want to generate AFP print files, you can specify these INI options:

```
< Printer >  
  PrtType = MYAFP  
< PrtType:MYAFP >  
  Class   = AFP
```

And the system will default these options for you:

```
< PrtType:MYAFP >  
  Module    = AFPPRT  
  PrintFunc = AFPPrint
```


Using Documaker Shading Patterns Instead of Shaded Bitmaps

You can replace the shading bitmaps in AFP files with Documaker's internal FAP shading patterns. Using Documaker's internal FAP shading patterns results in smaller and more efficient FAP files and you will have more flexibility in choosing patterns.

To use Documaker FAP shading patterns, include the ReplaceBitmap INI option, as shown here:

```
< PrtType:AFP >  
  ReplaceBitmap =
```

Note The system ignores this option if the AFP output file being loaded is one generated by Documaker because it automatically replaces shading bitmaps from internally-generated AFP files with FAP shading patterns when appropriate.

The system replaces all occurrences of the bitmap you specify with the shading pattern you choose. The system places the replacement shading pattern in the same location as the AFP bitmap. To replace multiple bitmaps, repeat the ReplaceBitmap option as necessary.

The bitmap patterns that are replaced must be named in bytes 10-17 of the Begin Image (D3 A8 7B) AFP structured field and the bitmap name listed in the ReplaceBitmap option must match the bitmap name in the Begin Image structured field. All Begin Image structured fields encountered that have names that match the name in the ReplaceBitmap option are replaced.

Note While the system does support color text, color bitmaps are not supported by the AFP loader of the MRG2FAP utility.

Printing Highlight Colors

The system supports AFP highlight color printing on printers from Xerox and Océ. Like other color printer support, the SendColor option must be set to Yes and the objects, such as text, lines, and shades must be set to *Print In Color*.

The RGB (red,green,blue) color setting for each FAP object is mapped to the closest AFP named color. The names of the available colors are as follows: blue, red, magenta, green, cyan, yellow, dark_blue, orange, purple, dark_green, dark_cyan, mustard, gray, and brown.

You use the NamedColors option in the AFP printer group to specify certain AFP named colors. For example, if you wanted all FAP (non-black) colors to be mapped to brown, you would use this INI option:

```
NamedColors = brown
```

To let the system map FAP colors to multiple colors, separate each color with a semicolon (;). For example, to use all of the default AFP named colors except brown, you would use this INI option:

```
NamedColors = Red;Blue;Magenta;Green;Cyan;Yellow
```

Note The order in which you name the colors does not matter. In addition, the LOG2PSEG and FAP2OVL utilities include a /C=color parameter, where color is the one of the named AFP colors.

Character Set and Code Page Font Information

When loading AFP, the system uses the information in the Character Set and Code Page Font fields in the FXR file instead of using the font information contained in the IBMXREF.TBL.

The AFP loader expects the AFP file's Map Coded Font (MCF) structured fields to contain references to AFP coded fonts. However, MCF structured fields can contain character set and code page information instead of the coded font information the FXR file requires.

Before version 11.2, for MCF structured fields that contained character set and code page information instead of coded fonts, you had to manually set up the IBMXREF.TBL file to resolve the character set/code page information to coded fonts in the FXR file.

Since the system includes character set and code page information in the FXR file, the AFP loader first checks the FXR file for this information and, if it exists, uses it. If the information does not exist, the AFP loader loads the information from the IBMXREF.TBL file.

Outputting Character Set and Code Page Information

You can output the AFP character set and code page combination instead of the coded font in the font list when you generate normalized AFP files. If you want the character set/code page combinations to be output in the AFP font list, instead of the coded fonts, you must add the DisplayCodedFont option, as shown here:

```
< PrtType:AFP >  
    DisplayCodedFont = No
```

Keep in mind the FXR file must contain the character set and code page entries in the AFP font record for this option to work. If you set the INI option to No and the character set and code page entries are not in the FXR file, the font list in the AFP file will contain only the coded fonts.

Note The AFP output record can only contain *either* coded fonts *or* character set/code page entries — it cannot contain a combination. It will default to coded fonts for all if the font for one or more objects does not contain character set/code page entries.

Using Multiple Code Pages

You can use multiple code pages for creating AFP output. While the standard 37 code page is the default code page, alternate code pages are frequently used for fonts set up for them. Here is a summary of the new font definition files which were created to let you specify code pages:

File	Description
CODED.FNT	The coded font definitions. This file specifies which AFP code page and AFP font character set make up the coded font.
CPDEF.FNT	The code page definitions. This file maps each AFP code page to a Windows character set.
CPGID.CP	The code page map file. This file contains the character identifiers (and associated EBCDIC hexadecimal code points) for an IBM code page and maps them to character identifiers (and associated ASCII code points) for a Windows ANSI or SYMBOL character set.

Here are the general syntax rules for all new font definition files:

- A semicolon (;) in the first column of any of these files will cause the line to be treated as a comment statement and ignored.
- Section headers within files are enclosed either in brackets (<> or []) with *no* spaces and must *not* be removed or changed.
- All values are case insensitive.
- If a parameter value is invalid and a default value exists, it will be substituted.
- All parameters are positional.
- Blanks are allowed between parameter values.
- The question mark (?) is used in some areas as a single wildcard character.
- If the resource file exists in DEFLIB directory and contains valid data conforming to these specifications, it will be loaded and used.
- If bad data is encountered in the file, either the offending record is ignored or a warning is issued. If the file is considered corrupt or invalid enough, it may not be used at all.

Using the CODED.FNT File

This file specifies which AFP code page and AFP font character set make up the coded font. The CODED.FNT file is necessary for basic multiple code page support.

When creating this file, keep these rules in mind:

- The coded font name and both parameters are required.
- A question mark (?) can be used as the wild-card character only for the second character in the coded font name and for any character of the character set name. This allows all the character rotations of the coded fonts to be handled with one entry for searching.
- After the coded font name, the character set name must be listed first, followed by the code page name.

- The character set and code page must be separated by a comma.

Here is an example of this file:

```
X?COL8=C?420080,T1000850
X?COL7=C?420070,T1000850
;Core
X?H210AC=C?H200A0,T1V10500
X?H210FC=C?H200F0,T1V10500
;FormMaker Fonts
X?FA????=C?FA????,T100ASC4
X?DA????=C?FA????,T1DOC037
X0P09X12=C0P09X12,T1DOC037
X0P12X16=C0P12X16,T1DOC037
```

Using the CPDEF.FNT File

This file maps each AFP code page name to its code page global identifier (CPGID) and to a Windows character set. If you do not have at least one valid entry in this file for each code page you want to use, the system uses the default code page.

When creating this file, keep these rules in mind:

- Parameters must be separated by a comma.
- AFP code page name and code page identifier are required.
- If you create your own code page, you must assign it a unique code page identifier. Leading zeros are invalid.
- Code Page Global Identifier (CPGID) attribute's possible values: IBM-defined CPGID or your own defined CPGID between 65280 and 65534, inclusively. This value matches the name of a code page map file.
- For each CPDEF.FNT entry, you must have a corresponding code page map file with the same name as the CPGID.
- Windows character set attribute's possible values: ANSI or SYMBOL.

Here is an example of this file:

```
<CODEPG>
;codepage = cpgid,wincp
;*****Put User-defined/Custom code pages Here *****
T100ASC4=361,ANSI
T1DOC037=37,ANSI
T1OMR=5280,ANSI
T1POSTBC=5280,ANSI
;***** End User-defined/Custom code pages *****
T1000259=259,SYMBOL
T1000290=290,ANSI
T1000293=293,ANSI
T1000310=310,ANSI
DEFAULT=361,ANSI
```

Using the CPGID.CP (Code Page Map) File

You must have a separate CPGID.CP file for each AFP code page entry in the CPDEF.FNT file. Each code page map file contains the character identifiers (and associated EBCDIC hexadecimal code points) for an IBM code page and maps them to character identifiers (and associated ASCII code points) for a Windows ANSI or SYMBOL character set. Code page map files are necessary for basic multiple code page support.

Note The actual file name is not CPGID.CP, but rather the *CPGID* value from the CPDEF.FNT file with an extension of *CP*. For instance, in the CPDEF.FNT example, the first two lines are:

```
T100ASC4=361,ANSI
T1DOC037=37,ANSI
```

So, since those two entries are in the CPDEF.FNT file, that means that there must be code page map files with named *361.CP* and *37.CP*.

Also, if these two entries are in the CPDEF.FNT file, but the corresponding 361.CP and 37.CP code page map files are not in DEFLIB, the translations for those fonts will not be correct.

When creating this file, keep these rules in mind:

- Parameters must be separated by blanks.
- All four parameters are required.
- “NOMATCH” means there is not a matching character in the Windows character set.

Here is an example of this file:

(395.cp for the T1000395 code page mapped to the Windows ANSI character set):

```
;T1000395 to ANSI
SP010000 40 SP010000 20
LA150000 42 LA150000 E2
LA170000 43 La170000 E4
LA130000 44 LA130000 E0
SP180000 8B SP180000 BB
SM560000 8C SM560000 89
SA000000 8D SP100000 2D
LI510000 8E NOMATCH 00
LI570000 8F NOMATCH 00
SM190000 90 SM190000 B0
LJ010000 91 LJ010000 6A
LF510000 A0 NOMATCH 00
;;;;;;;; ; SD150000 5E
;;;;;;;; ; SD130000 60
```

Using LLE Records to Link to External Documents

For AFP files, LLE (Link Logical Element) records let you link internal or external documents into the AFP presentation space. For example when you are creating a PDF file, you might want to include in the text hotspots that link to a URL. These hotspots, when clicked, open that document.

Note The LLE records are for use with text fields.

Place the LLE record immediately before the *BPT* – Begin Presentation Text record. Then, following the BPT record, you can have any number of PTX records containing a *TRN* (Transparent Data) control sequence, followed by a terminating *EPT* – End Presentation Text.

Here is an example of the LLE format:

Element	Description
5A	
00 32	record length
D3B490	LLE
00	Flags
00 00	reserved
01	Navigation Link Type
00	reserved
00 05	triplet length including this value
02	Link Source specification
/N	source text limited by triplet size) See below explanation of /N
00 11	triplet length including this value) 0x11 (17 decimal (2+1+14)
03	Link Target specification
http://xyz.com	target text limited by triplet size

In the above example, the text fields */N* and *http://xyz.com* would be encoded as hex EBCDIC. For example a source link such as:

```
00 05 02 /N
```

would be encoded as...

```
00 05 02 61 D5
```

The FAP library does not use the name (link source) member of the FAPLINK, therefore it is used for feature steering.

By specifying a */N* (NEXT) as the source name, the system applies the current instance of the LLE to the first occurrence of a PTX record containing a TRN (Transparent Data) control sequence record. Once the LLE link information has been applied to that particular PTX FAOBJECT, the system clears the LLE status so subsequent PTX records are rendered as non-hyperlinked text.

By default the LLE is applied to all subsequent PTX / TRN records until either an LLE is encountered with a */C* as its source link to enable the clearing of the active instance of the LLE, or to use a normal valid LLE to supersede the prior usage.

If you are not using a */N* or */C*, you may use the source name area of the LLE for a brief descriptive label.

Note The system does not support the use of the attribute link type or internal target links within FAP and therefore PDF documents

The system only supports the conversion of LLE records in FAPSTEXT objects and linking to external documents.

Adding Data to Begin Page and End Page Records

The AFP MO:DCA structured records for Begin Page (BPG) and End Page (EPG) include an 8-byte field that you can use to store the name of the form used to create the page. Adding form names to these records can allow certain AFP archival products to re-assemble portions of documents, instead of having to deliver the entire document.

Use the BPGScript option to add this data to the AFP Begin Page and End Page records. Here is an example:

```
< PrtType:AFP >
  BPGScript = example.dal
```

Option	Description
--------	-------------

BPGScript	Enter the name of the DAL script you want the system to execute at the start of every page to pass data to AFP Begin Page and End Page records in AFP print streams.
-----------	--

The DAL script you specify should contain a call to the AddComment function. The script can pass any string as a parameter to the AddComment function but strings longer than eight bytes are truncated.

For example, if you wanted to write the form name into the BPG (Begin Page) and EPG (End Page) records, your DAL script could include the following lines:

```
form_name = FormName()
ADDCOMMENT(form_name)
```

The BPGScript DAL script is similar to other functionality for adding comments to AFP print streams, such as DocuSaveScript, OnDemandScript, and TLEScript. You can use any combination of these script INI options.

Keep in mind...

- AFP records normally contain EBCDIC data. By default, the AddComment function converts the string to EBCDIC.
- Do not use the AddComment option to write ASCII text.

- Use only 7-bit ASCII characters in strings added via the AddComment function because 8-bit ASCII characters may be translated to EBCDIC characters other applications will not understand.
- AFP archival products may have specific requirements for the text in Begin Page (BPG) and End Page (EPG) records. For example, they may not expect spaces within the text. Make sure that you address these requirements when creating the AddComment strings in your DAL script.

DEFINING AFP PRINTER RESOURCES

FormDef

The system uses copy groups from its own FormDef named *FIFMMST.DAT*. Each copy group in a FormDef contains information about paper size, duplex, tray selection, jog, orientation, and so on. The FormDef must be available to PSF to print AFP data streams. You can use the AFPFMDEF utility to create or modify the FormDef.

Fonts

AFP fonts are designed solely for AFP printers. In IBM AFP terminology, a font is described by three components:

Coded Font

A coded font file contains references to specific character set and specific code page. Coded font files always begin with the letter *X*, such as *X0DATIN8*.

Code Page

In IBM AFP terminology, a code page file maps code points to an AFP character name in a character set file. Code page files always begin with the letter *T*, such as *TIDOC037*.

Character Set

A character set file contains the bitmap graphic of each character in the character set. Character set files always begins with the letter *C* (such as *C0FATIN8.240* or *C0FATIN8.300*). The character set file name extension (*240* or *300*) indicates whether the bitmap graphics are drawn at 240 or 300 dots per inch.

System Fonts

Documaker includes a set of system fonts you can use to create your documents. The system fonts include TrueType fonts and equivalent printer fonts in PCL, AFP (240 and 300 dpi), and Xerox font formats.

Overlays

Use the FAP2OVL utility to create AFP overlays from FAP files. The OVLCOMP utility also lets you create AFP overlays from FAP files. These overlays must be available to PSF to print AFP data streams when the SendOverlays option is set to Yes.

Page Segments

Use the LOG2PSEG utility to create AFP page segments from graphics (LOG files). These page segments must be available to PSF to print AFP data streams.

Note For information on system utilities, see the Utilities Reference.

AFP 2-up Support

The system include rules you can use to generate and merge print streams for AFP printing for printers that support 2-up printing. See the [Documaker Administration Guide](#) for more information.

INCLUDING DOCUMERGE FORM-LEVEL COMMENT RECORDS

You can include Documerge form-level comments in AFP print streams produced by Documaker. You may want to include form-level comments if you have a reprint utility program that needs information about a form before it can reprint it.

To include form-level comment records, add the FormNameCR option in your AFP printer control group and set it to Yes, as shown here:

```
< PrtType:AFP >
  FormNameCR    = Yes
  Module        = AFPPRT
  PrintFunc     = AFPPrint
  SendOverlays  = Yes,Enabled
  ...
```

Here is an example of the AFP records in an AFP print stream which includes the Documerge form level comment (NOP) records:

```
000, Begin, Document, 29,
001, Data, NOP, 84, %DMGFORMBEG%% DEC PAGE                                00001
AFP Docucorp 000001
002, Map, Medium Map, 16, PLUD
...
033, End, Page, 16,
034, Data, NOP, 84, %DMGFORMEND%% DEC PAGE                                00001
AFP Docucorp 000001
035, Data, NOP, 84, %DMGFORMBEG%% LETTER                                  00001
AFP Docucorp 000002
036, Begin, Page, 16,
...
053, End, Page, 16,
054, Data, NOP, 84, %DMGFORMEND%% LETTER                                  00001
AFP Docucorp 000002
173, End, Document, 16,
000, Begin, Document, 29,
001, Data, NOP, 84, %DMGFORMBEG%% OP714                                    00001
AFP Docucorp 000001
002, Map, Medium Map, 16, PLUO
...
```

AFP TROUBLESHOOTING

Floating Section Limitations

The system lets you compose a page from several sections. The system also lets you create overlays for these sections. There is one limitation when you print these sections on a landscape page. Overlays on a landscape page can only be placed vertically on the page. Overlays on a landscape page *cannot* be placed horizontally on the page.

This means, in your SetOrigin rule, you *cannot* specify any non-zero, positive number for the X-relative displacement. Create your FAP files accordingly, but keep in mind that they can be moved down but not across. This limitation exists only for AFP overlays, and only in landscape mode.

Objects Extending Beyond the Edges

Another type of error can occur if the overlay for a custom-sized section is too small for the objects (text, lines, graphics, and so on) contained within it. If the AFP overlay's page size is too small, objects may be clipped to the page size, printed as solid black rectangles, or trigger error messages.

Documaker Studio offers an Auto-size option which you can use to make sure the custom-sized section is large enough to contain all objects placed within it. Use this feature to prevent most custom page size problems.

Be careful placing text at the extreme left edge of the section because it may cause errors that the Auto-size option cannot detect. For instance, suppose you have this text label positioned on the left edge of the FAP file (left offset = 0):

Beneficiary

When printed, black rectangles or an error message may appear instead of the text.

This can occur because some of the characters in the italic font (Times New Roman) have a *negative left offset*. This means that the characters print to the left of where they would normally start. A negative left offset may be easier to understand by looking at these characters:

ef

Notice how the bottom of the *f* goes under the *e*. This is an example of a negative left offset. Because it is positioned to the left of where it would normally start, the character is now positioned off the left edge of the overlay.

This kind of detailed character information is not stored in the FXR file so Documaker Studio has no way to know there may be a problem. You can, however, move the text labels in the FAP file to correct the problem.

Conflicts Between Page and Form Orientation

If you create a custom-sized page, be aware of any conflict between page orientation and the form orientation. If the form orientation is not the same as the page orientation, the page will not print according to the page orientation, but will follow the form's orientation.

Note This happens only in case of custom size pages. Standard size pages obey the page orientation.

Multipage FAP Limitation

There is a problem when a landscape, multipage FAP has different page sizes on each page. All pages of a multipage FAP file should be the same size. As a workaround, use Documaker Studio to correct the page sizes. After saving the FAP file, you can then generate proper AFP overlays.

Printing rotated variable Fields

Here is a list of field options you can specify in the NAFILE.DAT file:

Option	Description
E	Error
M	Manual
P	Protected
G	Global scope (entire form set)
F	Form scope
H	Hidden field – a dummy field, not displayed or printed
N	Nonprintable field (displayed, not printed)
C	Send-copy-to field (receives current recipient name at print time)
9	Rotated 90 degrees
8	Rotated 180 degrees
7	Rotated 270 degrees

Some of these options require the FAP field attributes to be available at runtime, since the DDT file does not include the necessary information. Use the CheckImageLoaded rule to make sure this information is available.

AFP 240 DPI Print Problems

Due to differences in resolution on 240 and 300 dpi printers, a text string may print with slightly different lengths. One example where this may be noticeable is when the text is printed inside of a boxed region. Another example where this may be noticeable is when a text area contains an embedded variable field.

To minimize the print differences between 240 and 300 dpi printing, use the `SplitText` option. Make sure these options are in your printer `PrtType:xxx` control group:

```
< PrtType:AFP >
  SplitText      = Yes/No (default is No)
  SplitPercent   = ###    (% of space-width as max rounding error)
  Resolution     = ###    (default is 300)
```

If you set the `SplitText` option to Yes, each text string is checked to see if it needs to be split into sections for printing. The `SplitPercent` value helps determine when a text string must be split into sections for printing.

The `SplitPercent` option sets the percentage of the width of the space character to use as the maximum amount of rounding error that can accumulate in a string before it is broken into sections.

The `SplitPercent` value is from zero (0) to 100. Do not enter a value greater than 100. For example, if you set the `SplitPercent` option to 75, the string is broken into sections if the accumulated rounding error is greater than 75% of the width of the space character. This value is set to 50 by default.

Note Using 50 as the `SplitPercent` value is a good trade-off between the appearance and the performance impact on the GenPrint program and print spool size. Setting the `SplitPercent` option to a smaller value gives you a more accurate printout but slows the GenPrint program, increases the size of the print spool, and increases the amount of time it takes to print.

The `Resolution` option determines the rounding error. Most FXR files are built using 300 dpi fonts. This causes rounding errors when the FXR is used for printing to a 240 dpi printer. If you omit the `Resolution` option, the system uses the default setting of 300.

You need to know whether the FXR you are using was built by importing 300 dpi fonts or 240 dpi fonts. The standard FXR files are built using 300 dpi fonts. When an FXR is built using 300 dpi fonts, there are rounding errors when printing to a 240 dpi printer.

Here are some examples of options to use in different situations:

- If your font cross-reference (FXR) file was built from 300 dpi fonts and your printer resolution is 240 dpi, set the options as shown here:

```
< PrtType:AFP >
  SplitText      = Yes
  SplitPercent   = 50
  Resolution     = 240
```

- If your font cross-reference file was built from 240 dpi fonts and your printer resolution is 300 dpi, set the options as shown here:

```
< PrtType:AFP >
```

```
SplitText      = Yes  
SplitPercent   = 50  
Resolution     = 300
```

- If your font cross-reference file was built from 300 dpi fonts and your printer resolution is 300 dpi, you do not need to set the SplitText option.
- If your font cross-reference file was built from 240 dpi fonts and your printer resolution is 240 dpi, you do not need to set the SplitText option.

Chapter 3

Using the Bitmap Print Driver

This chapter provides information about Documaker's Bitmap Print Driver. This print driver lets you create bitmap output from Windows implementations of Documaker software.

This chapter includes the following topics:

- *Overview* on page 58
- *Setting Up INI Files* on page 59
- *Choosing Fonts* on page 65
- *Working with Color* on page 67
- *Handling Multiple Page Form Sets* on page 68
- *Selecting the Bitmap Print Driver* on page 73
- *Additional Considerations* on page 74

OVERVIEW

Using the Bitmap Print Driver, you can create bitmaps in several formats from a form set or a single FAP file (section). Output from the Bitmap Print Driver can be written to disk and stored in one or more files. You can view and print the output in any application that reads bitmap files. You can also archive output using other software applications.

The bitmap print driver works just like other print drivers available for Documaker applications. The Bitmap Print Driver lets you specify a bitmap format to use, such as LOG, TIF, JPG, or BMP.

Prerequisites

First make sure you have the correct system requirements to run your Documaker software. For instance, if you are using Documaker Desktop, see the [Documaker Desktop Installation Guide](#) for information on what you need to run those systems. For Documaker Server and Documaker Studio, see the [Documaker Installation Guide](#) for more information on system requirements.

Once you have made sure you have the correct system configuration to run Oracle Documaker software, follow these steps to use the Bitmap Print Driver:

1. Customize your INI files. See *Setting Up INI Files* on page 59.
 - Review *Choosing Fonts* on page 65
 - Review *Handling Multiple Page Form Sets* on page 68
2. Select the Bitmap Print Driver in your Documaker application. See *Selecting the Bitmap Print Driver* on page 73 for more information.

Note The Bitmap Print Driver is installed when you install Oracle Documaker.

SETTING UP INI FILES

You use INI options in your FSISYS.INI file to tell the system how you want the Bitmap Print Driver to work. These options differ slightly depending on the Documaker application you are using.

Note Before making any changes to these files, back up your INI files.

DOCUMAKER DESKTOP AND DOCUMAKER STUDIO INI OPTIONS

Include these INI options to set up the Bitmap Print Driver for Documaker Desktop and Documaker Studio:

```
< Printers >
  PrtType           = BMP
< PrtType:BMP >
  BMPTType          = TIF
  Module             = BPDW32
  PrintFunc          = BPDPrint
  Device             = NULL
  SendColor          = Yes, Enabled
  GrayShades         = Yes
  ForcePrintinColor = Yes
  SelectRecipients   = Yes, Enabled
  Resolution         = 300
  DefaultSymSet      = W1
  PageNumbering      = Yes
  RotateLandscapePages = Yes
  Fonts              = PCL, TTF, PS, AFP, XER
```

Option	Description
Printers control group	
PrtType	<p>You may have several printers defined using PrtType options, such as PCL, AFP, and XER. To this list, add another PrtType option set to identify the Bitmap Print Driver, as shown here:</p> <p>PrtType = BMP</p> <p>You can call the printer driver anything you like, <i>BMP</i> is just an example. Just make sure what you choose is reflected in the name of the PrtType:<i>BMP</i> control group.</p>
PrtType:BMP control group	
BMPTType	<p>Use this option to specify the bitmap format you want to create, such as compressed LOG, LOGPACK, TIF, MTIF, BMP, FNT, IMG, PNG, SEG, and JPG.</p> <p>The default is compressed LOG format.</p>
Module	<p>Enter BPDW32.</p> <p>This is the name of the program module which contains the print driver.</p>
PrintFunc	<p>Enter BPDPrint.</p> <p>This is the name of the program function that is the main entry point into the print driver.</p>

Option	Description
Device	<p>This option is used by GUI applications such as Documaker Studio or Documaker Desktop. Documaker Server ignores this option.</p> <p>Enter the name of the file or device where the output should be written.</p> <p>If this option is not set or is set to NULL, the system generates a file name based on the page number, such as 00000001.LOG for the first page, 00000002.LOG for the second page, and so on.</p> <p>Otherwise, the system uses the name you provided for the first page and then appends 00000002.log for the second page and so on.</p> <p>If you include a file extension in the file name for the device, that extension is used for all files produced. Otherwise, the system assigns the file extension to match the type of output you are producing.</p>
SendColor	<p>Enter No for black and white bitmaps. Enter Yes for color bitmaps.</p> <p>You also have these options:</p> <p>Enabled = Send Color field appears in the Print window and is active (available to be checked).</p> <p>Disabled = Send Color field appears in the Print window but is grayed out (not available to be checked).</p> <p>Hidden = Send Color field does not appear in the Print window</p> <p>For instance, Yes, Enabled indicates color bitmaps and displays the Send Color field on the Print window where it can be checked or not.</p> <p>Keep in mind that the higher the color depth and resolution, the longer it will take to create output. For example, changing from 24-bit color bitmap to monochrome (black and white) makes the bitmap 24 times smaller.</p> <p>For example, a 24-bit color bitmap at 300 DPI that measures an 8.5 x 11 inches will require a bitmap file that is roughly 25MB in size.</p> $300 \times 300 \times 8.5 \times 11 \times 24 / 8 = 25,245,000 \text{ bytes}$ <p>300 = resolution in DPI 8.5 X 11 = letter page size in inches 24 =color depth in bits 8 = bits in one byte</p> <p>The larger the bitmap, the slower the processing.</p>
GrayShades	Enter Yes to print in shades of gray. The default is No.
ForcePrintInColor	Enter Yes to print in color. The default is No.
SelectRecipients	<p>Enter No to disable the ability to select recipients. The default is Yes.</p> <p>Enabled = Appears in the Print window and is active (available to be checked).</p> <p>Disabled = Appears in the Print window but is grayed out (not available to be checked).</p> <p>Hidden = Does not appear in the Print window.</p>
Resolution	<p>Specify the bitmap resolution in pixels per inch. Valid entries range from 30 to 600 pixels per inch. The default is 300.</p> <p>Keep in mind that the higher the resolution and color depth, the longer it will take to create output. For example, changing the resolution from 300 DPI to 150 DPI makes the output four times smaller.</p> <p>Again, the larger the bitmap, the slower the processing.</p>
DefaultSymSet	<p>(Optional) Specify the symbol set.</p> <p>The default is W1 for TrueType (TTF) and PostScript (PS) font types.</p>

Option	Description
PageNumbering	(Optional) Enter Yes to turn on form or form set page numbering. The default is No.
RotateLandscapePages	(Optional) Enter Yes to rotate landscape pages left 90 degrees. The default is No.
Fonts	<p>(Optional) This only affects objects that have to be converted to bitmaps, such as charts, bar codes, and vectors.</p> <p>Use this option to specify the fonts you intend to use, in order. For example, if you set the Fonts option to <i>PCL,TTF</i>, the Bitmap Print Driver first locates the PCL font. If the PCL font does not exist, it finds the TTF font.</p> <p>The default font order is: PCL,AFP,XER,TTF,PS.</p>

In addition, make sure the following options are set correctly in your FSISYS.INI file. These options provide font and character set information the system needs. For more information, see *Choosing Fonts* on page 65.

```
< MasterResource >
  FontLib   = ..\frmes\fontlib\
  XRFFile   = REL113SM
< FMRes >
  DefLib     = ..\frmes\deflib\
```

Option	Description
MasterResource control group	
FontLib	Specify the font library you want to use.
XRFFile	Specify the font cross-reference (FXR) file you want to use, such as REL121.FXR.
FMRes control group	
DefLib	<p>Specify the path to the CODEPAGE.INI and other files for TrueType (TTF) and PostScript (PS) font types.</p> <p>See <i>Choosing Fonts</i> on page 65 for more information on the files required in the directory you specify here.</p>

Note If the FontLib and DefLib options point to the same directory, the Oracle Image Export software will find the plugin.ttf file, which is not a normal TrueType font file. The Oracle Image Export software is used in converting various file types into bitmaps, such as when you use the AddMultiPageBitmap rule. If the Oracle Image Export software uses the plugin.ttf to convert a file into a bitmap, text represented in the bitmap may not display properly. The plugin.ttf file is no longer shipped with Documaker but may exist on your machine from an earlier release.

Documaker uses the Java Runtime Environment (also known as the Java Virtual Machine) to support TrueType and PostScript fonts. For Documaker to use the JVM, the system looks for the jvm DLL/DSO (jvm.dll for Windows, libjvm.so for Linux and Solaris) and supporting files to be installed in one of these locations:

- In your library PATH
- In the ..\jre\bin\client directory, relative to the current directory

- In the `..\\jre\\bin\\client` directory, relative to any directory in your PATH

If you want to create a single TIFF file per form set, see *Handling Multiple Page Form Sets* on page 68 for more information.

DOCUMAKER SERVER INI OPTIONS

For multistep GenPrint and single-step GenData, you set up the Bitmap Printer Driver just as you would the other printer drivers. Here is an example of the INI options you would use to set up the GenPrint program in Documaker to use the Bitmap Print Driver:

Note See also *Handling Multiple Page Form Sets* on page 68 for information on setting the INI options necessary to have Documaker Server create one output file per form set.

When you print multiple transactions from Documaker Server, be sure to set up MultiFilePrint callback function. This function lets you run the GenData program in single-step mode. See the [Documaker Administration Guide](#) for more information.

```
< Printer >
  PrtType      = BMP
< Printer1 >
  Port         = DATA\~tmp.tif
< Printer2 >
  Port         = DATA\~tmp.tif
< Printer3 >
  Port         = DATA\~tmp.tif
< Printer4 >
  Port         = DATA\~tmp.tif
< Printer5 >
  Port         = DATA\~tmp.tif
< PrinterInfo >
  Printer      = Printer1
  Printer      = Printer2
  Printer      = Printer3
  Printer      = Printer4
  Printer      = Printer5
< PrtType:BMP >
  BMPTType     = TIF
  Module       = BPDW32
  PrintFunc    = BPDPrint
  Device       = dummy.txt
  SendColor    = No
  Resolution   = 300
  Fonts        = PCL, AFP, XER, TTF, PS
```

Option	Description
--------	-------------

Printer control group

PrtType	Enter BMP . You can call the printer driver anything you like, <i>BMP</i> is just an example. Just make sure what you choose is reflected in the name of the PrtType: <i>BMP</i> control group.
---------	---

PrinterX control group

Port	Enter the name of the print batch file for each designated printer. Note the control group name is defined by the printer option in the PrinterInfo control group. Keep in mind that the print batch file should always include a correct file extension. If, however it does not contain an extension, such as PORT=DATA\~TMP, the printer driver uses the PrtType as the file extension. In this case BMP, so it becomes DATA\~TMP.BMP.
------	--

PrinterInfo control group

Printer	Enter the designated printers for the print batches.
---------	--

Option Description

PrtType:BMP control group

BMPTYPE	Use this option to specify the bitmap format you want to create, such as compressed LOG, TIF, JPG, BMP, or MTIF for multipage TIFF. The default is compressed LOG format.
Module	Enter BPDW32 . This is the name of the program module which contains the print driver.
PrintFunc	Enter BPDPrint . This is the name of the program function that is the main entry point into the print driver.
Device	This option is ignored by the GenPrint program but should not be left blank or omitted. For instance, you could enter <i>dummy.txt</i> .
SendColor	Enter No for black and white bitmaps. Enter Yes for color bitmaps. Keep in mind that the higher the color depth and resolution, the longer it will take to create output. For example, changing from 24-bit color bitmap to monochrome (black and white) makes the bitmap 24 times smaller. For example, a 24-bit color bitmap at 300 DPI that measures an 8.5 x 11 inches will require a bitmap file that is roughly 25MB in size. $300 \times 300 \times 8.5 \times 11 \times 24 / 8 = 25,245,000 \text{ bytes}$ $300 = \text{resolution in DPI}$ $8.5 \times 11 = \text{letter page size in inches}$ $24 = \text{color depth in bits}$ $8 = \text{bits in one byte.}$ The larger the bitmap, the slower the processing.
Resolution	Specify the bitmap resolution in pixels per inch. Valid entries range from 30 to 600 pixels per inch. The default is 300. Keep in mind that the higher the resolution and color depth, the longer it will take to create output. For example, changing the resolution from 300 DPI to 150 DPI makes the output four times smaller. Again, the larger the bitmap, the slower the processing.
Fonts	(Optional) Use this option to specify the supported fonts you intend to use, in order. For example, if you set the Fonts option to <i>PCL,TTF</i> , the Bitmap Print Driver first locates the PCL font. If the PCL font does not exist, it finds the TTF font. The default font order is: PCL, AFP, XER, TTF, PS.

CHOOSING FONTS

When TrueType (TTF) and PostScript (PS) font types are used, the Bitmap Print Driver loads the INI file and looks for the DefLib option in the FMRes control group. The path you specified in the DefLib option tells the Bitmap Print Driver where to locate CODEPAGE.INI file. By default, the path is set to...

```
..\fmres\deflib
```

The CODEPAGE.INI file includes some of the default character sets.

Documaker uses the Java Runtime Environment (also known as the Java Virtual Machine) to support TrueType and PostScript fonts. For Documaker to use the JVM, the system looks for the jvm DLL/DSO (jvm.dll for Windows, libjvm.so for Linux and Solaris) and supporting files to be installed in one of these locations:

- In your library PATH
- In the..\jre\bin\client directory, relative to the current directory
- In the..\jre\bin\client directory, relative to any directory in your PATH

When you install the system, the JVM is installed as well. When converting text strings, Intellifont (PostScript) or TrueType fonts are required. These fonts have a file extension PFB or TTF. PostScript and TrueType font files must be installed in the font library. If a required font file is not located, a platform error occurs (only once for the same font error) and the conversion is not performed.

The font information comes from FXR file specified by XRFFile option in the Config control group.

Note You can use the Fonts option in Studio to open the FXR file and edit a selected font.

The Char Set ID (Character Set ID) field denotes the symbol set. It is used by the system to associate code points with characters in both PostScript and TrueType fonts. The default symbol set is *W1* but you can change it using the DefaultSymSet option in the PrtType:BMP control group. Other symbol sets are listed here:

Code	Description	Code	Description
US	7-bit ASCII	WL	Windows Latin/Baltic
W1	Windows Latin/ANSI	WR	Windows Latin/Cyrillic
WE	Windows Latin/East Europe	WT	Windows Latin/Turkish
WG	Windows Latin/Greek	WD	Windows Symbol

Other fonts like PCL, AFP, and Xerox fonts are also supported but the installation is simpler. The FMRes control group and DefaultSymSet option are not required so the supporting files for TrueType (TTF) and PostScript (PS) are not needed.

The Bitmap Print Driver can convert all FAP objects into bitmaps, such as bar codes, boxes, charts, lines, graphics, shades, text areas, text labels, variable fields, vectors, and so on.

Note If there is an error, review the error information in the trace file.

WORKING WITH COLOR

The bits-per-pixel of a page bitmap may be 1-bit, 8-bit, or 24-bit, depending on the objects on the page. For instance...

If	The bitmap will be
All objects are the same color	A 1-bit single color or BW (black and white)
Objects are single colors or use the orthogonal color palette	An 8-bit bitmap based on the orthogonal color palette
The bitmap is from a 24-bit graphic	A 24-bit true color bitmap

If you set the SendColor option to No or uncheck the corresponding field on the Print window the output will be in black and white. If you set the GrayShades option to Yes, 8-bit color bitmaps and 24-bit true color bitmaps are printed in 256 shades of gray.

Be aware that when converted to a bitmap, an object must have a color selection and the Print in Color option should be checked. Otherwise, it will print in black and white — unless you use the ForcePrintInColor option.

FORCING COLOR OUTPUT

You can tell the Bitmap Print Driver to create color output regardless of the object's Print in Color property setting and the print driver's SendColor option setting. To do this, include the ForcePrintInColor option:

```
< PrtType:BPD >
  ForcePrintInColor = Yes
```

Option	Description
ForcePrintInColor	Enter Yes if you want all objects on the form set to print in their default colors. The default (No) tells the system that the object's Print in Color property setting and the print driver's SendColor option determine if the object prints in color. This affects text labels, text areas, text fields, boxes, lines, bar codes, charts, shaded areas, graphics, vectors, and so on.

HANDLING MULTIPLE PAGE FORM SETS

The Bitmap Print Driver can produce bitmap files, typically in TIFF format, for each page of a form set or for the form set as a whole. Depending on the application you are using, you may need to set the INI options to produce the output you want.

FOR DOCUMAKER STUDIO AND DOCUMAKER DESKTOP

For Documaker Studio and Documaker Desktop, there are two ways to process a form set that contains multiple pages. You can create...

- One TIFF file for each page of the form set
- One TIFF file that contains the entire form set

Creating a TIFF File for Each Page of a Form Set

If you set the BmpType option in the PrtType:BMP control group to *TIF*, the Bitmap Print Driver creates a TIFF file for each page of the form set. For instance, if the form set consists of four pages, you would get four TIFF files. For reference, here is an example of the INI options you need to create a TIFF file for each page of the form set:

```
< LogoUnloader:TIF >
  Desc    = Tiff file
  Ext     = .TIF
  Func    = LOGUnloadTifFile
  Module  = LOGW32.DLL
```

Option	Description
Desc	Enter a description. The default is <i>Tiff file</i> .
Ext	Enter an extension for the output file. The default is <i>TIF</i> .
Func	Enter LOGUnloadTifFile .
Module	Enter LOGW32.DLL .

Note If you use the default unloader name, in this case *LogoUnloader:TIF*, the system provides the defaults shown above.

Creating a Single TIFF File for Each Form Set

If you set the BmpType option in the PrtType:BMP control group to *MTIF*, the Bitmap Print Driver creates a single TIFF file for each multipage form set. For example, if the form set consists of four pages, you would get one TIFF file. For reference, here is an example of the INI options you need to create a multipage TIFF file for a form set:

```
< LogoUnloader:MTIF >
  Desc    = Multipage Tiff file
  Ext     = .TIF
  Func    = LOGUnloadMultiTifFiles
  Module  = LOGW32.DLL
```

Option	Description
Desc	Enter a description. The default is <i>Multipage Tiff file</i> .
Ext	Enter an extension for the output file. The default is <i>TIF</i> .
Func	Enter LOGUnloadMultiTifFiles .
Module	Enter LOGW32.DLL .

Option	Description
Desc	Enter a description. The default is <i>Multipage Tiff file</i> .
Ext	Enter an extension for the output file. The default is <i>TIF</i> .
Func	Enter LOGUnloadMultiTifFiles .
Module	Enter LOGW32.DLL .

Note If you use the default unloader name, in this case *LogoUnloader:MTIF*, the system provides the defaults shown above.

In addition, make sure the options in bold are set similar to those shown here:

```
< PrtType:BMP >
  BMPTYPE      = MTIF
  Module       = BPDW32
  PrintFunc    = BPDPrint
  Device      = D:\print\
  SendColor    = Yes, Enabled
  Resolution   = 300
  Fonts        = PCL, TTF, PS, AFP, XER
< Printers >
  PrtType      = BMP
```

The entry for the BMPTYPE option must reflect the name of the LogoUnloader control group, in this case *MTIF*.

If you want to use a different name for the TIFF unloader, you must register it as a new TIFF unloader. For example, if you choose to use *MTF* to represent the multipage TIFF unloader, you need to register it as shown here:

```
< LogoUnloaders >
  LogoUnloader = MTF
< LogoUnloader:MTF >
  Desc    = Multipage TIFF file
  Ext     = .TIF
  Func    = LOGUnloadMultiTifFiles
  Module  = LOGW32.DLL
< PrtType:BMP >
  BMPTYPE = MTF
...
```

Note Please note that this applies to all unloaders.

Be sure to specify a location in the Device option where you want the Bitmap Printer Driver to send the output. If you leave this option blank or NULL, the system defaults to the current location.

Note The other INI options are discussed in *Setting Up INI Files* on page 59.

FOR DOCUMAKER SERVER

When using the Bitmap Print Driver with Documaker Server, you need to set up your system to produce one TIFF file that contains the entire form set. See *Creating a Single TIFF File for Each Form Set* on page 69 for a discussion of these options.

Note When you run Documaker Server, there is always a file name passed in so the Bitmap Print Driver driver does not generate a file name for each per page based on the page number. Instead, it tries to write to the same file again and again and you end up with only the last page because the previous pages were overwritten.

In addition, for Documaker Server the following control groups and callback function must be present to create multipage TIFF files in a batch environment:

Generating file names

If you are using multiple or single step processing and you want to generate 8- or 16 byte file names, include these options:

```
< Print >
  CallbackFunc = MultiFilePrint
  MultiFileLog = data\datlog.dat
```

See the discussion of the Port option in *Documaker Server INI Options* on page 63 for additional information on the setup you need to produce multipage TIFF files. For example, if the Port option is set as shown here:

```
< Printer1 >
  Port = BPDBat1.tif
```

The first file name generated will be *BPDBat1.tif*. The second and subsequent file names will be based on the first four bytes of your entry for the Port option plus a 4-byte sequence number — *BPDB0002*, *BPDB0003*, and so on.

To avoid overwriting files, if you are using single-step processing and the single page tiff or other bitmap unloader, set all of the Port options to the same four characters, as discussed in *Documaker Server INI Options* on page 63. Here is an example:

```
< Printer1 >
  Port = DATA\~TMP.TIF
```

For the first transaction, the first file name generated will be:

~Tmp.tif

Subsequent file names will look like this:

~Tmp00000002.tif
~Tmp00000003.tif

and so on. For the second and subsequent transactions, the system generates names similar to these:

~Tmp0002.tif
~Tmp000200000002.tif
~Tmp000200000003.tif
...
~Tmp0003.tif
~Tmp000300000002.tif
~Tmp000300000003.tif
...

Note You cannot produce single page TIFF files on z/OS because of the long file name requirements.

Generating Unique File Names

If you are using single-step processing and you want to generate unique file names, include these options:

```
< PrintFormSet >
  MultiFilePrint= Yes
  MultiFileLog = data\datlog.dat
```

You can use the PrintFormSet control group to produce a 46-byte unique output file name. This is typically preferred by Docupresentment processing. when you use the PrintFormSet control group to turn off the callback function to avoid confusion. See the Docupresentment documentation for more information.

For example, using options set up as shown here:

```
< PrtType:BPD >
  BmpType      = TIF
...
< Printer >
  PrtType      = BPD
< Printer1 >
  Port         = DATA\BMPBAT1.TIF
...
< PrintFormSet >
  MultiFilePrint= Yes
  MultiFileLog = data\datlog.dat
```

The Bitmap Print Driver will generate unique file names similar to these:

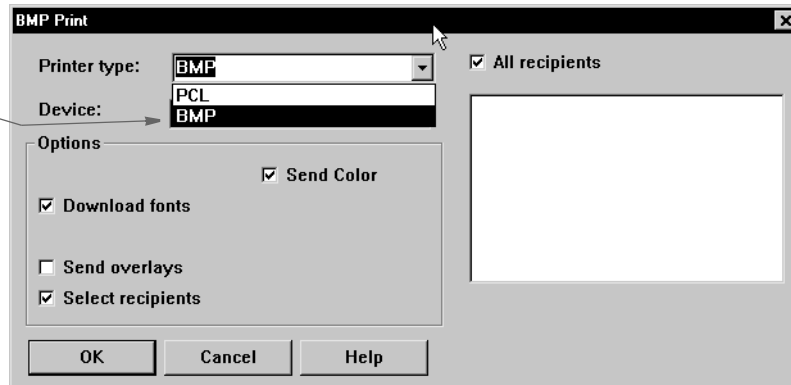
```
0zelzqDAmW8VnbnYZYSUkoeMt492V-iWeDOCGm9Dd7K5x.TIF
0zA3XvAkGr0VyoVQ1hwBPFV-OsAUc-uFZFIFwORGjLmTg.TIF
0ylx9_kotjF6--_xOfzx0-Ecu-kxnw-KzACdzpjbhBJ1P.TIF
0ylx9_kotjF6--_xOfzx0-Ecu-kxnw-KzACdzpjbhBJ1P00000002.TIF
0ylx9_kotjF6--_xOfzx0-Ecu-kxnw-KzACdzpjbhBJ1P00000003.TIF
0xBUheR9U5nI6kxsyig_4lld4imtj7BV2ygt2riLWRs9X.TIF
0WBYZ5U3Cq1i3weBZYwJ9b13zngcYzwhzxkmu9xCiB1fk.TIF
0WBYZ5U3Cq1i3weBZYwJ9b13zngcYzwhzxkmu9xCiB1fk00000002.TIF
```


SELECTING THE BITMAP PRINT DRIVER

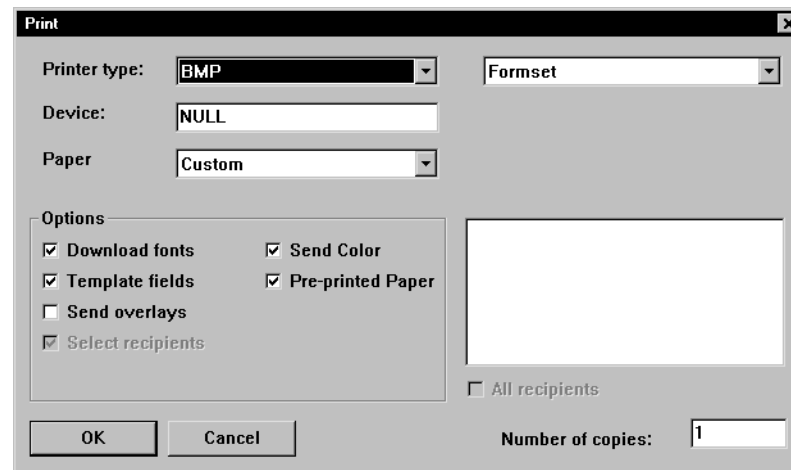
Selecting the Bitmap Print Driver is just like selecting any other print driver once you have installed the driver and set up the necessary INI options. The steps vary slightly, depending on the application you are using.

For instance, with Documaker Desktop, you simply open the form set you want to print then choose the Print option from the File menu.

Select BMP as the printer type and enter NULL in the Device field.



From Documaker Studio the steps are basically the same, you open the form set or section (FAP file) you want to print and choose File, Print. Here is an example of the Print window:



For Documaker Server, you simply set up the INI options. The following option tells the GenPrint program which printer driver to use:

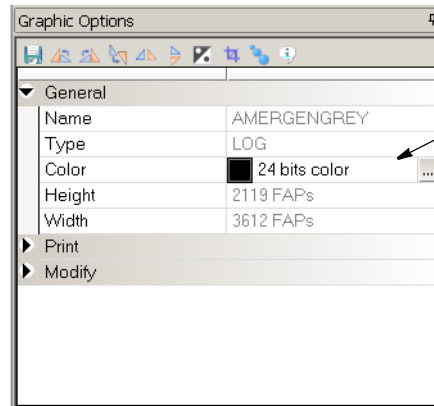
```
< Printer >
  PrtType   = BMP
```

For more information, see *Setting Up INI Files* on page 59.

ADDITIONAL CONSIDERATIONS

Keep in mind...

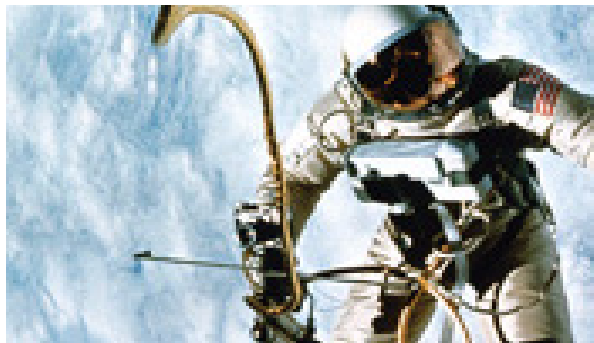
- The Bitmap Print Driver can produce 1-bit, 8-bit, and 24-bit bitmaps. The number of bits per pixel generated is based on the highest number of bits used. For example, if the page includes a 1-bit bitmap, an 8-bit bitmap, and a 24-bit bitmap, the Bitmap Print Driver will produce a 24-bit bitmap of the page.
- You can use the Graphics manager in Documaker Studio to find out the type of color bitmap you have.



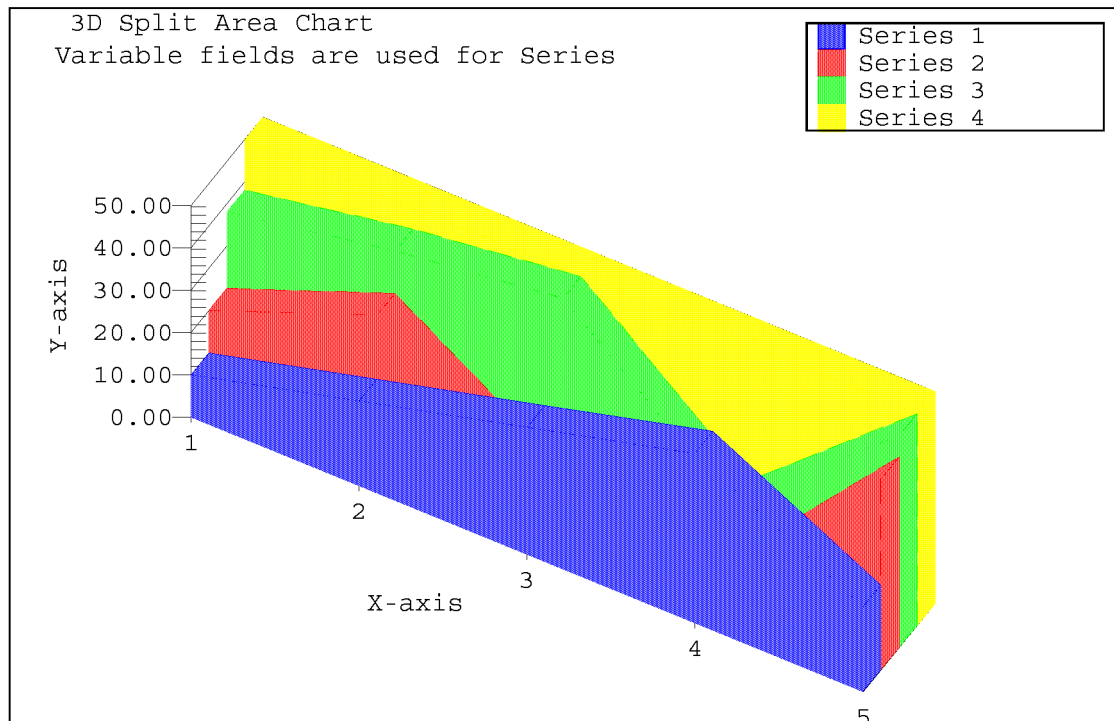
Studio shows you the type of color bitmap you have in the graphics' Color field

Note Other Oracle Insurance print drivers may accept bitmaps which are not 1-bit or 24-bit graphics and produce different results.

- If the page includes two 8-bit color bitmaps with different color palettes, the color palette of the first 8-bit bitmap encountered is used as the base color palette. This can cause the second 8-bit bitmap to look different from its original color. Here is an example:



- If a page has more than one color, such as a color chart, the Bitmap Print Driver creates a single 8-bit bitmap. Here is an example:



Chapter 4

Using the EPT Print Driver

This chapter provides information about Documaker's EPT Print Driver. This print driver is used to create print-ready files you can email to another user.

This chapter includes the following topics:

- *Overview* on page 78
- *Creating EPT Print Files for Documaker Desktop* on page 79
- *Creating EPT Print Files for Documaker Server* on page 84
- *Creating PDF Print Files* on page 86
- *Sending Emails in Multipart MIME Format* on page 87
- *Including Attachments with MPM Files* on page 90
- *Overriding Attached Files* on page 96
- *Using Email Aliases* on page 97

OVERVIEW

The system lets you create print-ready files that you can email to another user. The recipients can immediately print the file.

These print-ready files can be in a variety of formats, including:

- RTF
- PDF
- MPM

Like the other print drivers, the EPT Print Driver (EPTLIB) uses INI options to tell the system how to use the driver. Unlike other print drivers, the EPT Print Driver is essentially just a *wrapper* for a real print driver, so its INI options must include a reference to the actual print driver used to create the print-ready file, such as the PDF Print Driver (PDFLIB) or PCL Print Driver (PCLLIB).

There are also INI options for email processing, in addition to the regular email INI options.

CREATING EPT PRINT FILES FOR DOCUMAKER DESKTOP

The INI options for the EPT Print Driver are as follows:

```
< Printers >
    PrtType = EPT
```

Option	Description
--------	-------------

PrtType	Enter EPT. This option lets the system know that EPTLIB is a print driver so it will include it on the Print window when you print from Documaker Desktop.
---------	--

Use the PrtType:EPT control group to further customize the EPT Print Driver. For instance, you can add subject and message information and use the email address book when printing from Documaker Desktop. This lets you select print, choose the form set (form or page), then select the EPT print type.

The system then displays the email address book. You select the recipients and a window appears into which you can enter the subject and message text. You then choose to send or cancel the message.

Here is an example of the INI options you can use:

```
< PrtType:EPT >
    Class           = EPT
    Device          =
    PrtType         = RTF
    InitFunc        = EPTInit
    Module          = EPTW32
    PrintFunc       = EPTPrint
    RecipFunc       = CSTSetMailRecipgvm
    RecipMod        = CSTW32
    TermFunc        =
    DownloadFonts   =
    FileName        = EPTFILE.RTF
    KeepFile        = No
    Message         = Please respond as soon as possible. Thanks.
    PageNumbers     = Yes
    PrePrintedPaper = No
    SendColor       = Yes
    StreamBufferSize =
    Subject         = New Application
    Recipient       =
```

Option	Description
--------	-------------

Class	Use to specific the printer classification, such as AFP, EPT, PCL, XER, PST, or GDI. For the EPT Print Driver, enter EPT. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. These internal functions check the Class option to make sure the correct printer is available before continuing. The default is the first three characters of the Module option settings.
Device	This setting is ignored by the GenPrint program but is used when printing from GUI Documaker applications like Documaker Studio and Documaker Desktop.
PrtType	This option tells the EPT Print Driver which print driver to use to create the print-ready file. The default is RTF (Rich Text Format), which tells the system to use the RTF Print Driver.
InitFunc	Enter EPTInit. This tells the system to use a special initialization function called EPTInit which is located in EPTW32.DLL.

Option	Description
Module	This option tells the system to load the module which contains the print driver. The default is EPTW32.
PrintFunc	This option tells the system which print function to use. The default is EPTPrint.
RecipMod RecipFunc	The RecipMod and RecipFunc options tell the system which module and function to use to determine the recipient. Omit these options and the system uses the EPT Print Driver's default recipient function. For more information, see <i>Using the RecipMod and RecipFunc Options</i> on page 82.
TermFunc	This option tells the system to use a specified termination function. The default is EPTTerm, which is located in EPTW32.DLL
DownloadFonts	Enter No if you do not want to download fonts. The default is Yes. Many print options, such as the SendColor option, are set before the system calls the EPT Print Driver. These options are passed to another print driver which creates the file to be attached to the email. To have the EPT Print Driver use the DownloadFonts setting from another print driver (such as PDF), set your PrtType:EPT control group to look like this: <pre>< PrtType:EPT > PrtType = PDF DownloadFonts = [PrtType:PDF] DownloadFonts =</pre>
FileName	Use this option to provide the name of the output file to create. This is only used if the Device field is empty in the Print window (the batch file name is used for GenPrint). If the Device field and the FileName options are omitted, a temporary file name is used. Use a file name with an extension that matches the print driver type, such as RTF or PDF. For GenPrint, the file name is the name of the print batch.
KeepFile	Enter Yes if you want the system to keep the output file after it has been emailed. The default is No.
Message	Enter the text of the message you want to send.
PageNumbers	Enter Yes to have page numbers printed in the "Page X of Y" format. Many print options, such as the PageNumbers option, are set before the system calls the EPT print driver. These options are passed to another print driver which creates the file to be attached to the email. To have the email print driver use the PageNumbers setting from another print driver (such as PDF), set your PrtType:EPT control group to look like this: <pre>< PrtType:EPT > PrtType = PDF PageNumbers = [PrtType:PDF] PageNumbers =</pre> This way, if you change the PageNumbers option in the PrtType:PDF control group, those changes are automatically picked up in the PrtType:EPT control group.

Option	Description
PrePrintedPaper	<p>This option determines if the check box which lets you print or not print pre-printed objects appears on the Print window. It also lets you specify the default for this check box--checked or unchecked. You must add this option to the INI file if you want the check box to appear on the Print window.</p> <p>The default is to include the check box on the Print window, but leave it unchecked. All objects except fields can be designated as pre-printed on the object's Properties window in Studio.</p> <p>Many print options, such as the PrePrintedPaper option, are set before the system calls the email print driver. These options are passed on another print driver which creates the file to be attached to the email. To have the EPT Print Driver use the PrePrintedPaper setting from another print driver (such as PDF), set your PrtType:EPT control group to look like this:</p> <pre>< PrtType:EPT > PrtType = PDF PrePrintedPaper = [PrtType:PDF] PrePrintedPaper =</pre> <p>This way, if you change the PrePrintedPaper option in the PrtType:PDF control group, those changes are automatically picked up in the PrtType:EPT control group.</p>
SendColor	<p>Enter Yes to enable color printing. The default is No.</p> <p>Many print options, such as the SendColor option, are set before the system calls the EPT Print Driver. These options are passed on another print driver which creates the file to be attached to the email. To have the EPT Print Driver use the SendColor setting from another print driver (such as PDF), set your PrtType:EPT control group to look like this:</p> <pre>< PrtType:EPT > PrtType = PDF SendColor = [PrtType:PDF] SendColor =</pre> <p>This way, if you change the SendColor option in the PrtType:PDF control group, those changes are automatically picked up in the PrtType:EPT control group.</p>
StreamBufferSize	<p>Enter the number of bytes to use for buffering. You may use this option for performance tuning.</p> <p>The default is zero (0), which means to use the default buffer size.</p>
Subject	Enter the title of the message you want to appear on the Subject line of the email.
Recipient	<p>For the Recipient option, you can either include the actual email recipient or you can specify a field name where the system can go to look up the recipient. Here are some examples:</p> <pre>Recipient = Jim Doe</pre> <p>This example sends the email to an internal email recipient.</p> <pre>Recipient = jdoe@oracle.com</pre> <p>This example sends the email to an Internet email address.</p> <pre>Recipient = Fieldname:ADDRESS2</pre> <p>This tells the system to use the text in the ADDRESS2 field.</p> <p>If the email system cannot resolve recipients, or if you leave the Recipient option blank, an email address window appears so you can select an email address from the address book.</p> <p>The field lookup is a feature of the default recipient function in EPTLIB, which you can replace using these INI options:</p> <pre>RecipMod = CSTW32 RecipFunc = CSTSetMailRecip</pre>

Using the RecipMod and RecipFunc Options

The CSTSetMailRecip function displays a window which shows the subject and message text and lets you edit this text. This window also lets you provide the email recipient for Documaker Desktop.

Documaker Server lets you use these functions to set up recipients:

```
RecipMod   = CUSW32
RecipFunc  = CUSSetMailRecip
```

or

```
RecipMod   = CUSW32
RecipFunc  = CUSSetMailRecipGVM
```

Function	Description
CUSSetMailRecip	This function finds the print recipient and looks up the recipient in the RECIP_MAIL control group to get the email address of the recipient. Here is an example: <pre>< Recip_Mail > Agent = myagent@sampco.com Company = support@sampco.com</pre>
CUSSetMailRecipGVM	This function finds the recipient in a global variable, the name of which is defined in this INI option: <pre>< PrtType:EPT > Recipient = EAddress</pre> <p>Instead of using <i>EAddress</i> as the recipient name, the system uses it as the variable name to look up to find the recipient name. This global variable can have any name.</p>

Using the EPTSetRecipFunc Function

To set the recipient function without using the RecipMod and RecipFunc INI options, use the EPTSetRecipFunc function:

```
EPTRECIPFUNC _VMMAPI EPTSetRecipFunc(EPTRECIPFUNC newfunc);
```

Call it with the address of the recipient function:

```
EPTSetRecipFunc(func);
```

The EPTSetRecipFunc function returns the previous installed function, which can be used to set it back.

Here is an example:

```
< PrtType:EPT >
  RecipFunc = CSTSetMailRecipgvm
```

Using Custom Code

If you want to write your own custom recipient function, the recipient function you create must use this syntax:

```
DWORD _VMMAPI EPTDefSetRecipient(VMMHANDLE objectH,
                                char FAR * recip,
                                size_t len);
```

Parameter	Description
objectH	The object being printed (form set, form, or page)

Parameter	Description
recip	The recipient buffer
len	Length of the buffer, currently 80 characters

The return value should be Success or Failure. If the process fails, the message is not sent and Failure is returned from EPTPrint.

CREATING EPT PRINT FILES FOR DOCUMAKER SERVER

Note See the [Documaker Desktop Administration Guide](#) for information on setting up email support in Documaker Desktop.

To create EPT print files for Documaker Server, Set up your INI options as shown here:

```
< Printer >
  PrtType    = EPT
< PrtType:EPT >
  Module     = EPTW32
  PrintFunc  = EPTPrint
  InitFunc   = EPTInit
  TermFunc   = EPTTerm
```

These options tell the system which functions to call to execute the printing process.

```
PrtType = RTF
```

The PrtType option tells the EPT Print Driver which real print driver to use to create the print-ready file. If omitted, it defaults to the RTF Print Driver (Rich Text Format).

```
FileName = EPTFILE.RTF
```

This option provides the name of the output file to create. For GenPrint, the file name is the name of the print batch.

```
KeepFile = No
```

The KeepFile option tells the EPT Print Driver whether to keep the output file after it has been emailed. The default is No.

```
< Print >
  CallbackFunc = MultiFilePrint
  MultiFileLog = data\rtflog.dat
```

These options tell the system to divide large RTF files into smaller RTF files. If you omit these options, you will be able to view the first transaction, but not the following ones.

The RTFLOG.DAT file stores the information that defines which RTF file contains which transaction for which batch.

```
Recipient = Email Recipient
Subject   = File from Documaker Server
Message   = See the attached PDF file.
```

Use these INI options to set mail settings for the EPT Print Driver. The Subject and Message options specify the Subject line and Message text for the email message.

For the Recipient option, you can include the actual email recipient or you can specify a field name where the system can look up the recipient. Here are some examples:

```
Recipient = Jim Doe
```

This example sends the email to an internal email recipient.

```
Recipient = jdoe@oracle.com
```

This example sends the email to an Internet email address.

```
Recipient = Fieldname:ADDRESS2
```

This tells the system to use the text in the ADDRESS2 field.

If the email system cannot resolve recipients, or if you leave the Recipient option blank, an email address window appears so you can select an email address from the address book. The field lookup is a feature of the default recipient function in the EPT Print Driver, which you can replace using these INI options:

```
RecipMod = CSTW32
RecipFunc = CSTSetMailRecip
```

These options tell the system which module and function to use to determine the recipient. Omit these options and the system uses the default recipient function.

Documaker Server lets you use these functions to set up recipients:

```
RecipMod = CUSW32
RecipFunc = CUSSetMailRecip
```

or

```
RecipFunc = CUSSetMailRecipGVM
```

Function	Description
CUSSetMailRecip	This function finds the print recipient and looks up the recipient in the Recip_Mail control group to get the email address of the recipient. Here is an example: <pre>< Recip_Mail > Agent = myagent@sampco.com Company= support@sampco.com</pre>
CUSSetMailRecipGVM	This function finds the recipient in a global variable, the name of which is defined in this INI option: <pre>< PrtType:EPT > Recipient = EAddress</pre> <p>Instead of using <i>EAddress</i> as the recipient name, the system uses it as the variable name to look up to find the recipient name. This global variable can have any name.</p>

The recipient functions have the following syntax:

```
DWORD _VMMAPI EPTDefSetRecipient(VMMHANDLE objectH,
                                char FAR * recip,
                                size_t len);
```

Parameter	Description
objectH	The object being printed (form set, form, or page)
recip	The recipient buffer
len	Length of the buffer, currently 80 characters

The return value should be Success or Failure. If the process fails, the message is not sent and Failure is returned from the EPT Print Driver.

To set the recipient function without using INI options, use the EPTSetRecipFunc function:

```
EPTRECIPFUNC _VMMAPI EPTSetRecipFunc(EPTRECIPFUNC newfunc);
```

Call it with the address of the recipient function:

```
EPTSetRecipFunc(func);
```

The EPTSetRecipFunc function returns the previous installed function, which can be used to set it back.

CREATING PDF PRINT FILES

If you are creating PDF files for use with the EPT Print Driver, use these INI options:

```
< Printers >
  PrtType    = PDF
< PrtType:PDF >
  Module      = PDFW32
  PrintFunc   = PDFPrint
```

Keep in mind that when the PDF Print Driver is called from the EPT Print Driver, the current printer control group remains PrtType:EPT, not PrtType:PDF. Therefore, unless you add PDF-specific options, the system uses the INI settings it finds for PrtType:EPT.

Many print options, such as the DownloadFonts option, are set before the system calls the EPT Print Driver, which then redirects the print to another driver. So, to have the system use the correct PDF options, set your PrtType:EPT control group as shown here:

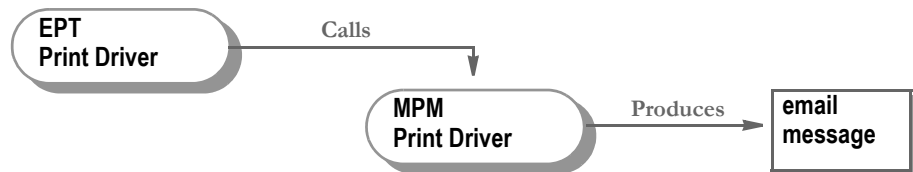
```
< PrtType:EPT >
  PrtType      = PDF
  DownloadFonts = [PrtType:PDF] DownloadFonts =
  SendColor     = [PrtType:PDF] SendColor =
```

This way if you change the options in the PrtType:PDF control group, those changes are automatically picked up in the PrtType:EPT control group.

SENDING EMAILS IN MULTIPART MIME FORMAT

Documaker applications can produce multipart MIME (MPM) output using the MPM Print Driver (see *Using the MPM Print Driver* on page 141 for more information). The EPT Print Driver sends the MPM output as email message text via the SMTP email service.

First, the EPT Print Driver calls the MPM Print Driver to generate email output in MPM format (HTML with the plain text option).



Then, the EPT Print Driver calls the SMTP email (SMM) driver to send the MPM output as an email body.



The MPM format generally has two parts: plain text and HTML, mail clients can read the email in either plain text or HTML format. Here is an example.

```

From: "Senders Name" <sender@sendersdomain.com>
To: "Recipient Name" <somerecipient@recipientdomain.com>
Message-ID: <5bec11c119194c14999e592feb46e3cf@sendersdomain.co m>
Date: Sat, 24 Sep 2010 15:06:49 -0400
Subject: Sample Multipart MIME
  
```

```

MIME-version: 1.0
Content-type: multipart/mixed;
boundary="MIMEBoundary84750C262B234972B66CDA11704EC46B"
  
```

This is message is sent in multipart MIME format.

```

--MIMEBoundary84750C262B234972B66CDA11704EC46B
Content-type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
  
```

This is the message body in plain text.

```

--MIMEBoundary84750C262B234972B66CDA11704EC46B
Content-type: text/html; charset="utf-8"
Content-Transfer-Encoding: 8bit
  
```

```

<html>
<head><title>A HTML email</title></head>
<body>This is the message body in HTML.</body>
</html>
  
```

```

--MIMEBoundary84750C262B234972B66CDA11704EC46B--
  
```

Use these INI options to set up the EPT Print Driver to enable the MPM Print Driver to produce MPM output and use the SMM email driver to send MPM output as email message body via SMTP:

```
< PrtType:EPT >
    MsgPrtType= MPM
    MessageFile= .\data\mpm.htm
```

Option	Description
MsgPrtType	Enter MPM to use the MPM Print Driver to produce Multipart MIME output.
MessageFile	(Optional) Enter the name and path of the message file which will be produced by the printer driver you specified in the MsgPrtType option. Here is an example: .\data\mpm.htm

Examples

Here is an example of how you would set up the EPT Print Driver to generate MPM output and send it as email message via SMTP:

```
< Printer >
    PrtType    = EPT
< Printers >
    PrtType    = EPT
    PrtType    = MPM
< PrtType:EPT >
    InitFunc   = EPTInit
    TermFunc   = EPTTerm
    Module     = EPTW32
    PrintFunc   = EPTPrint
    MsgPrtType= MPM
    MessageFile = .\data\mpm.htm
    Message    = Send transaction via email
    PrtType    = PDF
    FileName   = .\data\x.pdf
    KeepFile   = Yes
    Recipient  = jane.doe@example.com
    Subject    = EPT test
< Mail >
    MailType   = SMTP
< MailType:SMTP >
    AltFrom    = john.doe@example.com
    From       = john.doe@example.com
    MailFunc   = SMMMail
    Module     = SMMW32
    Name       = SMMMail
    Port       = 25
    ReplyTo    = john.doe@example.com
    Server     = mail.example.com
< SMTP_Attachment >
    Content-Type = "text/plain; charset=ascii"
```


Here is an example of how to set up the MPM Print Driver to generate email body text in MPM format:

```
< PrtType:MPM >
Device           = .\data\mpm.htm
Module           = MPMW32
PrintFunc        = MPMPrint
SendColor        = Yes, Enabled
SelectRecipients = Yes, Enabled
TemplateFields   = Yes, Enabled
PageBorder       = Yes
CreatePlainText  = Yes
DirLinks         = Yes
ForcePrintInColor = Yes
HR               = size=2 color=red width=100%
BitmapHTTP       = http://example.com/public_html
BitmapPath       = \\example.com\public_html\
```

Here is an example of how you would set up the PDF Print Driver to generate an email attachment in PDF format:

```
< PrtType:PDF >
Device           = .\data\x.pdf
Module           = PDFOS2
PrintFunc        = PDFPrint
DownloadFonts    = Yes
SendColor        = Yes, Enabled
SelectRecipients = Yes, Enabled
TemplateFields   = Yes, Enabled
Bookmark         = Yes, Page
Resolution       = 300
```

INCLUDING ATTACHMENTS WITH MPM FILES

You can use the EPT print driver to create a Multipart MIME (MPM) email that contains embedded PDF, RTF, PCL, HTM, XML, PST, and bitmap (BMP, JPG, PNG, TIF, and so on) data in the body of the message or as an attachment. For example, this lets you create an email file that contains an attached PDF document and then send it from an email delivery utility such as Document Factory.

Note The EPT print driver originally created an attachment and a simple message for an SMTP email client and then prompted the client to send the email. You could also attach a separate print file (PDF, PCL, and so on). The EPT print driver was enhanced in version 11.5 to also create a Multipart MIME (MPM) format output file via the MPM print driver.

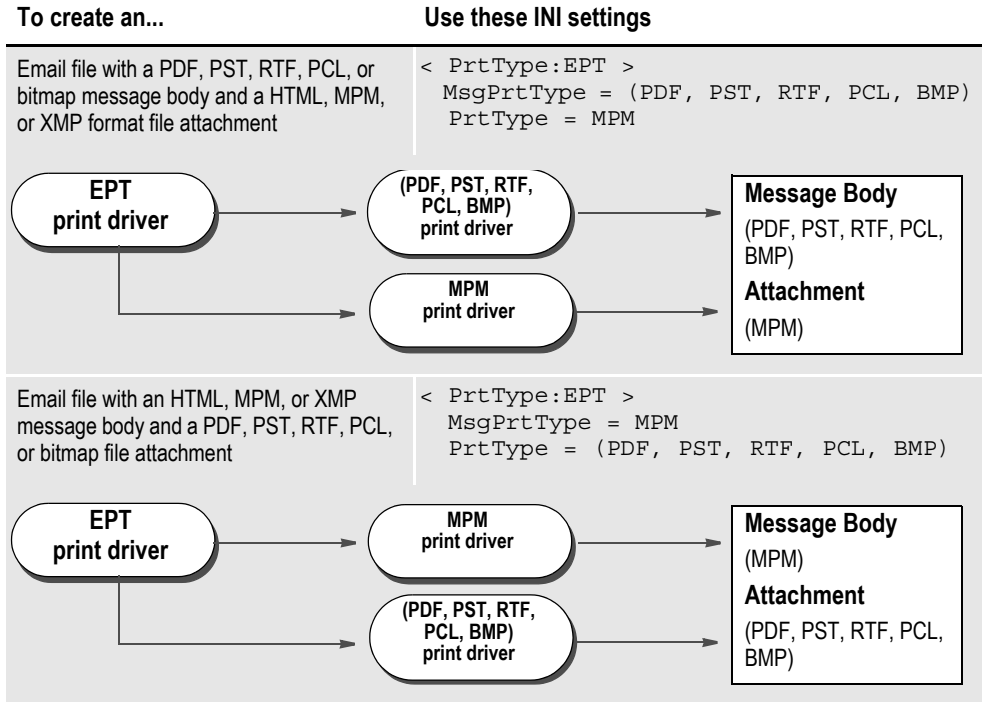
In version 12.0 patch 01, the EPT print driver was enhanced to let you embed print files into MPM email files. You can then send the email file via Document Factory or open it via Mozilla Thunderbird (version 3.1 or higher). If you open the MPM email file in Thunderbird, the embedded file appears as an attached file that you can open, save, or print, just as if it were a separate file.

For more information, see *Using the MPM Print Driver* on page 141.

The EPT print driver can use other print drivers to produce an email message in one print type and an attachment in another print type. This table explains the types of emails and the types of attachments it can generate:

To create an...	Use these INI settings
MPM message body with embedded graphics but no embedded print file	<pre>< PrtType:EPT > MsgPrtType = MPM PrtType = (blank)</pre>
<pre> graph LR EPT([EPT print driver]) --> MPM([MPM print driver]) MPM --> Result[Message Body (MPM format with embedded graphics) Attachment (none)] </pre>	
Email file with a message body and file attachment in HTML, MPM, or XMP format	<pre>< PrtType:EPT > MsgPrtType = (HTM, MPM, XMP) PrtType = (HTM, MPM, XMP)</pre>
<pre> graph LR EPT([EPT print driver]) --> D1([HTM, MPM, XMP print driver]) EPT --> D2([HTM, MPM, XMP print driver]) D1 --> Result[Message Body (HTM, MPM, XMP) Attachment (HTM, MPM, XMP)] D2 --> Result </pre>	

*1 - Encoded base64 data



*1 - Encoded base64 data

Note The EPT print driver lets you embed print file attachments but does not then signal the email client to send the email.

The email file contains the header information at the top, the message body, and the embedded print file toward the bottom of the email, as illustrated here:

Header	To: leslie.lee@oracle.com Reply-To: michael.norcon@oracle.com From: michael.norcon@oracle.com Subject: Company picnic
Body	Here are some dates and venues for the company picnic:
Embedded print file	Content-Type: Application/pdf; charset="utf-8" Content-Transfer-Encoding: base 64 Content-Disposition: attachment;filename=picnic.pdf (attachment content)

If you open the email in an email client, such as Mozilla Thunderbird, the embedded file appears as an attachment you can open, save, or print just as if it were a separate file.

Note Depending on your email client and the format of the email you create, you may have to adjust the configuration of the email client for it to properly display the email. Refer to the documentation for your email client for more information.

INCLUDING ATTACHMENTS WITH MPM FILES

You can use the EPT print driver to create a Multipart MIME (MPM) email that contains embedded PDF, RTF, PCL, HTM, XML, PST, and bitmap (BMP, JPG, PNG, TIF, and so on) data in the body of the message or as an attachment. For example, this lets you create an email file that contains an attached PDF document and then send it from an email delivery utility such as Document Factory. The content of the attachment is the same content as displayed in the email message body.

Note The EPT print driver originally created an attachment and a simple message for an SMTP email client and then prompted the client to send the email. You could also attach a separate print file (PDF, PCL, and so on). The EPT print driver was enhanced in version 11.5 to also create a Multipart MIME (MPM) format output file via the MPM print driver.

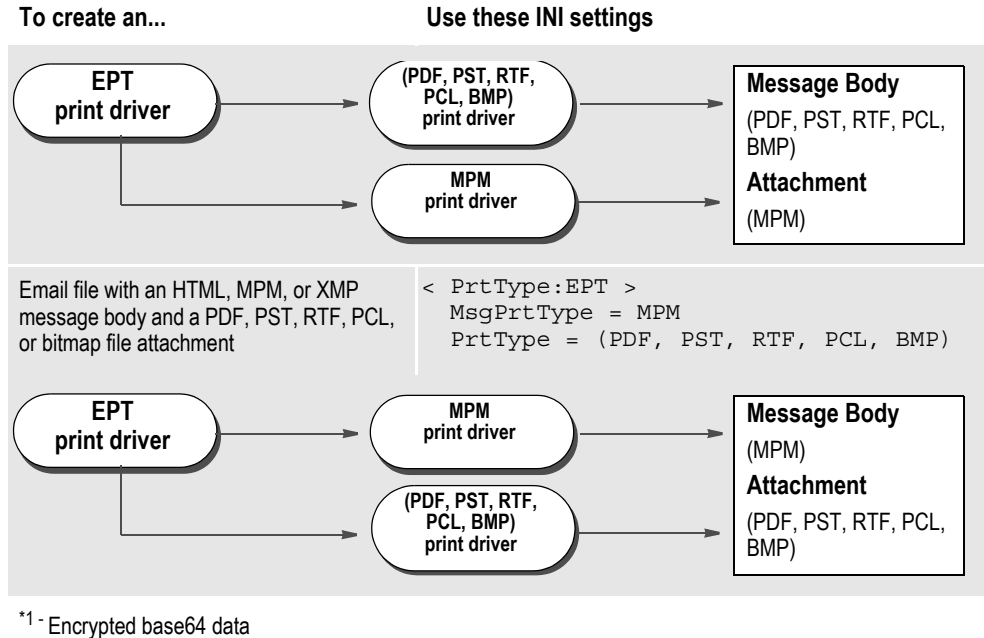
In version 12.0 patch 01, the EPT print driver was enhanced to let you embed print files into MPM email files. You can then send the email file via Document Factory or open it via Mozilla Thunderbird (version 3.1 or higher). If you open the MPM email file in Thunderbird, the embedded file appears as an attached file that you can open, save, or print, just as if it were a separate file.

For more information, see *Using the MPM Print Driver* on page 141.

The EPT print driver can use other print drivers to produce an email message in one print type and an attachment in another print type. This table explains the types of emails and the types of attachments it can generate:

To create an...	Use these INI settings
Email file with a message body, but with no attachment	<pre>< PrtType:EPT > MsgPrtType = MPM PrtType = (blank)</pre>
<pre> graph LR EPT([EPT print driver]) --> MPM([MPM print driver]) MPM --> Output[Message Body (MPM format with embedded graphics) Attachment (none)] </pre>	
Email file with a message body and file attachment in HTML, MPM, or XMP format	<pre>< PrtType:EPT > MsgPrtType = (HTM, MPM, XMP) PrtType = (HTM, MPM, XMP)</pre>
<pre> graph LR EPT([EPT print driver]) --> Driver1([HTM, MPM, XMP print driver]) EPT --> Driver2([HTM, MPM, XMP print driver]) Driver1 --> Output[Message Body (HTM, MPM, XMP) Attachment (HTM, MPM, XMP)] Driver2 --> Output </pre>	
Email file with a PDF, PST, RTF, PCL, or bitmap message body and a HTML, MPM, or XMP format file attachment	<pre>< PrtType:EPT > MsgPrtType = (PDF, PST, RTF, PCL, BMP) PrtType = MPM</pre>

*1 - Encrypted base64 data



Note The email file is distributed by the email server configuration. For more information see the [Documaker Administration Guide](#) or the [Documaker Enterprise Administrator Guide](#). For information on using the XMP library to create an XML file using IDS, see the [Internet Document Server Guide](#).

The email file contains the header information at the top, the message body, and the embedded print file toward the bottom of the email, as illustrated here:

Header	To: leslie.lee@oracle.com Reply-To: michael.norcon@oracle.com From: michael.norcon@oracle.com Subject: Company picnic
Body	Here are some dates and venues for the company picnic:
Embedded print file	Content-Type: Application/pdf; charset="utf-8" Content-Transfer-Encoding: base 64 Content-Disposition: attachment; filename=picnic.pdf (attachment content)

If you open the email in an email client, such as Mozilla Thunderbird, the embedded file appears as an attachment you can open, save, or print just as if it were a separate file. Here is an example:

Note Depending on your email client and the format of the email you create, you may have to adjust the configuration of the email client for it to properly display the email. Refer to the documentation for your email client for more information.

Click here to open, save, or print the attachment.



Creating the file on disk

To tell the EPT print driver to take a message and attachment in MPM format and create a file on disk which has embedded HTM, PCL, PDF, PST, RTF, XML or bitmap (JPG, PNG, TIF, and so on) data include the WriteToFile option:

```
< PrtType:EPT >
  WriteToFile =
```

Option	Description
WriteToFile	<p>Specify the path and file name, as shown here:</p> <pre>path\fname.ext</pre> <p>Where <i>fname</i> denotes the file name and <i>ext</i> the file extension.</p> <p>If you do not specify a path, the EPT print driver writes the MPM data to the current working directory.</p> <p>If you leave <i>fname</i> empty, the EPT print driver creates a globally unique identifier (GUID) file name for you. The default file extension is <i>.mht</i>. You can, however, specify another text type extension, such as <i>.eml</i> (for email, such as Microsoft Outlook Express), <i>.mpm</i> (for normal MPM) or <i>.txt</i> (for plain text).</p>

You can then send the file via Document Factory or open it in Mozilla Thunderbird.

Naming the message files

The FileName and MessageFile options, by default, use a globally unique identifier (GUID) file name with an extension which is defined in the PrtType and MsgPrtType options. You can also specify an extension and leave the file name blank, as shown here:

```
< PrtType:EPT >
  PrtType      = PDF
  FileName     = .\data\.pdf
  MsgPrtType   = XMP
  MessageFile  = .\data\.xml
```

Here is an example:



Creating the file on disk

To tell the EPT print driver to take a message and attachment in MPM format and create a file on disk which has embedded HTM, PCL, PDF, PST, RTF, XML or bitmap (JPG, PNG, TIF, and so on) data include the WriteToFile option:

```
< PrtType:EPT >
  WriteToFile =
```

Option	Description
WriteToFile	<p>Specify the path and file name, as shown here:</p> <pre>path\fname.ext</pre> <p>Where <i>fname</i> denotes the file name and <i>ext</i> the file extension.</p> <p>If you do not specify a path, the EPT print driver writes the MPM data to the current working directory.</p> <p>If you leave <i>fname</i> empty, the EPT print driver creates a globally unique identifier (GUID) file name for you. The default file extension is <i>.mht</i>. You can, however, specify another text type extension, such as <i>.eml</i> (for email, such as Microsoft Outlook Express), <i>.mpm</i> (for normal MPM) or <i>.txt</i> (for plain text).</p>

You can then send the file via Document Factory or open it in Mozilla Thunderbird.

Naming the message files

The FileName and MessageFile options, by default, use a globally unique identifier (GUID) file name with an extension which is defined in the PrtType and MsgPrtType options. You can also specify an extension and leave the file name blank, as shown here:

```
< PrtType:EPT >
  PrtType      = PDF
  FileName     = .\data\.pdf
  MsgPrtType   = XMP
  MessageFile  = .\data\.xml
```

OVERRIDING ATTACHED FILES

Keep in mind that the EPT Print Driver can use the `FSRSetFileAttachment` API.

This lets you create custom hooks to override the attached file and handle situations where you need to remove the attached file but still send the message.

USING EMAIL ALIASES

Multiple recipient addresses are not supported with the EPT Print Driver. If you need to send an email to, for instance, all agents, use an Email Application Server, such as Microsoft Exchange (MailType = MSM) or ccMail (MailType = CCM). With these products you can define an alias to represent a group of email addresses. You cannot set the MailType option to SMTP unless your SMTP server understands aliases.

Email Application Servers usually run on top of an SMTP service and let you manage email messaging more efficiently. When using an application such as Exchange, you can create a group (such as *TestGroup*) and you can specify the group name when you specify the Recipient option.

For example, if you set the MailType option to MSM in the Mail control group and you have this defined for the Recipient option:

```
< PrtType:EPT >  
  Recipient = TestGroup
```

This option is sent to the Exchange server which converts the alias (*TestGroup*) into its SMTP equivalent value, such as a list of email address similar to this:

```
hbean@oracle.com;jgaramond@oracle.com;tbottle@oracle.com...
```

The result is a message sent to the entire group represented by *TestGroup*.

Note To use this feature, you must also set up email-related INI options. These options are discussed in the [Documaker Desktop Administration Guide](#).

Chapter 5

Using XML Print Driver

The XMPLIB library allows you to use Documaker RP to create Docucorp Standard XML output. You can unload Docucorp Standard XML output from GenData or GenPrint programs (using the PrintFormset rule).

Here is an example of the INI setup this feature requires:

```
< Printers >
PrtType = XMP

< PrtType:XMP >
Module = XMPW32
PrintFunc = XMPPrint
```

Note No other INI options are needed.

Adding additional content to XML output files

To tell the XMP Print driver to write additional static section (FAP) content, font and attribute information to the XML output files include the Filter option

```
<PrtType:XMP>
Filter = Yes | No
```

Note **Yes** (default): Standard XML tag and attribute information is written to the XML output files.

No: Additional static section (FAP) content, font and attribute information is written to the XML output files.

To include fields not populated with data in the XML output files

To tell the XMP Print driver to omit field names which have not been populated with data from the XML output file

```
<PrtType:XMP>
OutputEmptyFields = Yes | No
```

Note **Yes**: If a field is not populated with data, the field name is still written to the output XML files.

No (default): If a field file is not populated with data, the field name is omitted from the output XML files.

When using with Documaker RP, it is recommended to use the MultiFilePrint functionality to create a separate XML file per transaction. If multiple XML files are written into the same file, the file will not load in an XML parser, browser, or editor.

Chapter 6

Using the GDI Print Driver

This chapter provides information about Documaker's GDI (Graphics Device Interface) Print Driver. This print driver is used on Windows implementations of Documaker software.

This chapter includes the following topics:

- *Overview* on page 102
- *Setting GDI INI Options* on page 104
- *Avoiding Problems with FAX Drivers* on page 107
- *Batch Printing to Files* on page 108

OVERVIEW

Oracle Insurance developed a Graphics Device Interface (GDI) print driver because it provides many opportunities for Windows platform users. For example, by using a GDI driver, you can now fax, since fax drivers can be installed into Windows as a GDI Windows printer driver.

Also, printing using GDI lets you print to printers that do not support any of the printer languages the system supports, such as inkjet printers. To make this driver even more useful, it includes the ability to scale output, which lets you shrink the printed output to the size of the paper.

The advantages of using the Graphics Device Interface (GDI) include:

- Ability to print to any printer attached via a Windows print driver
- Ability to print to any fax machine attached via a Windows print driver
- Ability to scale edge to edge forms to print within the printable area defined by the Windows print driver.

The disadvantages of using the Graphics Device Interface (GDI) include:

- Print quality is often poorer
- Inability to print a mixture of portrait and landscape forms
- Inability to print a mixture of simplex and duplex forms
- Inability to address the same printable area available when using our native print drivers.

Note If you do not specify the option for sending color to a GDI printer, the system converts color (4-, 8-, or 24-bit) graphics into monochrome before sending them to the printer driver. Depending on the bitmap, this conversion from color to monochrome may not yield acceptable results. Be sure to consider your printers capabilities when you are creating graphics.

If you elect to send color, including color graphics, to a GDI printer that does not support color, the printer driver determines what to do. Some ignore the color commands (printing in black), and some apply a gray-scale adjustment to the output to simulate the color changes. Some GDI printer drivers cannot accept color commands at all. If printing to your Windows-attached printer causes a program fault, or print failure, try turning off the Send Color option via the system's Print window and sending the output again.

How It Works

Most Windows applications print using the Windows GDI application programming interface. Essentially, the application uses commands similar to display commands to send print commands to the operating system. Windows, in turn, sends the commands to the currently installed Windows printer driver.

Note Printer manufacturers provide Windows printer drivers for their printers. These come on install disks from the manufacturer, or sometimes ship with Windows itself. Other types of drivers (such as fax drivers) can be installed as Windows printer drivers.

When a Windows program talks to the operating system using GDI, printer commands are not emitted in the native language of the printer by the program. The program *prints* to Windows, and Windows then *prints* to the installed printer driver. The printer driver then produces the native printer language commands, including the bitmap font definitions. If the printer driver belongs to a PCL printer, the print driver issues PCL commands, including fonts. In contrast, our PCL printer modules produce the PCL commands and fonts.

When you use our GDI driver, a Windows print driver will use the Windows screen fonts to print the document with its goal being to make the document look like it does on your screen.

Note In Documaker implementations, users typically decide what fonts they want to use and then install those fonts on the production printer. Documaker applications try to make the screen look like the printed output, not the other way around. Information from the production printer fonts is loaded into the font cross-reference file. The system uses this information to try to represent the printer fonts on screen. The system can also convert production printer fonts into PCL bitmap fonts. The PCL fonts the system produces look like the fonts used on your production printer.

GDI print quality, by definition, is based on the fonts used for display. The attributes which describe fonts in the font cross-reference file determine which screen fonts are used. The screen fonts used determine what you see on the screen and how GDI printed output will look.

So, the key to improving GDI print is to improve the fonts used in the display system. Some of this can be improved by making sure the font's character widths and family name is correct. There are INI options for improving the screen font substitutions, if names cannot be matched up.

For the best results, you should use exact matching screen fonts. The system comes with a set of TrueType fonts that match the printer fonts included with the system. Install and use these fonts for best results.

Note If you are instead working backward from existing production fonts, as is often the case, either an approximation must take place, or you have to find screen fonts built from the printer fonts.

SETTING GDI INI OPTIONS

You define the necessary printer options to print using the GDI printer driver. These options specify GDI output and are located in a `PrtType:xxx` control group, such as *PrtType:GDI*. Common GDI options are shown below, with default values in bold:

Option	Values	Description
Device	any file or device name	Not used by the GDI print driver.
Module	GDIW32	The name of the program module which contains the system's GDI print driver. See also the Class option. See also <i>Using Defaults for the Module and PrintFunc Options</i> on page 106.
PrintFunc	GDIPrint	The name of the program function that is the main entry point into the system's GDI print driver. See also <i>Using Defaults for the Module and PrintFunc Options</i> on page 106.
Resolution	300	Not used by the GDI print driver.
SendOverlays	Yes/ No	Not used by the GDI print driver.
OverlayPath	any directory	Not used by the GDI print driver.
OverlayExt	any file extension (OVL)	Not used by the GDI print driver.
PageNumbers	Yes/ No	Set to Yes to enable form or form set page numbering.
SendColor	Yes/ No Enabled/Disabled/ Hidden	Set to Yes to enable color printing. Enabled = Option appears in the Print window and is active (available to be checked). Disabled = Option appears in the Print window but is grayed out (not available to be checked). Hidden = Option does not appear in the Print window
DownloadFonts	Yes /No	Not used by the GDI print driver.
FitToWidth	Yes/ No	Scale pages to fit on the paper. This option will, if necessary, reduce the size of the page. It will not increase it.
TemplateFields	Yes/ No	Set to Yes to test print Xs in variable fields.
SelectRecipients	Yes /No Enabled /Disabled/ Hidden	Enabled = Option appears in the Print window and is active (available to be checked). Disabled = Option appears in the Print window but is grayed out (not available to be checked). Hidden = Option does not appear in the Print window.
PrintViewOnly	Yes/ No	If set to Yes, the view only sections will print. This does not apply to entry only sections, which are never printed. Entry only sections are usually worksheets. If the section is marked as hidden and view only, it will not print.

Option	Values	Description
PrePrintedPaper	Yes,Disabled	Determines if the check box which lets you print or not print pre-printed objects appears on the Print window. Also determines the default for this check box—checked or unchecked. You must add this option to the INI file if you want the check box to appear on the Print window. The default for this option includes the check box on the Print window and leaves it unchecked. All objects except fields can be designated as pre-printed on the object's Properties window.
Class	<i>(first three characters of the Module option)</i>	Specifies the printer classification, such as AFP, PCL, XER, PST, or GDI. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the Class option to make sure the correct printer is available before continuing.
SuppressDlg	Yes/ No	Set to Yes to suppress the Windows Print window.
GDIDevice		Specifies the Windows printer name. Click Start, Settings, Control Panel, Printers to see a list of the printers you can choose from. If you set the SuppressDlg option to Yes and leave this option blank, the system suppresses the Print window and automatically prints to the default printer.

Include these options in your FSISYS.INI file. In addition, you can add the following INI setting to automatically select landscape mode when printing any of the specified sections:

```
< VBPrOptions >
  Landscape = (list of landscape sections)
```

Beside the Landscape option, list the sections you want printed landscape. Separate each section with a comma.

Users can override this option at print time.

Note If you do not set the `SuppressDlg` option to Yes, the Windows Print window appears when you use the print device to spool the job. If you omit the `SuppressDlg` option or set it to No, the user can select which Windows print device to spool the output through. By setting this option to Yes, the Windows Print window (not the system's Printer window which normally appears first), will be automatically completed for the user.

If you set the `SuppressDlg` option to *Yes*, the default Windows printer is used unless the `GDIDevice` option specifies a printer. You can use the `GDIDevice` option to name a specific Windows print device for spooling the raw output. The name you specify must match one of the installed printers. You can see these printer names by going to the Control Panel and clicking the Printers icon.

If you misspell the printer name or specify one not installed for the `GDIDevice` option, the system will send the output to the default printer device or you will get an error and printing will stop. On Windows, an incorrect setting sends the raw output to spool to the default printer device.

Do not confuse the *SuppressDlg* option with the *SuppressDialog* option in the Printer control group in the `FSISYS.INI` file. The `SuppressDialog` option suppresses the system's internal Printer Selection window—the one that names which `PrtType:XXX` group from the INI file you wish to use. The `SuppressDlg` option suppresses the operating system's (Windows 32-bit) Printer Selection window.

Using Defaults for the Module and PrintFunc Options

Default values for the Module and PrintFunc options in the `PrtType:xxx` control group are provided when you use a standard print type name or print class, such as AFP, PCL, PDF, PST, VPP, XER, XMP, or GDI.

These defaults keep you from having to enter the Module and PrintFunc names in your INI file. For example, if you want to generate GDI print files, you can specify these INI options:

```
< Printer >
  PrtType   = MYGDI
< PrtType:MYAFP >
  Class     = GDI
```

And the system will default these options for you:

```
< PrtType:MYAFP >
  Module    = GDIPRT
  PrintFunc = GDIPrint
```

AVOIDING PROBLEMS WITH FAX DRIVERS

Use the FullSupport option to prevent problems with FAX drivers which can occur when you are printing from Documaker Desktop.

The GDI driver first looks for this INI option in the control group whose name reflects the Windows print driver, such as *HP LaserJet 4050 Series PS*.

If the FullSupport option is set to Yes, the GDI driver assumes the Windows print driver contains full print support and can handle form sets with mixed simplex and duplex forms (some FAX drivers crash when presented these kinds of forms).

Here is an example:

```
< HP LaserJet 4050 Series PS >  
    FullSupport = Yes
```

If not found there, the GDI driver looks for the FullSupport option in the control group for the printer type, such as *PrtType:GDI*. If you place the FullSupport option in the *PrtType:GDI* control group, it serves as a default for all GDI printers. Putting the option in for specific devices overrides this default.

BATCH PRINTING TO FILES

You can use the GDI print driver to print to a file by adding the `PrintToFile` option in your GDI printer control group. This lets you direct output to the path and file you specify — equivalent to checking the `Print to File` field on the `Print` window.

```
< PtrType:GDI >  
    PrintToFile = Yes
```

Option	Description
PrintToFile	Enter Yes to have the GDI print driver use the Port options as the output print file names for each batch when running the GenPrint program. The default is No.

In the GenPrint program, output print file names for each batch are specified using the `Port INI` option. When you use the GenPrint program with most Documaker print drivers, the `Port` option determines the name of the print stream created for each batch.

Normally, the GDI print driver prints directly to a Windows print driver and does not create files written to disk. By setting the `PrintToFile` option to `Yes` in your GDI printer control group, the GDI print driver creates a print stream for each batch based on the names specified in the `Port` options — just like the other Documaker print drivers.

Because the Documaker GDI print driver is not designed for batch print, these additional GDI print options are recommended when you set the `PrintToFile` to `Yes`:

```
< PtrType:GDI >  
...  
    SuppressDialog= Yes  
    GDIDevice      = (Windows printer name)  
    FullSupport    = Yes
```

Option	Description
SuppressDialog	Enter Yes to suppress the Windows Print window from appearing.
GDIDevice	Enter the name of the Windows print driver you want to use.
FullSupport	Enter Yes to tell the Windows driver to fully support duplexing, tray selection, and so on.

This feature is limited to using the GDI driver with GenPrint (multiple step batch print) to produce output print files and is limited to simple GenPrint (batch print) environments.

Keep in mind that all normal GDI print limitations (fidelity, tray selection, duplexing, and so on) apply, plus the following:

- Banner page processing may not work.
- Cannot use the `SetDeviceName` and `BreakBatch` DAL functions.
- Callback functions may not work.
- Single step processing does not work correctly (all transactions are printed to a single file).
- Multiple driver routers may not work.

- Printing from Studio may work but the Device setting will be used to create the file. Printing from Documaker Desktop may not work.
- Printing to fax drivers, email drivers, and so on may not work and other types of print or print features not previously discussed may not work.

In other words, trying to use PrintToFile option with anything except GenPrint running in a simple batch mode using a normal Windows print driver is not supported.

Chapter 7

Using the Metacode Print Driver

This chapter provides information about Documaker's Metacode Print Driver. This print driver is used by Xerox Metacode printers.

This chapter includes the following topics:

- *Overview* on page 112
- *Setting the Required JSL INI Options* on page 113
- *Using Mobius Metacode Print Streams* on page 132
- *Metacode Printer Resources* on page 133
- *Metacode Limitations* on page 134
- *Metacode Troubleshooting* on page 135

OVERVIEW

The Metacode language is the native mode language for Xerox 4000 and 9000 series printers. This language is superior to printing using line data with Xerox Laser Printing Systems (LPS). The advantages of using Metacode over line data printing include support for portrait and landscape text on the same page, support for different fonts on the same line, precise text positioning, and text justification. In addition, Metacode lets you merge multiple forms onto a single page.

Note All system print drivers support 24-bit color graphics. If your printer does not support color, the print driver will automatically convert the color graphics into monochrome graphics. Keep in mind that for the best performance you should avoid color graphics.

SETTING THE REQUIRED JSL INI OPTIONS

The system does not require you to use a special JSL on your printer to print its Metacode output. The Xerox Metacode printer driver is configurable based on options to produce Metacode which match your existing JSL settings. Here is an example of the PrtType:XER control group which contains these options:

```
< PrtType:XER >
  DJDEIden      = A'@@@DJDE'
  DJDEOffset    = 0
  DJDESkip      = 8
  OutMode       = BARR
  ImageOpt      = No
  CompressMode  = LIN
  JDEName       = META
  JDLCode       = NONE
  JDLData       = 0,255
  JDLHost       = IBMONL
  JDLName       = CBA
  PaperSize     = 0
  Device        = dummy.txt
  RelativeScan  = Yes
```

Several of these options are based on the comparable parameter values in the settings of the printer's JSL. A JSL may contain many JDLs from which to choose, or there may be multiple JSLs compiled into multiple JDLs.

A portion of a JDL may look like the following:

```
CBA:      JDL;
T1:      TABLE      CONSTANT=X'121212121212121212';
T2:      TABLE      CONSTANT=X'13131313131313131313';
T3:      TABLE      CONSTANT=X'FFFF26FFFF';
C1:      CRITERIA     CONSTANT=(0,9,EQ,T1);
C2:      CRITERIA     CONSTANT=(0,10,EQ,T2);
C3:      CRITERIA     CONSTANT=(1,5,EQ,T3);
VOLUME   HOST=IBMONL;
LINE     DATA=(0,255);
IDEN     PRE=A'@@@DJDE',
          OFF=0,
          SKIP=8;
ROFFSET  TEST=C1;
RSTACK   TEST=C2,DELIMITER=YES,PRINT=NONE;
RPAGE    TEST=C3,SIDE=NUFRONT,WHEN=NOW;

/* 8.5 x 11 job */
USA1: JDE;          /* JOB can be used in place of JDE */
OUTPUT          PAPERSIZE=USLETTER;

/* 8.5 x 14 job */
META: JOB;
VOLUME          CODE=NONE

/* Default job */
DFLT: JDE;
VOLUME          CODE=EBCDIC
END;
```

Here are the required options which are based on settings in the printer's JSL file.

JDLName

Represents the name of the JDL to use. The following table shows the relevant JSL statement for the earlier example and the proper option to use based on the JSL example.

JSL statement	CBA: JDL;
INI option	JDLName = CBA

JDEName

Represents the name of the job to use. A JDL may contain many jobs (JDEs) from which to choose. This JDE must contain a *VOLUME CODE=NONE* statement. The following table shows the relevant JSL statements for the earlier example and the proper option to use based on the JSL example.

JSL statements	META: JOB; VOLUME CODE=NONE
INI option	JDEName = META

DJDEIden, DJDEOffset, and DJDESkip

Represent the IDEN statement of the JDL. The value of the DJDEIden setting is a string constant. The types of string constants supported are ASCII (A'string'), EBCDIC (E'string'), Character ('string'), and Hex (X'string'). Octal, H2, and H6 strings *are not* supported.

Strings containing repeat counts, embedded hex values, and upper/lower case toggles *are not* supported. The following table shows the relevant JSL statements for the earlier example and the options to use based on the JSL example.

JSL statements	IDEN PRE=A'@@@DJDE', OFF=0, SKIP=8;
INI options	DJDEIden = A'@@@DJDE' DJDEOffset = 0 DJDESkip = 8

JDLCode

Represents the type of input format expected by the Xerox printer. Character translation occurs as necessary. Currently, the supported code types are *EBCDIC*, *ASCII*, *NONE* (same as ASCII), *BCD*, *H2BCD*, *H6BCD*, *IBMBCD*, and *PEBCDIC*. User-defined code translations are not supported.

Referring to the sample JSL, if the printer is normally started with STA DLFT,CBA then the JDLCode option must be set to *CODE = EBCDIC*. The system's option must contain the value of the CODE statement for the printer's normal operation. This table shows the relevant JSL statements for the earlier example and the proper option to use based on the JSL example.

JSL statements	DLFT: JDE; VOLUME CODE=EBCDIC
INI option	JDLCode = EBCDIC

JDLData

Represents the starting position and length of the print line data within an input data record. The LINE statement contains a DATA entry that holds these values. This table shows the relevant JSL statement for the earlier example and the proper option to use based on the JSL example.

JSL statement	LINE DATA=(0,255);
INI option	JDLData = 0,255

JDLHost

Represents whether the printer is normally in an on-line or off-line state. Currently, the only values we accept for this option are IBMONL (on-line) and IBMOS (off-line). The following table shows the relevant JSL statement for the earlier example and the proper option to use based on the JSL example.

JSL statement	VOLUME HOST=IBMONL;
INI option	JDLHost = IBMONL

ADDITIONAL REQUIRED INI OPTIONS

Below are the additional required options not based on the printer's JSL file.

OutMode

The OutMode option indicates the output format for the Metacode data stream generated by Documaker applications.

Use *BARR*, if the Metacode output is to be transmitted to the Xerox printer via BARR SPOOL hardware and software. When using the BARR setting, a length byte is placed at the start and end of each Metacode record.

Use *BARRWORD*, if the Metacode output is to be transmitted to the Xerox printer via BARR SPOOL hardware and software. BARRWORD should be used *only* if the Xerox printer can handle records longer than the 255 characters.

Use *PCO*, if the output is transmitted to the Xerox printer via PCO hardware and software (from Prism). When using the PCO setting, a 4-byte length field is placed at the start of each Metacode record.

Note The PCO interface has not been tested, but should work.

Use *JES2*, if the Environment option is set to MVS.

Use *MRG4*, if you will transmit the Metacode output to the mainframe using Commcommander or if you will archive it in Docusave (see *Creating Print Streams for Docusave* on page 301 for more information).

Use *LAN4235*, if the output is generated for a Xerox 4235 printer attached to a network.

Here is an example:

```
OutMode = BARR
```

ImageOpt

The ImageOpt option specifies if the graphics are being saved on Xerox printer as *IMG* files or as *FNT* files.

To use IMG files, the printer needs a special GVG or GHO hardware installed. Also, in the JSL you have to specify GRAPHICS = YES.

If you are using IMG files, vectors, in-line bitmaps or want to print charts, set the ImageOpt option to *Yes*; otherwise set it to *No*. Here is an example:

```
ImageOpt = No
```

If the system detects a problem when you are printing in-line bitmaps and vectors, it will display a message that tells you the type of graphic and image name. If the graphic is an in-line bitmap, it includes the name.

Note Metacode printers have a limit of 16 IMG files on a page.

Bypassing the Printing of In-Line Graphics

Use the ImageOptNotSet INI option to specify what action you want the GenPrint program to take when it tries to generate Metacode for a graphic, but determines the ImageOpt option is not set to Yes. By setting this option to None, you can tell the GenPrint program to bypass the printing of graphics and to not emit an error message.

```
< PrtType:XER >  
ImageOptNotSet = None
```

Option	Description
ImageOptNotSet	Use this option to tell the GenPrint program what kind of message it should emit when it encounters a graphic and the ImageOpt option is not set to Yes. You have these options: None - Tells the system to bypass printing the graphic. No message is generated. Warning - Tells the system to bypass printing the graphic and emit a warning message stating that the graphic could not be printed. This warning is written to the LOGFILE. Error - Tells the system to bypass printing the graphic and emit an error message stating that the graphic could not be printed. This message is written to the ERRFILE. The default is Error.

CompressMode

The CompressMode option compresses bitmaps output as inline graphics, such as charts and graphics with the inline graphics flag set. There are four compression modes available, which you can specify using the CompressMode option in the PrtType:XER control group:

- CompressMode = UNC
- CompressMode = ENC
- CompressMode = HTN
- CompressMode = LIN

UNC is the uncompressed or raw bitmap mode. If none is specified, the system defaults to *HTN* mode.

To demonstrate the effects of Metacode graphics compression, the following chart shows the GenPrint program run times and file sizes with the different compression options for a test environment containing in-line images.

Test	GenPrint time	File size
No charts (ImageOpt=No)	182 seconds (3:02)	697,599
UNC – uncompressed	309 seconds (5:09)	9,011,058
LIN compression	290 seconds (4:50)	1,589,226
ENC compression	301 seconds (5:01)	2,248,302
HTN compression	296 seconds (4:56)	1,831,050

Which compression method yields the smallest file size or the quickest compression time depends on the graphic bitmaps you are printing. In general, HTN or LIN compression provides the best results. HTN generally does best with graphics which contain more filled-in or shaded areas, while LIN performs better with graphics which contain more line art. Experiment with your sections to determine the best compression method.

The results of compression can be dramatic, as the table shows. The uncompressed print-ready file is over nine megabytes in size, while the compressed file size ranges from 18% to 25% of the uncompressed file. However, keep in mind that while the reduced file sizes save disk space and reduce transmission times, these files must be decompressed by the printer at print time, which is done automatically by the print controller.

CompileInStream

The CompileInStream option determines whether the FAP files have been loaded. If set to *Yes*, the print driver compiles the print stream using FAP files. Make sure the DownloadFAP option in the RunMode control group is set to *Yes*. If set to *No*, pre-compiled MET files are used.

The print driver creates the print stream using pre-compiled Metacode files. Use the FAP2MET utility to create pre-compiled Metacode files. The GenPrint program loads pre-compiled Metacode members from the PMETLIB PDS under z/OS. On other platforms, the PMetLib option specifies the directory which contains the pre-compiled MET files. If you do not set this option, the system uses the setting for the FormLib option in the MasterResource control group.

Note To use FRM files in your Metacode print stream, set the CompileInStream INI option to No in the Xerox printer control group. Using FRM files enhances performance in high volume situations that use a repeated background form on every page.

Device

This is the name of the file or device, such as LPT1, where the Metacode print stream should be written. This option is ignored by the GenPrint program but should not be left blank or omitted. For instance, you could enter *dummy.txt*.

RelativeScan

When set to Yes, the RelativeScan option tells the system to consolidate all records in the print stream. When set to No, this option tells the system to omit Relative Scan records when consolidating records. If you are using GenPrint version 9.0 or higher you will probably want to leave this option at its default setting (Yes) for maximum optimization.

SPECIFYING INSTALLABLE FUNCTIONS

For the Xerox print driver, you must specify the following set of installable functions in the PrtType:XER control group:

```
OutputFunc      = XEROutput
OutMetFunc      = XEROutMet
InitFunc        = XERInit
TermFunc        = XERTerm
Module          = XERW32
PrintFunc       = XERPrint
```

Using Defaults for the Module and PrintFunc Options

Default values for the Module and PrintFunc options in the PrtType:xxx control group are provided when you use a standard print type name or print class, such as AFP, PCL, PDF, PST, VPP, XER, XMP, or GDI.

These defaults keep you from having to enter the Module and PrintFunc names in your INI file. For example, if you want to generate XER print files, you can specify these INI options:

```
< Printer >
  PrtType    = MYXER
< PrtType:MYAFP >
  Class      = XER
```

And the system will default these options for you:

```
< PrtType:MYAFP >
```

```
Module      = XERPRT
PrintFunc   = XERPrint
```

OPTIONAL INI OPTIONS

Setting the End of the Report

Use the JDLRStack option to set the criteria which signals an *end of report* condition to the printer. In the JDL sample listed earlier, the RSTACK statement performed a criteria test named C2. The C2 test checks a specific part of each input line against the string named T2. If the string T2 matches an input data record at position zero (0) for a length of 10 bytes, an *end of report* condition is signaled. Only CONSTANT criteria using an EQ operator are supported.

Setting the JDLRStack option is optional. If your printer is used for both Metacode and text file print jobs, you *must* set this option. Using the JDL sample listed earlier, the option should be:

JSL statements	T2: TABLE CONSTANT=X'13131313131313131313'; C2: CRITERIA CONSTANT=(0,10,EQ,T2); RSTACK TEST=C2,DELIMITER=YES,PRINT=NONE;
INI option	JDLRStack = 0,10,EQ,X'13131313131313131313'

Starting New Pages

Use the JDLRPage option to set the criteria which signals a *jump to front side of a new sheet* to the printer. In the JDL sample listed earlier, the RPAGE statement performed a criteria test named C3. The C3 test checks a specific part of each input line against the string named T3. If the string T3 matches an input data record at position zero (0) for length of 5 bytes, a *jump to new sheet* condition is signaled because of the *SIDE=NUFRONT* statement. Only CONSTANT criteria using an EQ operator are supported. For the JDLRPage option to work properly, the *SIDE=NUFRONT* and *WHEN=NOW* statements must be used as a part of the RPAGE settings in the JSL file.

Setting the JDLRPage option is optional. If the print job contains duplex pages alternating with simplex (one-sided) pages, this option provides a way to leave blank the backsides of certain pages. Using the JDL sample listed earlier, the option should be:

JSL statements	T3: TABLE CONSTANT=X'FFFF26FFFF'; C3: CRITERIA CONSTANT=(1,5,EQ,T3); RPAGE TEST=C3,SIDE=NUFRONT,WHEN=NOW;
INI option	JDLRPage = 1,5,EQ,X'FFFF26FFFF'

The Metacode print driver automatically places the `SIDE=NUFRONT` statement on all front pages when operating in duplex mode. This lets the system support print stream sorting facilities such as Mobius InfoPak. Also, the `SIDE=NUBACK` statement is now added to blank back pages when in duplex mode.

These statements eliminate the need for the `ADDPAGES` utility which some systems used with Mobius InfoPak support. Without this functionality the first page of an output may print on the back of a previous output.

You will need to add the `SIDE=NUFRONT` statement on all front pages printed, not only those pages that specify a tray change. This is necessary to handle the end of job condition where the last page prints on the front and is moved by InfoPak.

Also, the system will now add a `SIDE=NUBACK` statement for pages that start on the back side of the page, leaving the front side blank.

Note You cannot configure these statements. The system automatically enters them into the print stream. You do not need to add `SIDE=NUFRONT` and `SIDE=NUBACK` statements to your Xerox printer control group (PrtType:XER).

Adding an OFFSET Command

Prior to version 11.3, the first Metacode print stream the system produced would include this statement:

```
DJDE SIDE=NUFRONT,END
```

while the remaining print streams the system produced would include this statement:

```
DJDE SIDE= (NUFRONT,OFFSET) ,END
```

This means the first Metacode print stream will not have a statement which includes the `OFFSET` command.

If your printer requires the `OFFSET` command to be in all statements, including the first `DJDE` statement, add the `DJDEForceOffsetEnd` option to your INI file, as shown here:

```
< PrtType:XER >
  CodeDef          = dcascii9
  Device           = X.MET
  DJDEIden         = E' $$XEROX '
  DJDEOffset       = 0
  DJDESkip         = 8
  DJDEForceOffsetEnd = Yes
```

Option	Description
DJDEForceOffsetEnd	Enter Yes to make sure there is an <code>OFFSET</code> command in every <code>DEJDE</code> statement, including the <code>DJDE</code> statement for the first print stream. The default is No, which omits the <code>OFFSET</code> command from the <code>DJDE</code> statement in the first Metacode print stream. Only set this option to Yes if you must include the <code>OFFSET</code> command for your printer. Most printers do not require <code>OFFSET</code> in the first <code>DJDE</code> statement.

Jogging Pages

Use the `JDLROffset` option to set the criterion that tells the printer to initiate a page offset in the output bin. This option has not been fully implemented.

In the JDL sample, the ROFFSET statement performed a criteria test named *C1*. The C1 test checks a specific part of each input line against the string named *T1*. If the string T1 matches an input data record at position zero (0) for length of 9 bytes, a page offset is initiated. Only CONSTANT criteria using an EQ operator are supported.

Setting the JDLROffset option is optional. Using the JDL sample listed earlier, the option should be:

JSL statements	T1: TABLE CONSTANT=X'1212121212121212'; C1: CRITERIA CONSTANT=(0,9,EQ,T1); ROFFSET TEST=C1;
INI option	JDLROffset = 0,9,EQ,X'1212121212121212'

You can also jog form sets by transaction instead of by batch. In some situations, this can make manual assembly easier. To do this, set the OffsetLevel option to *Formset*, as shown here:

```
< PrtType:XER >
  OffsetLevel = Formset
```

This adds an additional 'OFFSET' parameter to the SIDE=NUFRONT command, which tells the printer to jog after each transaction.

Specifying Spot Color

Use the PrinterInk option to specify the color of ink loaded on a Xerox highlight color printer. You can set this option to one of the following colors:

Blue	Red	Green	Ruby	Violet	Brown
Gray	Cardinal	Royal	Cyan	Magenta	

Blue is the default if you omit this option. This option is used with the SendColor option. If you set the SendColor option to *Yes*, be sure to also set the PrinterInk option. Here is how you would specify cyan as the color of the ink stored on the printer:

```
PrinterInk = cyan
```

Chart Performance and Print Quality

By default, charts are rendered at 150 dpi (dots per inch) in a Metacode print stream. This setting typically provides for a smaller print stream and optimal performance from the GenPrint program.

Charts are scaled by the printer to their proper size and are printed as 300 dpi bitmaps. Because fewer dots are used at these lower dpi settings, you may notice some loss of detail in the printed output and effects such as:

- The circle which makes up the pie chart is less precise
- The lines used in a chart are thicker

Test charts printed to see if the loss of detail is acceptable. In general, horizontal and vertical lines scale with little or no loss of precision. Arcs and diagonal lines may lose some detail.

To disable rendering charts at 150 dpi, add the following option to the Xerox printer control group, usually named *PrtType:XER*:

```
ChartResolution = 300
```

The only other acceptable value for this option is 150. This option does not affect graphics printed as inline graphics.

Optimizing Metacode Print Streams

The GenPrint program lets you produce optimized Metacode print streams. You may want to consider using optimization if your Metacode output causes the printer to cycle down (wait) while printing.

This condition can occur when Metacode records cannot be transferred fast enough to the printer. Optimization helps remedy this situation by combining Metacode print records into larger and fewer records. Reducing the number of records that must be transmitted reduces the amount of time needed to spool the Metacode print stream to the printer. The cost is decreased GenPrint performance. You can also use the METOPT utility to optimize normal (non-optimized) Metacode output. For more information on this utility, see the Utilities Reference.

To have the GenPrint program produce optimized Metacode output streams, add this FSISYS.INI option to have the GenPrint program sort and consolidate records to create more efficient print streams:

```
< PrtType:XER >  
  Optimize = Yes
```

The Optimize option defaults to No, which tells the GenPrint program to run without sorting and consolidating records.

You can enable some extra error checking during optimization. If optimization encounters critical errors, such as the inability to find or open a file, it will notify you and stop immediately. It can report actual or potential non-critical problems it encounters while it runs. For instance, if optimization finds Metacode records that may prevent the file from printing, it can warn you.

To have optimization notify you if it spots potential problems, add the following option to your PrtType:XER control group:

```
< PrtType:XER >  
  ValidLevel = 0 (default)
```

Enter zero (0) to tell the utility not to report non-critical problems. Enter one (1) to tell the utility to report warnings for non-critical problems, but continue optimizing. Enter two (2) to tell the utility to report warnings for non-critical problems and attempt to fix the problems. Enter three (3) to tell the utility to report warnings for non-critical problems and exit immediately.

Regardless of the option you choose, if you receive any warnings, be sure to closely check both the original and, if applicable, the optimized file.

Using a Common Font List

The METOPT utility and the Metacode print driver let you use common font lists at the beginning of a Metacode print stream. A common font list names all of the Xerox fonts that will be used by the print job.

By knowing all of the fonts up front, the Metacode driver can issue a single DJDE FONTS command once at the beginning of the job and avoid issuing DJDE FONTS commands on subsequent pages. This helps some Metacode printers print jobs at their highest rated speed.

In the CommonFonts control group, you will see a list of options similar to these:

```
< CommonFonts >
  Names = 28
  Name1 = FORMSX
  Name2 = FXUNBD
  Name3 = FXUNN6
  Name4 = FXCON6
  Name5 = FXUNN8
  Name6 = FXUNN0
  Name7 = FXUNBH
  . . .
  Name28 = FXUNIO
```

The first option, Names, defines the number of font name entries that follow. The following options specify the Xerox fonts which will be used in the print job.

Note The format used for the CommonFonts control group is the same as that used by Documerge. Therefore, if you used this in Documerge, you can copy that INI control group into your Documaker INI file.

To use common font lists, you must use the METOPT utility or use the Metacode print driver and have the following INI options in the Xerox print group:

```
< PrtType:XER >
  Optimize = Yes
  MaxFonts =
```

Option	Description
Optimize	To use common font lists, set this option to Yes.
MaxFonts	Set this option to the maximum number of fonts your printer can handle in a single DJDE command. This number will vary based upon the printer's memory and configuration. The maximum value is 99 and the default is 20.

If the number of fonts in your common font list exceeds the MaxFonts value, the system outputs the MaxFonts number of fonts in the DJDE FONTS command. The DJDE FONTS command will contain the names of the fonts used on that page plus additional fonts from the common fonts list until the MaxFonts number of fonts is reached.

If the system encounters a page that uses a font not specified in the common fonts list (or the prior DJDE FONTS command to be more precise), it issues a new DJDE FONTS command which appends to the common font list the new fonts for that page.

Setting a Default Paper Size

Use the PaperSize option to set a default paper size when converting Metacode print streams using the Internet Document Server or the MRG2FAP utility.

```
< PrtType:XER >
  PaperSize = 0
```

Enter	Description
zero (0)	for letter size (default)
1	for legal size
2	for A4 size
3	for executive size
98	for a custom size

Automatically Sizing Sections

You can have the system automatically size FAP files converted from Metacode files, (usually Documerge EDL members). This lets you create the FAP files as custom sized sections that are the minimum size required to contain all of the converted objects from the Metacode file.

To have the system automatically size the FAP files, include this INI option in the Xerox printer group you are using to convert the Metacode file:

```
< PrtType:XER >  
    AutoSize = Yes
```

If you omit this option, the system creates full page size sections.

Keep in mind...

- The system will not automatically size the section if the converted Metacode file results in a multipage section.
- If the section is automatically sized and the result is a custom sized section, the Metacode loader does not try to determine if the section is landscape and does not rotate landscape objects.

Inline Graphic Performance and Print Quality

Graphics at 75, 100, or 150 dpi, printed using inline graphics, are scaled by the printer to their proper size and printed as 300 dpi bitmaps. Because fewer dots are used at these lower dpi settings, you may notice some loss of detail in the printed output and effects such as:

- Arcs and circles are less precise
- The lines used in a graphic are thicker

Test LOG files printed as inline graphics to see if the loss of detail is acceptable. In general, horizontal and vertical lines scale with little or no loss of precision. Arcs and diagonal lines may lose some detail.

To avoid scaling inline graphic LOG files, use Documaker Studio or Logo Manager to scale your graphics to 300 dpi. Most graphics are normally 300 dpi and most graphics are not printed as inline graphics.

Adding Color to Charts

Use the ColorCharts option to print the graphic portion of the chart in color.

```
ColorCharts = Yes
```

This option is used with the SendColor and PrinterInk options.

Using Named Paper Trays

By default, Metacode output specifies the main tray for pages that use Tray 1. The AUX tray is specified for all other trays. If you have named trays in your JSL, specify these named trays in your options. An example of this option is shown here:

```
Tray1 = ONE1  
Tray2 = TWO2  
Tray3 = THREE3  
Tray4 = FOUR4
```

Specifying the Printer Model

Use the PrinterModel option to specify the particular printer model you are using. There may be subtle differences between printer models that can affect the output sent to the printer. Currently, only the 3700 printer requires this setting. An example of this option is shown here:

```
PrinterModel = 3700
```

Specifying the Resolution

Use the Resolution option to specify the printer's dots per inch resolution. Currently, only 300 dpi is supported, which is also the default.

```
Resolution = 300
```

Displaying Console Messages

Use the OTextString option to display a message on the printer console. The text you specify is sent before the print job starts. For example, this lets you display the message, *Put BLUE paper in tray 1* before a print job starts. Here is an example:

```
OTextString = "Put BLUE paper in tray 1"
```

The system also supports multiple OTEXT messages in the Metacode print driver at a print batch level. Additionally, the system lets OTEXT messaging generate multiple messages per print batch. To turn on multiple OTEXT messaging, add this option to the FSISYS.INI file

```
< PrtType:XER >  
MultipleOText = Yes
```

The default is No.

This tells the system to ignore the OTextString value in the PrtType control group and instead use the ones found in the appropriate print batch group.

For example, if you have three print batches, called BATCH1, BATCH2, and BATCH3, under each separate batch group, put required number of sequential messages for that batch:

```
< BATCH1 >
```

```
...
OTextString1 = "Batch 1 OText String1"
OTextString2 = "Batch 1 OText String2"
OTextString3 = "Batch 1 OText String3"
< BATCH2 >
...
OTextString1 = "Batch 2 OText String1"
OTextString2 = "Batch 2 OText String2"
OTextString3 = "Batch 2 OText String3"
< BATCH3 >
...
OTextString1 = "Batch 3 OText String1"
OTextString2 = "Batch 3 OText String2"
OTextString3 = "Batch 3 OText String3"
***
```

Keep in mind that the index tags OTextStringX (where X is a number) must start with one (1) and be sequential. The system stops writing OTEXT records to the batch when it finds a tag that is out of sequence. Here is an example:

```
OTextString1 = "Batch 3 Otext String 1"
OTextString3 = "Batch 3 Otext String 3"
```

In this example, only the first one would display on the screen, because OTextString2 is not encountered next.

Stapling Forms

Some Metacode printers include a stapling feature. The system supports this feature, but it has not been tested and is not warranted.

Using this feature, forms printed on certain Metacode printers can be stapled if you specify a StapleJDEName option in the PrtType control group. This causes a new JDE to be specified on forms that need to be stapled.

It is assumed that the Staple JDE option has the same settings as the normal JDE specified except for the additional STAPLE command. You specify which forms should be stapled using Documaker Studio.

This option only affects implementations which print to Metacode printers with the optional stapling feature. An example of this option is shown here:

```
StapleJDEName = JDESTP
```

Duplex Switching

In earlier versions of the system, a Metacode print stream began and continued as a simplex job until the system encountered a page that needed to be duplex. At that point, the duplexing option was turned on. From that point forward, the print stream remained in duplex mode. For performance reasons, the system did not switch out of duplex mode. Research showed that for most cases, this was the most efficient way to drive the printer.

If, however, you are directing the printer output stream to a *value-added* process, you may want to include the actual duplex selection information with each form set. Without the commands to specify the duplex state, some value-added processes may not work properly. By setting the DJDELevel option to *Formset*, each form set will include a duplex command which specifies either simplex or duplex mode (DJDE DUPLEX=YES or NO always appears at the beginning of every new form set). A value other than *Formset* causes the duplex commands to be output as before. Here is an example:

```
DJDELevel = Formset
```

Using VSAM to Store Resources

The system lets you store DDT files, precompiled Metacode resources, NA and POL files, and transaction trigger files in VSAM KSDS (Virtual Storage Access Method/Key Sequence Data Set) data sets. If you use this feature, you must set the following options in the VSAM control group in the FSISYS.INI file:

```
< VSAM >
DDTVSAM = DD:DDTVSAM    DDT files
METVSAM = DD:PMETVSAM   PreCompiled Metacode files
VSAMRCPTB = DD:SETRCPVS Transaction Trigger file
VSAMNA = DD:NAFILE      NA and POL files
```

For more information on implementing VSAM support under z/OS, see [Optimizing Performance in the Documaker Installation Guide](#).

PrintViewOnly

If set to Yes, this option tells the system to print the view only sections. The default is No. This does not apply to entry only sections, which are never printed. Entry only sections are usually worksheets. If the section is marked as hidden and view only, it will not print.

Caching Files to Improve Performance

The following options let you minimize the opening and closing of frequently used PDS members by retaining, or caching, file handles and file data. In many cases the default values are sufficient, but for specific cases in which you use many different sections, you may need to increase these caching values to improve performance.

Here are the options you can customize:

```
< Control >
CacheFAPFiles =
RuleFilePool =
LogCaching =
CacheMethod =
```

Option	Description
CacheFAPFiles	Specifies the number of FAP files to keep available for re-use without re-loading them from disk. The default is 100.
RuleFilePool	Specifies the number of DDT files to keep available for re-use without re-loading them from disk. The default is 100.

Option	Description
LogCaching	Enter No if you do not want the system to log caching statistics. The default is Yes.
CacheMethod	Use to set the type of caching method. You can choose from LFU (least frequently used), LRU (least recently used), or LFUO (least frequently used optimized). LFUO is the default.

MET files contain pre-compiled Metacode information produced by the FAP2MET utility. The GenPrint program loads MET members from the PMETLIB PDS under z/OS. On other platforms, the PMetLib option specifies the directory containing the pre-compiled MET files.

If not set, the system uses the setting for the FormLib option in the MasterResource control group. The CacheFiles option keeps frequently used MET members available for re-use. This option is placed in the PrtType:XER control group in the FSISYS.INI file, as shown here:

```
< PrtType:XER >
  CacheFiles = 100 (default is 100)
  InitFunc   = XERInit
  TermFunc   = XERTerm
```

Caching statistics for FAP files, DDT files and Xerox resources such as pre-compiled Metacode files (PMETs) and forms (FRMs) are collected and can be placed in the LOGFILE.DAT file. These statistics show the following information:

Item	Description
Method	The caching method you are using (LFUO, LFU, or LRU).
Size	The size of the caches. The default is 100.
Hits	The number of times the system tried to load a resource from the cache and found it there.
Misses	The number of times the system tried to load a resource from the cache and did not find it there.
Total	The combined hits and misses. This represents the number of times the system tried to load a resource from the cache.
Purges	The number of times the system had to remove a resource from the cache to put another resource into the cache. The system decides which resource to remove based on the method. If you are using LFUO or LFU, the least frequently used resource is removed. If you are using LRU, the least recently used resource is removed.

Using the Loader

The system lets you load print-ready Metacode files. For this feature to work, the print-ready Metacode file must have the same extension as the Ext option in the Loader:MET control group in the FAPCOMP.INI file. Here is an example:

```
< Loader:MET >
  Desc      = Xerox Metacode (*.MET)
  Ext       = .MET
  LoadFunc  = XERLoadMet
  Module    = XERW32
< Loaders >
  Loader    = MET
```


Along with the Metacode loader feature, another INI option is required in the PrtType:XER control group. The DefaultFont option defines the default font to use to indicate the names of any graphics in the print-ready Metacode file.

The graphics do not appear in Studio when the print-ready Metacode file is opened. Instead the name of the graphic appears, in the default font, and the space taken by the graphic is indicated. In addition, the default font is also used for displaying any text that references a font not present in the font cross-reference file.

To set the default font, enter the name of a Xerox font file contained in the font cross-reference file as shown here:

```
< PrtType:XER >
  DefaultFont = FXTIN8
```

If there are any graphics in the MET file, the system requires a LOGO.DAT file in the FormLib directory so it can display graphics properly for all rotations. The LOGO.DAT file, which is a semicolon-delimited file, should look similar to this:

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

Here are a few points to keep in mind when using this feature:

- The PrtType settings must match the setting used to produce the print-ready Metacode file.
- Rotated text will not display properly.
- Blank pages are created for simplex forms printed in duplex mode.
- This feature slows the printing of large print-ready files (more than 100 pages).
- If there is a reference to a FRM file in the MET file, the system cannot display the MET file.
- The system cannot display charts and graphics.

Using the Class Option

You can use the following INI option to specify the printer classification, such as AFP, PCL, XER, PST, or GDI. If you omit this option, the system defaults to the first three letters from the Module option.

```
< PrtType:XER >
  Class = XER
```

Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the Class option to make sure the correct printer is available before continuing.

Adding User-Defined DJDE Statements

You can place the AdditionalDJDE option anywhere in the PrtType:XER control group. Each AdditionalDJDE value represents a distinct and separate DJDE statement, given verbatim. You can include as many AdditionalDJDE statements as needed. All of the located AdditionalDJDE statements are inserted into the print stream. You can also specify the batch in which to output the DJDE statement. Here is an example:

```
< PrtType:XER >
  AdditionalDJDE= "BATCH1";FEED=COVER,;
  InitFunc       = XERInit
  ...
  AdditionalDJDE= "BATCH1";STOCKS=BLUE,;
  ...
  AdditionalDJDE= JDL=DPLJDL,JDE=STRTON,;
```

The first two occurrences only apply to the BATCH1 batch. The third occurrence has no batch specified, so this DJDE statement is written to all print batches.

Keep in mind that these user-defined DJDE statements are placed after the BEGIN DJDE record and before the other DJDEs that are always inserted, such as FONTS. Make sure the DJDE syntax is correct and that the new DJDE records do not interfere with the ones automatically inserted into the print stream by the system.

Also, it is very important that you follow the correct syntax when coding the INI line. If you enter an invalid batch name, no corresponding batch will be found and the DJDE line will be ignored or not output in any batch. And, if the DJDE syntax is incorrect, the printer will issue error messages or unpredictable print results may occur.

Using Third-Party Software to Read Metacode Files

If you use third-party software to read Documaker-produced Metacode files and that software needs the DJDE, RSTACK, and RPAGE commands to begin with a carriage control value other than the default value of *0x01*, you can use the DJDECarrControl option to handle this. You simply enter a value in the form of a string constant. These string constants are supported:

- ASCII (A'string')
- EBCDIC (E'string')
- character ('string')
- hex (X'string')

Note The character string produces an EBCDIC string, same as E'string'.

The default value is 1 (X'01'). Here is an example:

```
< PrtType:XER >
  DJDECarrControl = X'09'
```

Keep in mind that any carriage control value will be accepted and no attempt is made to make sure a valid carriage control is used.

Specifying the Paper Stock

Using Documaker Studio you can specify what paper stock the form should print on. This will help users who have more than nine types of paper stocks. Here is an example of the INI options you could set up:

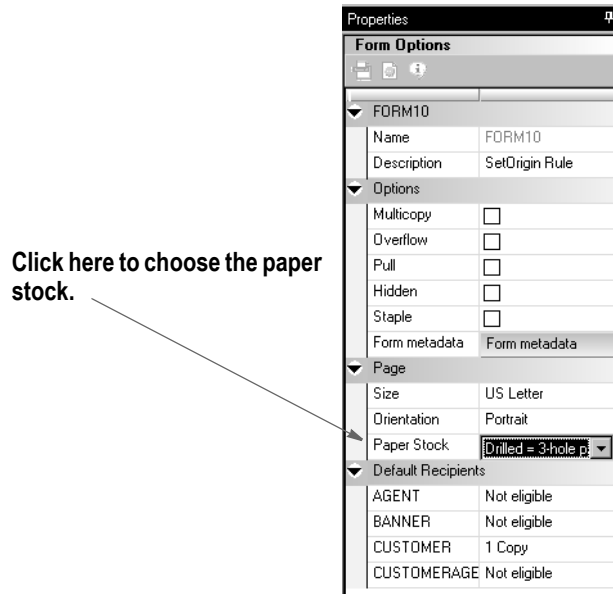
```
< PaperStockIDs >
  PaperStockID = Drilled
  PaperStockID = 201b
  ... (and so on)
< PaperStockID:Drilled >
```

```

Description = 3-hole paper
< PaperStockID:201bW >
Description = 201b White Paper
DJDE = DJDE name

```

Once you have set up the appropriate PaperStockID options, you will see those options available via Studio's Form manager. Just open a form and select the appropriate paper stock in the Paper Stock field on the Properties tab, as shown here:



Your selection is reflected in the POL file produced by the GenData program. In this example, the form called DEC PAGE has a paper stock ID of *Drilled*.

```

; SAMPCO; LB1; DEC
PAGE; | | FORMPAPERSTOCK=Drilled| | ; R; ; QPRUNA| DL (3360, 18600) <AGENT, COMPANY, INSURED>

```

In the Metacode printer control group, you must set the TrayUsePaperStockID option to Yes, as shown here:

```

< PrtType:XER >
TrayUsePaperStockID = Yes

```

If the TrayUsePaperStockID option is set to Yes, the Metacode print driver takes the form's PaperStockID and tries to find the DJDE INI option for it in the INI file when it emits the tray command.

Keep in mind...

- The paper stock selection applies to the entire form
- Only the Metacode print driver uses the paper stock selection
- Only Documaker Studio lets you select the paper stock

USING MOBIUS METACODE PRINT STREAMS

You can use Mobius to archive Metacode print streams and also use Docupresentment (IDS) to retrieve archived Metacode print streams and produce or present PDF files.

You can retrieve the archived Metacode print streams using Mobius' ViewDirect APIs. The ViewDirect APIs are built to communicate with the Mainframe Mobius Archive via TCP/IP. If you license the Mobius' ViewDirect APIs, you can write a custom rule to retrieve your archived Metacode print streams.

To do this, include these options in your FSISYS.INI file (for Studio and the MRG2FAP utility):

```
< PrtType:XER >
  OutMode= MOBIUS
< Loader:MOBIUS >
  Desc   = Mobius Metacode files (*.MET)
  Func   = XERLoadMobius
  Module = XEROS2
< Loaders >
  Loader = MOBIUS
< Control >
  Mobius = XER
```

Where *XER* is the printer control group that contains the Mobius Metacode information.

To use the Mobius Metacode loader in Docupresentment, use the same MTCLoadFormset rule you would use to load a Documerge Metacode print stream.

To specify a Mobius Metacode print stream, instead of a Documerge print stream, the Xerox printer control group must include this INI option:

```
< PrtType:XER >
  OutMode = MOBIUS
```

Metacode print streams retrieved from a Mobius archive have a special record blocking scheme and use special comment records to indicate the fonts used. This version adds support for reading Metacode print streams retrieved from a Mobius archives.

Use XERLoadDocuMerge as the loader function. It checks for an OutMode setting of MRG2, MRG4, or ELIXIR. You must add MOBIUS to the list of allowed OutMode settings and you must add your Mobius comment checking to XERLoadMet, when the OutMode option is set to *MOBIUS*.

Note The loader functions convert a particular type of file, such as a PCL print stream, a Metacode print stream, an RTF file, and so on, into an internally formatted file. Once converted, the system can then do a variety of things with that file, like display it in Studio, print it on a supported printer, or save it as another type of file, such as a FAP file, RTF file, or a print stream file.

The loader included in this version can also be used in other Documaker products. For instance, Studio can use it to load Mobius Metacode, then display, modify, and save the result as a FAP file or print to a supported printer. It can also be used by the METDUMP utility to dump information about the Mobius Metacode print stream.

METACODE PRINTER RESOURCES

A number of resources are used in the printing process. These resources generally reside on the printer's disk drive.

Fonts

Xerox fonts are ASCII fonts. Xerox fonts are not scalable and do not rotate. There is one font file for each rotation and different files are required for different sizes. The file extension is *FNT* and file names are up to six characters long. Xerox fonts in all four orientations (portrait, landscape, inverse portrait, and inverse landscape) are included in the system fonts provided with Documaker.

Forms

Xerox forms are precompiled electronic files containing static text, boxes, graphics, and so on, ready to be merged with variable data. Forms always have the extension *FRM*. Like fonts, the maximum file name is six characters. You use the FAP2FRM utility to create Xerox forms from FAP files.

Images

Xerox images are large bitmaps or raster patterns that are stored in a special file format. These images are merged onto the forms which are then merged with the variable data. The file extension is *IMG* and the maximum file name is six characters.

Note	You must install a GVG hardware card on the printer to print IMG files. You can use the LOG2IMG utility to create Xerox images from LOG files. For more information on this utility, see the Utilities Reference.
-------------	---

Logos

Logos are small bitmaps stored in a different format than IMG files. The extension is *LGO* and the file name is six characters long. You can only use Xerox logos inside a FRM file. You cannot invoke them directly in the data stream.

Note	These LGO files are quite different than the graphics (.LOG) files used in Documaker Studio and Logo Manager. Documaker software does not use Xerox LGO files.
-------------	--

METACODE LIMITATIONS

Xerox Images

The maximum number of images and inline graphics per page is 16.

HMI Support

HMI (horizontal motion index) is supported for zero (0) and 270 degree rotated text on portrait forms only. HMI combines separate text labels which are positioned on the same line and which use the same font into a single Metacode record. FAP files with justified paragraphs can benefit from this feature. Use the FAP2MET utility to implement HMI into pre-compiled MET files.

Changing the Paper Size on the 4235 Printer

You can not easily change paper sizes in one job. Each job is controlled by a JDE. If you need to pull paper from bins of different sizes, you have to call a different JDE each time you change from one paper size to another. This is similar to staple support. There is no code to invoke different JDEs for change of paper size.

Xerox Forms

If a Xerox form (FRM file) contains more than 48 blocks (each block is 512 bytes), your printer may not have enough memory to print it.

The CD (Character Dispatcher) memory is divided into three regions. The first region loads all fonts used on a page. The second region is used for TL/DLs which contain inline Metacode (may only be variable data if you use an FRM). The third region loads the TL/DLs from an FRM file, if one is being used for the page.

If you have version 2 of the printer software, your printer supports eight TL/DL buffers of 3K each (same as 48 blocks of 512 bytes each) for inline Metacode. With version 3.5 of the printer software, the limit was increased to 16 buffers of 3K each.

Note Our testing shows that with version 3.5, TL/DLs from FRMs (the third region of CD memory) are still limited to 8 TL/DL buffers of 3K each (same as 48 blocks of 512 bytes each).

Typically, Xerox 9700 and 9790 printers still have the older release installed. If so, you may want to upgrade to version 3.5. The Xerox 4000 series printers (4050, 4850, and so on) always come with version 3.5 or higher.

When you are not using FRM files in a print stream, the system does not use the CD memory reserved for FRM files.

METACODE TROUBLESHOOTING

Unexpected Color Output

Even though you set the SendColor option set to *No*, you still get color output when printing. This occurs when:

- You specified *Print in color* for some elements of the FAP file
- You precompiled the FAP files with the */C* option on FAP2MET
- A *SUB INK* command was issued on the printer

If ink substitution occurred because of an operator command, such as SUB INK BLUE (or RED or GREEN), the colored components of the precompiled MET file will be brought in with color attribute turned on and printed with color. This happens regardless of how you set the SendColor option. To print in black and white, either re-run the FAP2MET utility with no */C* flag, or use the *END* command to cancel ink substitution on the printer.

Unexpected Black and White Output

Even though you set the SendColor option to *Yes*, you still get black and white output when printing. Use this checklist to make sure you have done everything to print in color:

- Make sure you specified *Print in color* for the color elements in your FAP file, such as text, shaded areas, lines, and so on.
- Make sure you precompiled the FAP files with the */C* option on FAP2MET;
- If you are using precompiled FAP files, make sure you compiled those FAP files using the FAP2CFA utility.
- Make sure you run the GenPrint program with the SendColor option set to *Yes*.

Highlight Color Should Match the PrinterInk Option

The PrinterInk option causes a DJDE ILIST command which specifies the highlight color to use. If a different highlight color is installed on the printer, the printer follows the procedure specified in the ABNORMAL statement in the JDL and JDE loaded. The ABNORMAL procedure specifies whether the job should continue, abort, or stop. If no ABNORMAL procedure is declared, the default is for the printer to stop so a new ink cartridge can be loaded. Besides the ABNORMAL statement, the printer operator can override the ink setting using the SUB command (for example, SUB INK BLUE or SUB INK CURRENT).

LOG File Orientation

To print a portrait section which contains a graphic on a landscape form using pre-compiled MET files, set the LoadFAPBitmap option to *Yes*. This is necessary because the graphic name must change from the portrait (zero degrees) name to the landscape (270 degrees) name.

Output Catching Up with the Input

If your printer cycles down and displays a message stating that the output caught up with the input, it indicates the average number of records per physical page is greater than the maximum number of records that can be transferred across the channel in the time allowed for a page.

This situation causes the printer to cycle down so it can buffer more pages before it continues. This table shows the maximum average number of records that can be transferred across the channel in time to support the printer running at rated speed:

Printer	Maximum Records Per Page
4050	285
4090	155
DP96	149
41/4635	105
DP180	78

To resolve this problem, you need to optimize the Metacode print stream. For more information, see *Optimizing Metacode Print Streams* on page 122.

Printing Rotated Variables

Here is a list of field options you can specify in the NAFILE.DAT file:

Option	Description
E	Error
M	Manual
P	Protected
G	Global scope (entire form set)
F	Form scope
H	Hidden field (such as a dummy field, neither displayed nor printed)
N	Nonprintable field (displayed, not printed)
C	Send-copy-to field (receives current recipient name at print time)
9	Rotated 90 degrees
8	Rotated 180 degrees
7	Rotated 270 degrees

Note For legacy MRLs, some of these options require the FAP field attributes to be available at run time because the DDT file does not include the necessary information. You can use the `CheckImageLoaded` rule to make sure this information is available.

There are no DDT files in MRLs created using Documaker Studio and Studio automatically handles including `EjectPages` rules for you. The following instructions only apply if you have legacy MRLs and are still using the legacy Image Editor.

The legacy tools, such as Image Editor, Logo Manager, Form Set Manager, and so on, were omitted from Documaker releases beginning with version 12.0.

Multipage Sections

When you use multipage FAP files and pre-compiled MET files, you must use the `EjectPage` rule. This rule enables the printing of multipage sections. Here are the steps to apply the rule in Image Editor:

1. Open the FAP file in Image Editor.
2. Select Format, Image Properties and then click the Load DDT button.

Image Editor detects that the section contains multiple pages and inserts into the DDT file as many `EjectPage` rules as there are pages.

You must have a variable field on each page. The variable field can be a dummy field that is hidden.

When you implement multipage FAP files and pre-compiled MET files, keep these requirements in mind:

- Only multipage FAP files are applicable.
- Multipage FAP files cannot be mixed with single page FAP files on the same form. The system cannot easily determine the page number in this case.
- The multipage FAP file came from Documaker Studio or Image Editor and therefore there is only one section per page, hence, each page on the form has a section list that contains one and only one section.
- The index of the page on which that section resides within that form is the number of the page.
- Multipage sections can be duplexed by setting the form to either *Front* (long edge binding) or *Short* bind (short edge binding). Internally created sections will be set to *Rolling* for the remaining pages.

Note If a form begins with a rolling duplex option, the print drivers begin printing on the blank back page of the previous form. Any form that starts with rolling and begins a form set is treated as the front page of a rolling set.

Operator Command, FEED, Causes Duplex Problems

If you enter an operator command to specify an input tray—because for instance, one paper tray is empty and while you refill it you want the printing to continue using another tray—you can no longer select trays from DJDEs in the job stream. Instead, you will get messages which tell you tray selection was suspended by an operator override.

All paper feed from that point forward, will be from the tray specified in the operator command. This can cause duplex jobs to print incorrectly if you have completed printing on a front page and the next page should print from a different tray.

To correct this situation, enter a *FEED=MAIN* command. This command tells the printer to switch to tray 1 and enables tray selection through DJDE commands so the next paper selection command is obeyed.

Line Density Errors

As the speed of the printer increases, there is less and less time available to the character dispatcher to form the scan line and send it to the image generator. Here is some information on how this affects the various Xerox printers:

- Since the 4135 printer is the fastest of the Xerox printers using the older CD/IG, the chances of running out of time and causing a *line density error* are greatest with this model.
- The Xerox 4050 and 4850 printers are too slow for this to be a problem. These printers allow more fetches from the font memory per scan line.
- The Xerox 4635 printer's image generation module has been revamped to such an extent that Xerox almost guarantees there will never be a line density exceeded error on a 4635 printer.
- The 4235 printer is slow and works quite differently than the centralized printers.
- If a job works fine on a 9790 printer but fails on a 4135, the number of character fetches is likely on the borderline of failure.

If you experience line density problems, check your FAP files for the following:

- Text superimposed on shaded areas.
- Large number of text lines with small fonts.
- Large number of horizontal lines whose thickness is measured in an odd number of dots. If you change the thickness of a horizontal line from three dots to two or four dots (0.01" to .006666" or 0.013333"—24 FAP units to 16 or 32 FAP units), it reduces the character count from two to one.

The Xerox line drawing font has three horizontal line drawing characters which specify lines with thicknesses of two, four, and eight dots (.006666", .013333" and .026666" or 16, 32, and 64 FAP units). Odd thicknesses require the printer to overlay or overlap multiple lines.

- Large number of small boxes, many of which have common boundaries. On paper it looks like one line. Actually, there may be two or more character fetches for the same black dots. Create these kinds of boxes by drawing lines rather than boxes.

Output Data Length Validation

Metacode printer JSL specifies the length of data that can be received. This data length must match the value output into the Metacode print stream. You specify the data length in the JSL as shown here:

```
LINE DATA = (0,213)
```

You specify the data length in the PrtType:XER control group in the FSISYS.INI file, as shown here:

```
< PrtType:XER >
  JDLData = 0,213
```

In this example, the JSL specifies a maximum data length of 213, so the INI option has a matching value. The maximum length value is also used in the Metacode print driver to make sure no more than the specified amount of data is output in any Metacode record. If the amount of data to be emitted in the record exceeds this amount, an error message such as the following appears:

```
Record Length 214 is too long - maximum length is 213.
```

Note Under z/OS, Metacode output files are VB datasets. The JCL specifies a maximum length of a record (LRECL). If an attempt is made to write a record longer than the LRECL value, the write will fail and an error message appears.

Be advised that under z/OS, with VB datasets, the LRECL size includes a 4-byte record length, known as the *RDW*. The RDW is implicitly added to the front of each variable length record. Therefore, you should set the LRECL value for the Metacode output dataset to a number equal to the JSL maximum length plus four to account for the RDW bytes at the front of the record. For the above example, set the LRECL of the Metacode output file to 217.

USING XEROX FORMS (FRMs)

The system lets you use Xerox form (FRM) resources when you print to Xerox Metacode printers. FRMs are printer resident resources that contain static full-page images. The system can use FRMs during the print process.

You can convert frequently used static full-page images into FRMs using the FAP2FRM utility. To indicate an image is resident on the printer as a FRM file, use Studio's Form Set manager. The Printer Resident field indicates the image is a pre-compiled resource resident on the printer—as opposed to a pre-compiled resource that needs to be downloaded to the printer.

Here are some guidelines for using Xerox forms (FRMs):

- Create one FAP file per page. If there is a text area, do not put variable data within the text area.
- The image size must be one of the standard paper sizes, such as US Letter, Legal, A4, or Executive.
- Because Xerox printers can only accept file names up to six characters in length, the image name can be up to six character long. If, however, it is a multipage FAP file, the name can consist of no more than four characters to accommodate the two-character number added by the FAP2FRM utility. Here are some examples: *TEST01.FRM* for the first page, *TEST02.FRM* for the second, and so on.
- Use the FAP2FRM utility to convert FAP files into FRM files. For multipage FAP files, create multiple FRM files. The names are appended with two-digit numeric suffixes.

- On workstations, store the FRM files in the same directory as the FAP files. On z/OS, keep them in a PDS attached to the PFRMLIB DD name. On AS400 systems, use the FRMFile option in the Data control group to specify to store the FRM files.
- Use the Printer Resident field in Studio's Form Set manager to mark individual forms as *printer resident*. After you do this, the FORM.DAT file contains the *V* image option which indicates the image is resident on the printer. When you run the GenPrint program, a `DJDE FORMS=fname` command is inserted for the corresponding images. The remainder of the images are printed by inline Metacode, possibly using precompiled MET files.
- Install the FRM files on the Xerox printer using the XERDNL utility. Copy the resulting *.DAT files to the printer. To make sure the forms are installed on the printer, use the SAMPLE console command to print the form files.

BARRWRAP

The BARRWRAP utility converts Metacode output from JES2 format into BARR format.

The BARR interface attachment for Metacode printers requires that the Metacode print stream files contain BARR specific information. The BARRWRAP utility adds this information to an existing Metacode print stream file, which lets you print the output file via the BARR interface.

After you run the utility on a Metacode file, `76 1A FF 00` is added at the beginning of the file. This tells BARR the file is a Metacode file. A byte denoting the record length is also added at the beginning and end of each record in the file.

Use this utility when you test the GenPrint program on z/OS. If the z/OS system is not directly channel-attached to the Xerox printer, you must download the print streams to an z/OS system—use no ASCII translation, but do use CRLF. Then, using BARRWRAP, the print stream is packaged to successfully pass through BARR/SPOOL.

Note Occasionally, the binary data contained in a Metacode file has a sequence of hex bytes (`x'0D0A'`) which could be misinterpreted as a carriage return/line feed. This is true particularly for charts and other inline graphics. Convert such data streams using the BARRWRAP utility on the z/OS platform before you download them with the no ASCII and no CRLF (binary) options.

TRANSFERRING FILES FROM XEROX FORMAT FLOPPIES

Resources saved on a 5 1/4-inch floppy, using *FLOPPY SAVE file.ext*, are saved in a special Xerox format. For use in the system, or for transferring to a 4235 printer, you must convert these resources into DOS format.

Chapter 8

Using the MPM Print Driver

This chapter provides information about Documaker's MPM (Multipart MIME) Print Driver.

Use the MPM print driver to produce output in multipart MIME (MPM) format that contains both plain text and generic HTML contents. The MPM print driver enhances the EPT print driver so you can send data in MPM format in the email body.

The generic HTML format produced by the MPM print driver differs from the dynamic HTML produced by the HTML print driver. The generic HTML produced is based on HTML version 4.0 and uses more HTML elements such as TABLE, TR, TD, and so on, to lay out FAP objects in relative positions. FAP objects that can be printed in HTML format include lines, boxes, shaded areas, graphics, charts, bar codes, vectors, texts, and text areas.

This chapter includes the following topics:

- *Setting MPM INI Options* on page 142
- *Example MPM Document* on page 145
- *Designing Forms* on page 146

SETTING MPM INI OPTIONS

Here is an example of the INI options you use to set up the MPM print driver:

```
< PrtType:MPM >
  Module           = MPMW32
  PrintFunc        = MPMPrint
  OutputFunc       = PRTStdOutputFunc
  OutputMod        = PRTW32
  Device           = .\data\x.htm
  TemplateFields   = Yes, Enabled
  SendColor        = Yes, Enabled
  ForcePrintInColor= Yes
  CreatePlainText  = Yes
  PageBorder       = Yes
  CollapsePage     = Yes
  HR               = Size=2 Width=100% Color=Black
  DirLinks         = Yes
  BitmapResolution = 300
  BitmapPath       = \\mpmexample.com\html\
  BitmapHTTP       = http://mpmexample.com/html
  Fonts            = TTF
  MonospaceFonts   = `Letter gothic`, `Courier`
  AddPropFonts     = Yes
  ProportionalFonts= `Arial`, `Times`
  Debug            = No
< MasterResource >
  FontLib          = .\fontlib\
```

Option	Description
PrtType:MPM control group	
Module	(Optional) Enter MPMW32 . This is the name of the program module which contains the print driver
PrintFunc	(Optional) Enter MPMPrint . This is the name of the program function that is the main entry point into the print driver.
OutputFunc	(Optional) Enter the name of the output function. The default is PRTStdOutputFunc.
OutputMod	(Optional) Enter the name of the output module. The default is PRTW32.
Device	(Optional) This option is used by GUI applications such as Documaker Studio or Documaker Desktop. Documaker Server ignores this option. If this option is not set or is set to NULL, the system generates a file name, such as 09876543.HTM. If you include a file extension in the file name for the device, that extension is used for all files produced. Otherwise, the system assigns the file extension to match the type of output you are producing.
TemplateFields	(Optional) Set to Yes to test print Xs in variable fields. You can also include these additional options: Enabled = The Template Fields option appears in the Print window and can be checked or unchecked. Disabled = The Template Fields option appears in the Print window but is grayed out (not available to be checked). Hidden = The Template Fields option does not appear in the Print window. The default is No, Enabled.

Option	Description
SendColor	<p>(Optional) Enter No to print in black and white. Enter Yes to print in color. If you enter Yes, all applicable objects will be printed in color if the object's Color property is properly set.</p> <p>You can also include these additional options:</p> <p>Enabled = Send Color field appears in the Print window and is active (available to be checked).</p> <p>Disabled = Send Color field appears in the Print window but is grayed out (not available to be checked).</p> <p>Hidden = Send Color field does not appear in the Print window</p> <p>For instance, Yes, Enabled indicates color output and displays the Send Color field on the Print window where it can be checked or not.</p>
ForcePrintInColor	(Optional) Enter Yes to enforce printing in color. The default is No.
CreatePlainText	(Optional) Enter Yes to produce HTML and plain text. The default is No, which produces only HTML.
PageBorder	(Optional) Enter Yes to draw a border around the page. The default is No.
CollapsePage	(Optional) Enter Yes to remove white space at the bottom of page. The default is No.
HR (Header Rule)	<p>(Optional) Displays a line between pages. You can configure the size (height in pixels), width, and color. For instance, here is an example of how you can set up this option:</p> <p>HR = Size=2 Width=100% Color=Black</p> <p>This tells the system to make the line two pixels in height, make it the full width of the page, and color it black.</p>
DirLinks	(Optional) Enter Yes to include Next and Prev links on the HTML pages. The default is No.
BitmapResolution	<p>(Optional) This option only affects charts and bar codes. The MPM print driver converts these objects into bitmaps.</p> <p>Specify the resolution in pixels per inch. Valid entries range from 30 to 600 pixels per inch. The default is 300.</p> <p>Keep in mind that the higher the resolution and color depth, the longer it takes to create output. For example, changing the resolution from 300 DPI to 150 DPI makes the output four times smaller. The larger the bitmap, the slower the processing.</p>
BitmapPath	(Optional) Specify the location where the bitmaps are output. The default is the file path. You can enter either UNC or URL paths.
BitmapHTTP	(Optional) Specify the HTTP address of the location specified in the BitmapPath option so the bitmaps can be referenced by the MPM output. If this location does not exist, the system uses the location you specified in the BitmapPath option.
Fonts	<p>(Optional) This option only affects charts and bar codes which contain text. The MPM print driver converts these objects into bitmaps.</p> <p>Use this option to specify the fonts you intend to use, in order. For example, if you set the Fonts option to <i>PCL,TTF</i>, the system first locates the PCL font. If the PCL font does not exist, it finds the TTF font.</p> <p>The default font order is: PCL,AFP,XER,TTF,PS.</p>
MonospaceFonts	<p>(Optional) Specify the monospaced fonts you want to use as default fonts for web display. The default font list is shown here:</p> <p><code>`Letter Gothic`, `Courier`</code></p> <p>This tells the system to first use Letter Gothic and if that font is not found, use Courier.</p>

Option	Description
AddPropFonts	(Optional) Enter Yes if you want the system to use proportional fonts, then list the proportional fonts you want the system to use in the ProportionalFonts option The default is No.
ProportionalFonts	(Optional) If you entered Yes in the AddPropFonts option, specify the proportional fonts you want to use as default fonts for web display. Enclose each font name in apostrophes and separate fonts with commas. The default font list is shown here: `Arial`, `Times` This tells the system to first use Arial and if that font is not found, use Times.
Debug	(Optional) Enter Yes to tell the system to write error or warning information into a trace file for debugging purposes. The default is No.

Master Resource control group

FontLib	(Optional) Specify the font location. The default location is .\fmres\deflib\.
---------	--

You must set up the MultiFilePrint callback function if the CreatePlainText option is set to Yes when using the MPM print driver with the (single step) GenData and (multistep) GenPrint programs. Here is an example of how to set up the MultiFilePrint callback function:

```
< Print >  
  CallbackFunc = RULMultiFilePrint  
  MultiFileLog = data\datlog.dat
```

Note For more information on the MultiFilePrint callback function, see the [Documaker Administration Guide](#).

EXAMPLE MPM DOCUMENT

Here is an example of a simple MPM document:

```
MIME-version: 1.0
Content-type: multipart/alternative; boundary="--
MIMEBoundaryA62305AD43B64DADBD3A2B077B8AE71D"
```

This email is sent in multipart MIME format!

```
--MIMEBoundaryA62305AD43B64DADBD3A2B077B8AE71D
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: quoted-printable
```

Test MultiPart MIME format

```
--MIMEBoundaryA62305AD43B64DADBD3A2B077B8AE71D
Content-Type: text/HTML; charset="utf-8"
Content-Transfer-Encoding: 8bit
```

```
<HTML>
<HEAD>
<TITLE>HTML Document</TITLE>
</HEAD>
<BODY>
<TABLE>
<TBODY>
<TR>
<TD>
<P>Test MultiPart MIME format</P>
</TD>
</TR>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

```
--MIMEBoundaryA62305AD43B64DADBD3A2B077B8AE71D--
```

DESIGNING FORMS

Keep in mind...

- During form design, do not overlap FAP objects. Overlapping can cause the misplacement of objects in the output. So if you see that a FAP object is missing or misplaced, the first thing to check is whether the object overlaps another object.
- Make sure the appropriate fonts are included in the font cross reference (FXR) file and make sure the font file is in its designated location and you have properly set up the font library and path. If you see that text or a text area is missing, first check for overlapping objects and then make sure the font is listed in the FXR file and exists in the designated location.
- Do not use a horizontal line as a text underline. The horizontal line tags include space which keeps the horizontal line from being associated with the text above it. Horizontal lines are generally used as separators, such as page or paragraph separators. If, however, you position the horizontal line close enough to the text, the system automatically changes it to an underline.
- Avoid using vertical lines. These types of lines will not appear in the generic HTML format output. For example, do not use lines to compose a box or table. Instead, use box objects to compose a table.
- Because charts, bar codes, and vectors are converted into bitmaps and assigned a name based on the object name you assigned in Name field under General Properties in Studio, make sure each object has a unique name. For instance, if a chart has an object name of CHART#0001, it will be converted into a bitmap named CHART#0001.JPG. If another chart has the same object name, it will overwrite the first chart.
- When a vector is converted into a bitmap, its solid areas may overlap other objects or push other objects away.

Note The output may differ slightly because the various objects that comprise the content are positioned via relative positions in HTML format and because some spaces are inserted by HTML tags.

Chapter 9

Using the PCL Print Driver

This chapter provides information about Documaker's PCL Print Driver. This print driver is used on Windows.

This chapter includes the following topics:

- *Overview* on page 148
- *Setting PCL INI Options* on page 149
- *Defining PCL Printer Resources* on page 161

OVERVIEW

Hewlett-Packard created the Printer Control Language (PCL) to provide a way for application programs to control a range of printer features across a wide array of printing devices.

The PCL language has evolved over time. For the most part, system-produced PCL output will run on any printer that supports PCL 5 or PCL 6. There are separate drivers for these two versions of the PCL language.

To support color printing, the printer must support PCL 5c, which contains color extensions. To support more than two paper trays, the printer must support PCL 5e.

Note All system print drivers support 24-bit color graphics. The PXL (PCL 6) driver supports monochrome, 8-bit color (256 color), and 24-bit color graphics.

If your printer does not support color, the print driver will automatically convert the color graphics into monochrome graphics. Keep in mind that for the best performance you should avoid color graphics.

SETTING PCL INI OPTIONS

You must define the necessary printer options for the GenPrint program to produce PCL output. These options specify PCL output and are located in a *PrtType:xxx* control group, such as *PrtType:PCL* for PCL 5 or *PrtType:PXL* for PCL 6. Common PCL printer options are shown below, with default values in bold:

Option	Values	Description
Device	any file or device name	The name of the file or device (LPT1) where the PCL print stream should be written. This setting is ignored by the GenPrint program but is used by Studio and other Documaker system programs.
Module	PCLW32	The name of the program module which contains the PCL print driver. See also the Class option. For PCL6, enter PXLW32 . See also <i>Formdef = F1DOCUMK</i> : on page 40.
PrintFunc	PCLPrint	The name of the program function that is the main entry point into the PCL print driver. For PCL6, enter PXLPrint . See also <i>Formdef = F1DOCUMK</i> : on page 40.
Resolution	300	The dots per inch resolution of the printer which will receive the PCL data stream.
SendOverlays	Yes/No	Set to Yes if you created PCL overlays for each FAP file. This option is not supported for PCL 6.
OverlayPath	any directory	Set to the directory containing the PCL overlays for each FAP file. The default is the FormLib option of the MasterResource control group. Here is an example: <pre>< MasterResource > FormLib = <CONFIG:Batch Processing> FormLib = <CONFIG:Batch Processing> FormLib = ./forms/</pre> This option is not supported for PCL 6.
OverlayExt	any file extension (OVL)	The file extension of the PCL overlays. This option is not supported for PCL 6.
PageNumbers	Yes/ No	Set to Yes to enable form or form set page numbering.
SendColor	Yes/ No Enabled/ Disabled/Hidden	Set to Yes to enable color printing. Enabled = Option appears in the Print window and is active (available to be checked). Disabled = Option appears in the Print window but is grayed out (not available to be checked). Hidden = Option does not appear in the Print window.
HighlightColor	Yes/ No	Set this option and the SendColor option to Yes to use simple color mode. See <i>Using Simple Color Mode</i> on page 156 for more information. This option is not supported for PCL 6.

Option	Values	Description
DownloadFonts	Yes/No	Set to Yes to enable downloading of PCL fonts. For PCL6, you must enter Yes because internal font selection is not supported.
TemplateFields	Yes/No	Set to Yes to test print Xs in variable fields.
FitToWidth	Yes/No	Not supported by either PCL print driver.
AdjLeftMargin	Yes/No	Automatically adjusts the left margin to compensate for the 1/4-inch left margin added by PCL printers. Yes = Automatically adjust the left margin. Forms print exactly as they appear on screen (default). No = Do not adjust the left margin. Forms may not print correctly on PCL printers after performing a retrieve function. This option is not supported for PCL 6.
SelectRecipients	Yes/No Enabled/ Disabled/Hidden	Set to No to disable the ability to select recipients. Enabled = Appears in the Print window and is active (available to be checked). Disabled = Appears in the Print window but is grayed out (not available to be checked). Hidden = Does not appear in the Print window.
PrintViewOnly	Yes/No	If set to Yes, the view only sections will print. This does not apply to entry only sections, which are never printed. Entry only sections are usually worksheets. If the section is marked as hidden and view only, it will not print.
PrePrintedPaper	Yes,Disabled	Determines if the check box which lets you print or not print pre-printed objects appears on the Print window. Also determines the default for this field—checked or unchecked. You must add this option to the INI file if you want the field to appear on the Print window. The default includes the field on the Print window and leaves it unchecked. All objects except fields can be marked pre-printed on the object's Properties window.
Class	(first three characters of the Module option)	Specifies the printer classification, such as AFP, PCL, XER, PST, or GDI. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the Class option to make sure the correct printer is available before continuing.
StapleBin		Set this option to the PCL printer escape sequence that selects the bin that contains the staple attachment. Use a tilde character (~) in place of the binary escape character. This option is not supported for PCL 6.
PJLCommentScript		To add PJP comments to a PCL print stream, enter the name of the DAL script you want the system to run. This DAL script creates the control strings and adds them as ASCII comments. This option is not supported for PCL 6.

Option	Values	Description
PJLCommentOn	batch/formset	Use this option to add PjL comment records to the beginning of every form set or batch. This option is not supported for PCL 6.
OutputBin		Enter the printer escape sequence to select the normal output bin (for non-stapled forms) if non-stapled forms are being sent to the wrong bin. This option is not supported for PCL 6.

Note The default FAPCOMP.INI file should include the PrtType:GDI control group and options in addition to the PrtType:PCL or PrtType:PXL control group.

Using Defaults for the Module and PrintFunc Options

Default values for the Module and PrintFunc options in the PrtType:xxx control group are provided when you use a standard print type name or print class, such as AFP, PCL, PDF, PST, VPP, XER, XMP, or GDI.

These defaults keep you from having to enter the Module and PrintFunc names in your INI file. For example, if you want to generate PCL print files, you can specify these INI options:

```
< Printer >
  PrtType    = MYPCL
< PrtType:MYAFP >
  Class      = PCL
```

And the system will default these options for you:

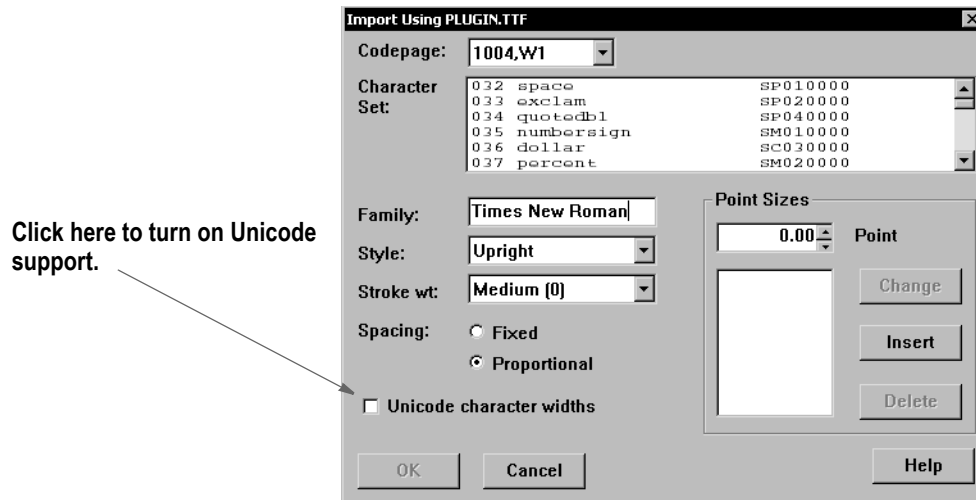
```
< PrtType:MYAFP >
  Module     = PCLPRT
  PrintFunc  = PCLPrint
```

USING PCL 6

PCL 6 is a stack-based protocol (similar to PostScript) composed of attributes and operators that let you define paths, clip paths, pens, brushes, fonts, raster patterns, and so on. PCL6 also supports 16-bit character codes which makes it a better choice for supporting Unicode than PCL 5.

The PCL 6 driver lets you download both PCL bitmap fonts and TrueType fonts. You must specify the TrueType font file name in the Font File entry of the PCL printer section in the font cross-reference (FXR) file.

To turn on Unicode support, check the Unicode Character Widths field when you insert a TrueType font into the FXR file. Unicode support lets you use additional characters and languages supported by the TrueType font.



Keep in mind...

- The PCL 6 driver supports PCL bitmap fonts so you can use master resource libraries (MRLs) designed for PCL 5. Just remember to make the appropriate changes to your INI options.
- When printing using a TrueType font, only the characters used on the form are downloaded into the print stream. This reduces the size of print stream files, particularly if the TrueType font includes support for Asian languages.

In comparison to the PCL 5 printer driver, the PCL 6 driver has these limitations:

- No overlay support
- No support for a separate downloadable font file which contains multiple PCL fonts
- No internal printer font support
- Less paper tray support, no INI options to specify which PCL commands to use
- No INI options to specify PCL commands to output bin or staple bin
- No highlight color support
- No comment script support

PRINTING UNDER WINDOWS

Windows XP/2000 does not recognize printer ports such as LPT1. If you are using Windows XP/2000, you must change the PrtType control group in the FSIUSER.INI file to reflect the print server name and print device. Here is an example:

```
< PrtType:PCL >
    Device = \\FSISRV03\\OPTRA1
```


USING HIGH-CAPACITY TRAYS 3 AND 4 ON HP 5SI PRINTERS

The system defines document attributes in a device-independent fashion. In prior versions, PCL support was based on options available to PCL 5 and similarly configured printers. The newer HP 5SI printer offers additional capabilities which depend upon (at least somewhat) commands that exist in PCL 5e. To add to the confusion, HP is not always consistent with its own terminology. Here is how the system treated PCL in prior versions:

Note The ability to define trays or use the Tray# option is not supported for PCL 6.

System term	PCL command	PCL term	HP 4 term	HP 4si term	HP 5si term
Tray 1 (Main)	~&l1H	Tray 2 (upper)	PC Tray	Upper tray	Tray 2 (upper drawer)
Tray 2 (Aux)	~&l4H	Tray 3 (lower)	MP	Lower tray	Tray 3 (lower drawer)
Tray 3 (Man)	~&l2H	Manual feed	Tray 1	Manual feed	Tray 1 (manual side feed)
Tray 4 (Env)	~&l3H	Envelope feed	Tray 1	Manual feed	Tray 1 (manual side feed)
n/a	~&l5H	HCI, first tray	LC Tray	n/a	First tray of HCI
n/a	~l20H	HCI, second tray	n/a	n/a	Second tray of HCI

The terms for the current version are shown below, with changes highlighted:

System term	PCL command	PCL term	HP 4 term	HP 4si term	HP 5si term
Tray 1 (Main)	~&l1H	Tray 2 (upper)	PC Tray	Upper tray	Tray 2 (upper drawer)
Tray 2 (Aux)	~&l4H	Tray 3 (lower)	MP	Lower tray	Tray 3 (lower drawer)
Tray 3	~&l5H	HCI, first tray	LC Tray	n/a	First tray of HCI
Tray 4	~l20H	HCI, second tray	n/a	n/a	Second tray of HCI

The command ~&l5H (first high-capacity tray) is supported by PCL 5, but the hardware is not typically found on HP printers. The command ~&l20H requires PCL 5e.

You can use these INI options:

```
< PrtType:PCL >
  Tray1 = pcl command sequence (default is ~&l1H)
  Tray2 = pcl command sequence (default is ~&l4H)
  Tray3 = pcl command sequence (default is ~&l5H)
```

Tray4 = pcl command sequence (default is ~&l20H)

Keep in mind the paper size overrides the tray selection.

Note See also for *Handling Multiple Paper Trays* on page 306 more information.

If you depend on the prior sequence, you can return to the original operation by specifying:

```
< PrtType:PCL >
  Tray3 = ~&l2H
  Tray4 = ~&l3H
```

Note The tilde (~) represents the escape character and is translated internally. The third character in each sequence shown is a lowercase *L*.

Using a Staple Attachment

In your PCL printer group, usually PrtType:PCL, add the StapleBin option to use a staple attachment on your PCL printer.

Set the StapleBin option to the PCL printer escape sequence that selects the output bin which contains the staple attachment. Use a tilde (~) in place of the binary escape character.

Here is an example:

```
~&l2G      (tilde, ampersand, lower case l, 2, upper case G)
```

This example shows the escape sequence used to select an optional lower (rear) output bin that may have a staple attachment. Check with your printer manual for the escape sequence you should use.

The OutputBin option should contain the printer escape sequence needed to select the normal output bin (for non-stapled forms). Using the OutputBin option is not necessary unless you notice the non-stapled forms are being sent to the wrong output bin. This INI option is only necessary when you have both stapled and non-stapled forms in the same print batch.

OVERRIDING PAPER SIZE COMMANDS AND TRAY SELECTIONS

You can include additional PCL 5 printer commands which you can use to override both the paper size and the tray selection. For instance, you can use this technique to get an envelope feeder to work.

Note Before the release of version 11.1, you could only specify the PCL 5 command for the system to emit when a form is specified to use a certain paper tray (for more information, see *Using High-Capacity Trays 3 and 4 on HP 5SI Printers* on page 153).

When you include a PCL paper (page) size command, the system does not emit its own paper (page) size PCL command based on the form's page size. This lets you use a page size the system does not support.

For example, suppose you want to print on #10 business envelopes (4 $\frac{1}{8}$ inch by 9 $\frac{1}{2}$ inch) using an optional envelope feeder on your PCL printer. The PCL command to select a paper (page) size for printing COM-10 (Business 4 $\frac{1}{8}$ x 9 $\frac{1}{2}$ inches) is shown here:

```
~&l81A
```

The PCL command to feed an envelope from an optional envelope feeder is shown here:

```
~&l6H
```

If your system contained a form for printing on an envelope and the form was specified to print from tray 4, you would use this INI setting:

```
< PrtType:PCL >
...
Tray4 = ~&l81A~&l6H
```

Because some characters are hard to distinguish, refer to this table for an explanation of the characters shown for the Tray4 field, in order:

Character	Description
~	A tilde
&	An ampersand
l	A lowercase L
8	The numeral eight (8)
1	The numeral one (1)
A	An uppercase A
~	A tilde
&	An ampersand
l	A lowercase L
6	The numeral six (6)
H	An uppercase H

When writing PCL commands as an INI setting, the tilde (~) is used as a substitute for the PCL escape character (x1B).

The PCL 5 Technical Reference manual contains information on PCL commands used to select paper trays and paper sizes. You can get a copy of the PCL 5 Technical Reference manual by going to the following web site and entering the phrase *PCL technical reference* in the search window:

www.hp.com

Note When printing envelopes, you may want to design your form (section) in landscape mode. When printing on PCL printers, there are unprintable margins on the left/right edge of ¼ inch and top/bottom edge of 1/6 inch. These unprintable margins apply when printing envelopes. Remember to account for these unprintable margins when designing your form (section).

USING SIMPLE COLOR MODE

The PCL print driver supports PCL simple color mode in addition to full RGB color support. PCL simple color mode uses a 3-plane CMY palette. The 3-plane CMY palette contains these indexed colors:

```
0 - White
1 - Cyan
2 - Magenta
3 - Blue
4 - Yellow
5 - Green
6 - Red
7 - Black
```

To specify highlight color printing for PCL, include these INI options:

```
< PrtType:PCL >
  SendColor = Yes
  HighlightColor = Yes
```

Marking Objects to Print in Color

For any object, such as lines, boxes, or text, select the Print in Color option on the Color Selection window if you want the object to print in a color other than black. Keep in mind...

- If the object is black and is not marked as Print in color, the system prints the object using a black color index.
- If the object has a color other than black and is marked as Print in color, the system prints it using a highlight color index.
- Charts print in black, although you can print chart labels in the highlight color.

Specifying the Highlight Color to Use

You can use these INI options to specify the PCL color commands to use for printing the black and highlight colors. The default values are shown here:

```
< PrtType:PCL >
  HighlightColorCmd = ~*v3S
  HighlightBlackCmd = ~*v7S
```

Note that the tilde (~) is used in place of the PCL escape character (hex 1B) and that the number used in the command corresponds to the color indexes specified earlier, such as 3=Blue and 7=Black.

To use a different highlight color, include the HighlightColorCmd option. To use a different black color, specify the HighlightBlackCmd option.

Printing on Different Types of Printers

Printing black and white, highlight color, and full color print streams on black and white, highlight color, and full color PCL printers will produce varying results.

You can usually send a color PCL print stream to a black and white PCL printer without any problem—everything comes out black and white. PCL printers usually ignore any commands they do not understand.

If, however, you send a highlight color PCL print stream to a full color PCL printer, the result may be slightly different than if you send it to a highlight color printer.

Bitmap graphics in a highlight color print stream may print as cyan on a full color printer. Bitmaps are a sequence of binary data—zeros (0) and ones (1)—so the zeros may print as white, while the ones may print as cyan. On a highlight color printer, the bitmap is printed as expected using the black or highlight color.

If you send a full color PCL print stream to a highlight color printer, your results may vary based on the printer model and printer settings.

CREATING COMPRESSED PCL FILES

You can create compressed PCL files using Documaker. This is typically used with IDS because Windows does not let you print files that are a mixture of simplex and duplex pages from Acrobat. The whole document has to be printed the same way. IDS, however, lets you print a file to a local PCL printer which preserves the file's duplex information.

Use these options, which call the PRTZCompressOutPutFunc function, to compress an output file, such as a PCL print batch file:

```
< PrtType:PCL >
  OutputMod = PRTW32
  OutputFunc = PRTZCompressOutputFunc
```

Note The output is compressed, regardless of the file's extension. You must decompress the file before you can print it.

Bitmap Compression

The PCL print driver also supports bitmap compression. To disable bitmap compression, add the following INI option to the PCL printer control group:

```
< PrtType:XXX >
    Compression = No
```

ADDING PRINTER JOB LEVEL COMMENTS

Printer Job Language (PCL) comments are supported by some PCL printers (not PCL 6). One type of command lets you add a comment to your PCL print stream. The PJJ comment does not affect printing but can pass information to other products that look for specific information in PJJ comment records, such as an imaging system.

Note Imaging products can be used to archive PCL print streams. These products often require a control record at the beginning of the PCL print stream. These options and DAL functions let you create that control record.

To add PJJ comments, add the following INI option to the PCL print group:

```
< PrtType:PCL >
    PJJCommentScript = imaging.DAL
```

The PJJCommentScript option specifies the DAL script you want to run. This DAL script creates a control string and adds it as an ASCII comment. Here is an example of the DAL script:

```
* Add imaging comment - use default APPIDX record.
Comment = AppIdxRec( )
AddComment(Comment,1)
Return('Finished!')
```

Notice the use of the second parameter to the AddComment DAL function. The *1* indicates the string should be an ASCII string. If you omit this parameter, the system converts the string into an EBCDIC string.

You can also use the PJJComment option to tell the system to add PJJ comments to the beginning of every form set or print batch. Here is an example:

```
< PrtType:PCL >
    PJJCommentScript = imaging.DAL
    PJJCommentOn      = formset
```

ADDING DATA FOR IMAGING SYSTEMS

The PCL print driver can add free form text or data at the beginning of a batch or each form set within the batch. This can help you interface with imaging systems such as RightFax.

Use the TEXTScript INI option to specify the DAL script you want to run. This DAL script creates a free form data or text buffer and adds it to the print stream.

Here is an example of the DAL script:

```
* Populate the PCL stream comment with these values from RCBDFD
faxnum = trim(GVM('FaxNumber'))
faxname = trim(GVM('FaxName'))
```

```
AddComment('<TOFAXNUM:' & faxnum & '>',1)
AddComment('<TONAME:' & faxname & '>',1)
```

```
Return
```

Notice the use of the second parameter to the AddComment DAL function. The 1 indicates the string should be an ASCII string. If you omit this parameter, the system converts the string into an EBCDIC string. You can also use the TEXTCommentOn option to tell the system to add free form text or data to the beginning of every form set or print batch. Here is an example:

```
< PrtType:PCL >
  TEXTScript      = imaging.DAL
  TEXTCommentOn = formset
```

LIMITING THE NUMBER OF EMBEDDED PCL FONTS

If the DownloadFonts option is set to Yes, when the GenPrint program generates PCL print the fonts used in each transaction are embedded into that transaction's portion of the print stream.

For example, if the first transaction in the print stream references fonts A and B and the second transaction references fonts A, B, and C, the section of the print stream that contains the print records for the first transaction would include embedded font data for fonts A and B. The section of the print stream that contains the print records for the second transaction would include embedded font data for fonts A, B, and C.

PCL Print Stream	With the DownloadFonts option set to Yes:
Transaction 1 (using fonts A and B)	The system includes embedded font data for fonts A and B.
Transaction 2 (using fonts A, B and C)	The system includes embedded font data for fonts A, B, and C.
Transaction 3	

Typically, some fonts are used in multiple transactions, such as A and B in this illustration. This means those fonts are embedded multiple times. You can, however, use the InitFunc and TermFunc options to tell the GenPrint program to only embed a font once. Here is an example of the INI settings you would need:

```
< PrtType:PCL >
  InitFunc      = PCLInit
  TermFunc      = PCLTerm
  DownloadFonts = Yes
```

Continuing with the example, with these settings fonts A and B would be embedded into the section of the print stream that contains print records for the first transaction and only font C would be embedded into the section of the print stream that contains print records for the second transaction. If subsequent transactions only reference fonts A, B, or C, no other fonts would be embedded into the print stream.

PCL Print Stream
Transaction 1 (using fonts A and B)
Transaction 2 (using fonts A, B and C)
Transaction 3 (using fonts A and B)
Transaction 4 (using fonts A and B)

With the InitFunc, TermFunc, and DownloadFonts options set as shown above:

The system includes embedded font data for fonts A and B.

The system includes embedded font data for font C.

No additional embedded font data is included for this transaction because fonts A and B have already been embedded.

Using these INI settings to avoid the redundant embedding of font data results in smaller print streams and faster processing times.

DEFINING PCL PRINTER RESOURCES

A number of resources are used in the printing process. These resources reside in directories specified in the MasterResource control group.

Fonts

The system supports PCL bitmap fonts. These fonts reside in the directory specified in the FontLib option in the MasterResource control group when you set the DownloadFonts option to *Yes*. The system includes utilities for creating PCL fonts from PostScript, TrueType, Xerox, or AFP fonts.

Overlays

Use the OVLCOMP utility to create PCL overlays from FAP files. These overlays must reside in the directory specified in the OverlayPath option in the MasterResource control group when you set the SendOverlays option to *Yes*.

Note Because the PCL 6 driver supports PCL bitmap fonts, you can use master resource libraries (MRLs) designed for PCL 5. Just remember to make the appropriate changes to your INI options.

Chapter 10

Using the PDF Print Driver

This chapter provides information about Documaker's PDF Print Driver. This print driver creates Adobe Portable Document Format (PDF) files from output from Oracle Documaker applications such as Documaker Server and Documaker Desktop.

This chapter includes the following topics:

- *Overview* on page 164
- *Setting Up the PDF Print Driver* on page 166
- *Customizing PDF Output* on page 183
- *Working With Fonts* on page 208
- *Adding Security to PDF Files* on page 218

OVERVIEW

Adobe Portable Document Format (PDF) is a document language developed by Adobe Systems, Inc., that allows formatting and graphics information to be stored along with text. Using PDF files, you can make sure form sets viewed and printed match the originals.

A document stored in PDF format can be transmitted to and viewed on many types of systems. There are PDF viewer applications available for many platforms, both as stand-alone programs and as add-ons for existing applications (such as word processors and Internet web browsers). You can download Acrobat Reader from Adobe Systems' web site:

www.adobe.com

Print output directed to the PDF Print Driver is written to disk and stored in one or more files. You can then view these files using Acrobat Reader.

PDF Versions

When it creates a PDF file, Documaker's PDF Print Driver begins using the lowest possible PDF version (version 1.3). If features are found in the transaction that require a higher version of Acrobat, the PDF Print Driver increases the required version accordingly.

When new features are introduced to the PDF specification, older versions of Acrobat will not support them. If you open a PDF file that includes features not supported by your version of Acrobat (or Acrobat Reader), Acrobat warns you that some features may not be supported. Here are some examples:

If you want...	You need...
The PDF viewer to display the document title.	PDF version 1.4
To specify the default page scaling option for the Print window.	PDF version 1.6
To specify the default duplex options for the Print window.	PDF version 1.7
To use the PDF page size to select the paper tray when printing.	PDF version 1.7
To specify the range of pages to be printed.	PDF version 1.7
To specify the number of copies to print.	PDF version 1.7

The PDF version numbers correspond to the following versions of Acrobat and Acrobat Reader:

This version of Acrobat	Supports
4.0 and higher	PDF version 1.3
5.0 and higher	PDF version 1.4
6.0 and higher	PDF version 1.5
7.0 and higher	PDF version 1.6
8.0 and higher	PDF version 1.7

SETTING UP THE PDF PRINT DRIVER

First make sure you have the correct system requirements to run your Documaker software. For instance, if you are using Documaker Desktop, see the [Documaker Desktop Installation Guide](#) for information on what you need to run those systems.

For Documaker Server and Documaker Studio, see the [Documaker Installation Guide](#) for more information on system requirements.

Once you have made sure you have the correct system configuration to run Oracle Documaker software and have installed your Documaker software, follow these steps to use the PDF Print Driver:

1. Customize your INI files. See *Setting PDF Options* on page 167. Also review *Additional Feature Setup* on page 171
2. Select the PDF Print Driver in your Documaker application.

Note The PDF Print Driver is installed when you install Oracle Documaker.

SETTING PDF OPTIONS

Make sure you have the following options in your FSISYS.INI file:

Note *Before making any changes to these files, back up your INI files.*

```
< Control >
  LoadPrintOnly    = Yes
< Printers >
  PrtType           = PDF
< PrtType:PDF >
  Device            = E:\TEST.PDF
  Bookmark          = Yes, Page
  DownloadFonts     = No, Enabled
  Module            = PDFW32
  PageNumbers       = Yes
  PrintFunc         = PDFPrint
  SendOverlays      = No, Enabled
  SendColor         = Yes, Enabled
  Class             = PDF
  DisplayMode       = UseOutlines
  PaperSize         = 0
  Linearize         = Yes
  SubsetAllEmbeddedFonts = Yes
  ForceColorBitmaps= No
  FontCompression  = 2
```

Option	Description
--------	-------------

Control	
---------	--

LoadPrintOnly	Enter Yes to tell the system to load print only forms.
---------------	--

Printers	
----------	--

PrtType	Enter PDF.
---------	------------

PrtType:PDF	
-------------	--

Device	<p>Enter the path and file name for the PDF output. Here is an example:</p> <pre>Device = F:\PDF\~KEYID ~DATE ; [%I-%M %p];.PDF</pre> <p>In this example, the system will write a file to the \PDF directory on the F: drive. The file name would include a document ID (KeyID), such as a policy number, the time the file was created, and it will have an extension of <i>PDF</i>. For example, the file written to disk would have a name similar to:</p> <pre>F:\PDF\A100 [3-47PM] .PDF</pre>
--------	--

Option	Description
Bookmark	<p>This option contains these values. Separate the values with commas (,).</p> <ul style="list-style-type: none"> The first value enables bookmarks. Enter Yes to tell the system to create bookmarks. The default is No. The second value indicates the lowest level for which the system will create bookmarks. You can choose from <i>Formset</i>, <i>Group</i>, <i>Form</i>, <i>PageOnly</i>, or <i>Page</i>. For example, if you enter <i>Form</i>, the system creates bookmarks for each form set, for each group in all form sets, and for each form in all of the groups. The default is <i>Page</i>. Use the third value to turn off generic bookmarks for all pages. For example, enter No if you do not want every page to have a bookmark like Page 01, Page 02, Page 03, and so on. If you only want TLE bookmarks to appear, enter No for the third value. Use the fourth value to indicate the lowest level to which bookmarks can be expanded. It takes the same values as the second value and also defaults to <i>Page</i>. <p>In the bookmark pane in Adobe Reader, only levels of bookmark above the one you specify here are expanded. If this parameter is assigned a value lower than that of the second parameter, the system automatically sets it to the value in the second parameter. If you chose <i>PageOnly</i> in the second parameter, this parameter is ignored.</p> <p>For example, this entry:</p> <pre>Bookmark = Yes, Page, Yes, Page</pre> <p>Tells the PDF Print Driver to...</p> <ul style="list-style-type: none"> Create bookmarks for each form set, group, and page Create anonymous bookmarks Expand the form set and group bookmarks in the Bookmark pane, but hide the page bookmarks.
DownloadFonts	Set to <i>No</i> , <i>Enabled</i> . Set to <i>Yes</i> if you want to embed fonts. See <i>Embedding Fonts</i> on page 210 for more information.
Module	<p>The name of the program module which contains the system's PDF print driver. See also the <i>Class</i> option.</p> <p>See also <i>Using Defaults for the Module and PrintFunc Options</i> on page 170.</p>
PageNumbers	Set to <i>No</i> if you do not want page numbers. Defaults to <i>Yes</i> .
PrintFunc	<p>The name of the program function that is the main entry point into the system's PDF print driver.</p> <p>See also <i>Using Defaults for the Module and PrintFunc Options</i> on page 170.</p>
SendOverlays	Set to <i>No</i> , <i>Enabled</i> .
SendColor	Set to <i>Yes</i> , <i>Enabled</i> .
Class	<p>Specifies the printer classification, such as AFP, PCL, XER, PST, or GDI. If you omit this option, the system defaults to the first three letters from the <i>Module</i> option.</p> <p>Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the <i>Class</i> option to make sure the correct printer is available before continuing.</p>

Option	Description
DisplayMode	<p>This option lets you control how the PDF file is initially displayed. To have the PDF file open...</p> <ul style="list-style-type: none"> • With bookmarks (document outline) visible, enter <i>UseOutlines</i> • With thumbnail images visible, enter <i>UseThumbs</i> • In full-screen mode, with no menu bar or other window controls visible, enter <i>FullScreen</i> • In default mode, with neither bookmarks or thumbnails visible, enter <i>UseNone</i>
PaperSize	<p>This option selects the paper size. The most commonly chose options are:</p> <p>0=Letter (default) 1=Legal 2=A4 3=Executive 98=Custom</p> <p>For a list of all options, see <i>Paper Sizes</i> on page 180.</p> <p>When deciding the size the system first sets the page size to the size of the first image on the page. If the page size is Custom, the page size will be set to the form size.</p> <p>If the page size is <i>Letter</i> (the default), the PDF Print Driver checks the PaperSize option. If the PaperSize is specified, the system uses it to determine the size.</p> <p>If the PaperSize option is set to Custom and page size is less than Letter, the page size will set to Letter. Otherwise, the system uses the custom width and height.</p>
Linearize	<p>Enter Yes to create linearized PDF files. See <i>Creating Linearized PDF Files</i> on page 192 for more information.</p>
SubsetAllEmbeddedFonts	<p>The default is Yes, which tells the PDF Printer Driver to embed only the font glyphs used in the document when it creates a PDF file. This is true for both normal PDF files and PDF/A-1b-compliant files. The result is smaller PDF files.</p> <p>If you enter No, the PDF Printer Driver embeds all of the glyphs for the fonts if embedding is also enabled.</p> <p>See <i>Subsetting Fonts</i> on page 214 for more information.</p>
ForceColorBitmaps	<p>Enter Yes if you want the PDF Driver to print images in color even if the images are not set to print in color (the image does have to be a color image, of course). The default is No.</p> <p>See <i>Forcing the PDF Driver to Print Color Images</i> on page 197 for more information.</p>
FontCompression	<p>Use this option to compress embedded fonts. Enter zero (0), 1, 2, or 3 to indicate the level of compression. Zero indicates no compression and three indicates the highest level of compression. The default is two (2).</p>

Option	Description
TrackFieldEdits	<p><Control></p> <p>TrackFieldEdits = Yes (Default is No)</p> <p>This option causes PDFs generated by the system to highlight those fields that were manually edited by the user.</p> <p>This option has several considerations:</p> <ol style="list-style-type: none">1.The same setting, TrackFieldEdits must also be set to Yes in the <Control> group. Note – this option allows changes to display to the user while editing in Documaker Desktop, WIP Edit or Data Entry Check within Studio.2.If the users empties a field of content, nothing will be highlighted.3.Fields that display something other than text will not be highlight, even if the underlying value is changed - e.g. barcode fields.4.Text areas with embedded fields will highlight the only the field content as changed.5.Multiline text fields will highlight in entirety if edited or assigned content via paragraph selection. Note that fields embedded within the content will show independent highlights if you don't edit the multiline text content directly.6.Charts or other objects that might use hidden fields to gather data will not highlight even if the fields that are referenced in the object change.)

Using Defaults for the Module and PrintFunc Options

Default values for the Module and PrintFunc options in the PrtType:xxx control group are provided when you use a standard print type name or print class, such as AFP, PCL, PDF, PST, VPP, XER, XMP, or GDI.

These defaults keep you from having to enter the Module and PrintFunc names in your INI file. For example, if you want to generate PDF print files, you can specify these INI options:

```
< Printer >
  PrtType   = MYPDF
< PrtType:MYAFP >
  Class     = PDF
```

And the system will default these options for you:

```
< PrtType:MYAFP >
  Module     = PDFPRT
  PrintFunc  = PDFPrint
```

3. If you have a Printers control group, simply add this option to that control group:

```
PrtType = PDF
```

If you do not have a Printers control group, add this control group and option. For example, your Printers control group might look like this:

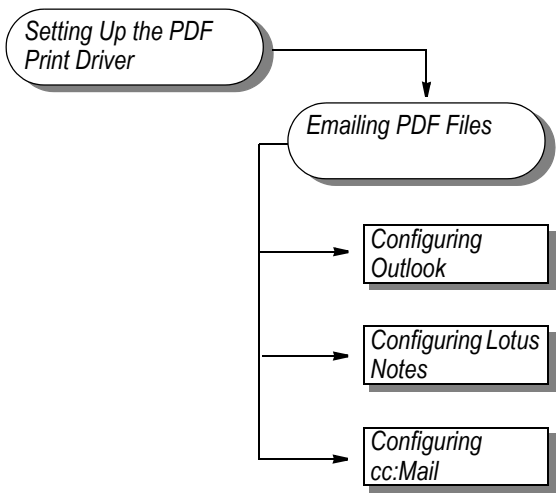
```
< Printers >
  PrtType = PDF
```

Your system can now use the PDF Print Driver. The following topic describes some additional features you can set up to work with the PDF Print Driver.

Note If you are using strong encryption, you must use Acrobat 5.x or higher. Otherwise, you can use Acrobat version 4.x or higher.

ADDITIONAL FEATURE SETUP

This illustration shows the various steps involved in setting up the PDF Print Driver, setting up your system to email PDF files, and configuring various email systems. It also shows how to set up the PDF Print Driver and then configure your system to send recipient output to multiple printers.



This table shows your options:

For information on	See	You must perform this task
Installing and configuring the PDF Print Driver	<i>Setting Up the PDF Print Driver</i> on page 166	Before you can email PDF files or print recipient output to multiple devices
Emailing PDF files	<i>Emailing PDF Files</i> on page 172	After setting up the PDF Print Driver
Configuring Microsoft Outlook	<i>Configuring Outlook</i> on page 173	After setting up the PDF Print Driver and setting up your system to email PDF files
Configuring Lotus Notes	<i>Configuring Lotus Notes</i> on page 173	After setting up the PDF Print Driver and setting up your system to email PDF files.
Configuring cc:Mail	<i>Configuring cc:Mail</i> on page 173	After setting up the PDF Print Driver and setting up your system to email PDF files

Emailing PDF Files

Once you set up the PDF Print Driver, make the following changes in your INI files to set up your system to email files.

Note You can find additional information on email support in the [Documaker Desktop Administration Guide](#).

Your INI files should have these settings:

```
< Printers >
  PrtType    = EPT
< PrtType:EPT >
  FileName   = F:\PDF\~KEYID ~DATE ; [%I-%M %p];.PDF
  InitFunc   = EPTInit
  KeepFile   = Yes
  Message    = PDF File attached...
  Module     = EPTW32
  PrintFunc  = EPTPrint
  PrtType    = PDF
  RecipFunc  = CSTSetMailRecip
  RecipMod   = CSTW32
  Subject    = PDF File from Documaker
  TermFunc   = EPTTerm
```

The FileName option is where you specify the path and file name for the PDF file. Be sure to include the spaces and characters as noted.

In this example, the system writes a file to the \PDF directory on the F: drive. The file name includes the policy number (KeyID), the time the file was created, and an extension of *PDF*.

For example, the file written to disk might look like the one shown here:

```
F:\PDF\A100 [3-47PM] .PDF
```

When you finish with these changes, you can then perform one of the following tasks to configure your email system.

- *Configuring Outlook* on page 173
- *Configuring Lotus Notes* on page 173
- *Configuring cc:Mail* on page 173

Configuring Outlook

Once you set up the PDF Print Driver, configure your system to email PDF files, and follow the steps below, you can use the system to create a PDF file of a form set and then email that file via Outlook to a recipient.

Add these settings to your FSIUSER.INI file using a text editor:

```
< Mail >
  MailType = MSM
< MailType:MSM >
  MailFunc = MSMMail
  Module   = MSMW32
  Name     = MSMail
  Password = (intentionally left blank)
  UserID   = (user ID to access mail)
```

If you are unsure of the user ID, check your control panel for mail setting or with your Administrator.

Configuring Lotus Notes

Once you set up the PDF Print Driver, configure your system to email PDF files, and follow the steps below, you can use the system to create a PDF file of a form set and then email that file via Lotus Notes to a recipient.

Add these settings to your FSIUSER.INI file using a text editor:

```
< Mail >
  MailType = VIM
< MailType:VIM >
  MailFunc = VMMail
  Module   = VIMW32
  Name     = Notes (Name of the Notes Server)
  Password =
  UserID   =
```

Configuring cc:Mail

Once you set up the PDF Print Driver, configure your system to email PDF files, and follow the steps below, you can use the system to create a PDF file of a form set and then email that file via cc:Mail to a recipient.

Add these settings to your FSIUSER.INI file using a text editor:

```
< Mail >
  MailType = CCM
< MailType:CCM >
  MailFunc = VMMail
  Module   = VIMW32
  Name     = CCMail (Name of the mail server)
  Password =
  UserID   =
  DataPath = V:\ccdata (Default path for the email system)
```

USING THE PDF PRINT DRIVER WITH GENPRINT

The GenPrint program creates the print stream for each recipient batch and sends it to a printer output file. A *batch active* flag tells PRTLIB's installed output function to keep the current file open and to append each output group to the active stream, only closing the file at the end of the batch.

In most GenPrint processing situations, this is how you want it to work. For PDF, however, you need to break each recipient record into a separate file. The following topics describe how to send each form set to a separate file.

Changing the GenPrint Program

Here are examples of how you can set up the PDF Print Driver depending on whether you use single-step or multiple step processing.

Example 1 - For multiple step processing

Use the MultiFilePrint() callback function when running the Documaker Server (GenPrint) in multiple step mode. This callback function is included with the GenPrint program and lets you create a separate file for each transaction.

This example modifies the FSISYS.INI file to set up the PDF Print Driver for use with Documaker Server running in multiple step processing mode, meaning the GenTrn, GenData, and GenPrint programs are run separately.

Open the FSISYS.INI file with a text editor and add these options:

```
< Print >
  CallbackFunc = MultiFilePrint
  MultiFileLog = data\pdflog.dat
< Printer >
  PrtType      = PDF
< PrtType:PDF >
  Device       =
  Module       = PDFW32
  PrintFunc    = PDFPrint
< Printer1 >
  Port         = data\pdfbat1.pdf
```

Use the Port option to specify the path and file name for the PDF file. In this example, the system creates a file in the *data* directory. The file name will be *pdfbat1* and it will have an extension of *pdf*.

Your system can now use the PDF Print Driver to convert transactions or form sets into PDF files.

Example 2 - For single-step processing

This example modifies the FSISYS.INI file to set up the PDF Print Driver for use with Documaker Server running in single-step processing mode, meaning the GenTrn, GenData, and GenPrint programs are combined to run as a single step.

Open the FSISYS.INI file with a text editor and add these options:

```
< PrintFormset >
  MultiFilePrint= Yes
  MultiFileLog = data\pdflog.dat
< Printers >
  PrtType      = PDF
< PrtType:PDF >
```

```
Device      =  
Module      = PDFW32  
PrintFunc   = PDFPrint  
< Printer1 >  
Port        = data\pdfbat1.pdf
```

Use the Port option to specify the path and file name for the PDF file. In this example, the system creates a file in the `\data` directory. The file name will be *pdfbat1* and it will have an extension of *pdf*.

Your system can now use the PDF Print Driver to convert transactions or form sets into PDF files.

Setting the CheckNextRecip Option

When you use the PDF Print Driver, make sure the CheckNextRecip INI option is set to No (the default is No.) The GenPrint program uses this option to look ahead to subsequent recipient records and queue up recipient records that match the same form set.

This improves system performance when many recipient batch records are placed in the same print batch and sorted together. However, it is essential that each recipient record be viewed as a separate print transaction for this to work. Without this option disabled, each file will contain multiple recipients for the same form set, which is probably not what you want.

Note With the release of version 11.2, the default for this option is No. In prior releases, the default was Yes.

Using Overlays

You cannot use overlays with the PDF Print Driver. There is no way to generate PDF overlays or use them at print time. Because of this, the GenPrint program ignores the SendOverlays option when it prints to the PDF Print Driver. FAP files and bitmaps must be loaded, which is indicated by setting these INI options:

```
DownloadFAP = Yes  
LoadFAPBitmap = Yes
```

Using the MultiFilePrint Callback function

If you specify the MultiFileLog option in the Print control group, the specified file is created at the start of the GenPrint program when the callback is installed. The file is closed at the end of the GenPrint program when the callback is uninstalled.

At the end of each transaction, a new output file name is constructed and the GenPrint program's normal behavior of only outputting to one file is overridden. MultiFilePrint makes the following assumption about the output file name:

```
XXXX####  
XXXX = four characters that are preserved.  
#### = four characters set to a zero-filled sequence number.
```

MultiFilePrint assumes that the original print batch name ends in 0001. The second file opened will be 0002, and so on, up to 9999. MultiFilePrint assumes that no single recipient batch contains more than 9999 recipient batch records. If this is the case, a custom version of MultiFilePrint is required.

Avoid this approach, however, since this is a large number of output files to attempt to track and manage. MultiFilePrint does work with multiple print batches, and each batch can contain up to 9999 recipient records.

If you turned on logging, as each file is completed the system creates a log record in the log file you specified.

Scaling and Cropping Graphics

You can use the Scale and Crop parameters to the AddMultiPageBitmap rule to resize loaded graphics or remove parts of those graphics. See the discussion of the AddMultiPageBitmap rule in the Rules Reference for more information.

Using the Log File

The log record has the following format:

```
;FIELD1;FIELD2;FIELD3;FIELD4;FIELD5;FIELD6;FIELD7;FIELD8;FIELD9;
```

```
FIELD1 = Logical recipient batch file name  
FIELD2 = Physical (full file name) recipient batch file  
FIELD3 = Group name 1 (such as Company)  
FIELD4 = Group name 2 (such as LOB)  
FIELD5 = Group name 3 (usually empty)  
FIELD6 = TransactionId (such as Policy No.)  
FIELD7 = Transaction type  
FIELD8 = Recipient type (specified in the POLFILE or FORM.DAT files)  
FIELD9 = Print output file (full file name)
```

The log file is provided for use by a custom application and implementation to handle the management and distribution of the many individual output files.

Caching Fonts

Caching fonts lets you load commonly used fonts into memory. To enable font caching, make sure these options are in your INI file:

```
< PrtType:PDF >  
  InitFunc  = PDFInit  
  TermFunc  = PDFTerm  
  CacheFiles= 16
```

The default for the CacheFiles option is 16, meaning 16 fonts are kept in memory. If you set this option to a higher value, more fonts are kept in memory and more memory is devoted to storing fonts, which may slow performance.

Generating Separate PDF Files

You can generate separate files for each transaction when you choose PDF (or RTF) from WIP or batch print.

The name of the files will have a rolling number appended to the end of the name that starts the process and is filled in on the Print window. This is automatically handled and you do not have to set INI options to get the WIP or batch print to work as long as your PrtType name is PrtType:PDF.

There are several INI options you can use to override the naming process and also name other print drivers that require this unique handling.

```
< BatchPrint >
  NoBatchSupport= PDF
  PreLoadRequired= PDF
```

These are the default settings and cannot be overridden. However, you can specify other PrtType print driver definitions you want to fall into these same categories.

Option	Description
NoBatchSupport	Indicates that the named PrtType items, separated by semicolon, do not really support batch transactions and require special handling.
PreLoadRequired	Lets you specify all the PrtType items, separated by semicolon, that should be forced to load the form set prior to the starting print. Most print drivers don't require this special requirement, but some, such as PDF do.

Also, you can name PrtType specific items under the BatchPrint control group to override the normal Device naming option. Here is an example:

```
< BatchPrint >
  PDF = ~HEXTIME .PDF
  RTF = ~HEXTIME -~KeyID .RTF
```

Any batch print sent to PrtType:PDF (picking PDF on the Print window) will override the name and store the current hexadecimal date and time, such as BCF09CA4.PDF, which is an eight-character name, as the name of each transaction's output.

Also, you can combine INI built-in calls as shown in the RTF example. Here any WIP or batch print sent to RTF will name the files using the HEXTIME and the KeyID from the WIP transaction. This will result in names similar to this: BCF099A4-123456.RTF

Note that you must leave a space after the built-in INI function name for it to work properly. That space will not appear in the resulting output name.

Generating a Single PDF File

In addition to generating separate files for each transaction, the PDF Print Driver can print an entire batch of transactions to a single PDF file. To print an entire batch of transactions to a single PDF file, add this option:

```
< PrtType:PDF >  
    SpoolBatches = Yes
```

Option	Description
SpoolBatches	<p>Enter Yes to tell the PDF Print Driver to print an entire batch of transaction to a single PDF file. The default is No.</p> <p>Keep in mind...</p> <ul style="list-style-type: none">• To create a linearized PDF file (for page-at-a-time downloading), you must set this option to No.• If you are using single step processing, you must set this option to No.

You must also turn off the MultiFilePrint callback function. To do this, remove or comment out the CallbackFunc option. Here is an example of commenting out the CallbackFunc option:

```
< Print >  
;    CallbackFunc = MultiFilePrint
```

You cannot generate linearized PDF files if you set the SpoolBatches option to Yes because when linearizing, the PDF Print Driver must have the entire contents of the PDF file in memory. Since batches can be very large, it is not practical to keep an entire batch in memory.

For more information on creating linearized PDF files, see *Creating Linearized PDF Files* on page 192.

LIMITATIONS

The PDF Print Driver does not currently support the full set of Adobe Acrobat PDF capabilities. The following are some of the product's limitations.

Type 1 fonts

The PDF Print Driver checks to see if the beginning of the setup data matches one of the system font names. If the text matches one of the fonts, the print driver checks the font attributes for weight and style (italic or not), and uses that information to complete the mapping.

For example, if the setup data starts with *Albany*, the font has a weight of zero, and it has the italic style set, the PDF Print Driver will map it to Helvetica-Oblique.

PDF objects

Although Acrobat Reader supports variable fields, radio buttons, push buttons, and list boxes, the PDF Print Driver does not support the creation of these objects within a PDF file.

The PDF Print Driver does support hypertext links created in text labels, variable fields, and logos. You create these links in Documaker Studio as you build the sections that comprise your forms.

Note Keep in mind the PDF Print Driver does support Unicode with TrueType fonts and it supports all paper sizes.

Acrobat and PDF versions

Since some settings require specific PDF versions, you should consider the version of Acrobat or Adobe Reader needed to fully support each PDF version.

This version of Acrobat	Supports
4.0 and higher	PDF version 1.3
5.0 and higher	PDF version 1.4
6.0 and higher	PDF version 1.5
7.0 and higher	PDF version 1.6
8.0 and higher	PDF version 1.7

If you are creating documents that will be distributed widely, consider only using features that only require Acrobat 5 (PDF 1.4) or Acrobat 6 (PDF 1.5). This will help make sure all users can view and print your documents correctly.

PAPER SIZES

Here is a complete list of all the paper sizes you can choose from in the PaperSize INI option:

For	Enter
US letter	zero (0). This is the default.
US legal	1
ISO A4	2
US executive	3
US ledger	4
US tabloid	5
US statement	6
US folio	7
US fanfold	8
ISO A0	20
ISO A1	21
ISO A2	22
ISO A3	23
ISO A5	25
ISO A6	26
ISO A7	27
ISO A8	28
ISO A9	29
ISO A10	30
ISO 2A	32
ISO 4A	34
ISO B0	40
ISO B1	41
ISO B2	42
ISO B3	43
ISO B4	44
ISO B5	45

For	Enter
ISO B6	46
ISO B7	47
ISO B8	48
ISO B9	49
ISO B10	50
ISO 2B	52
ISO 4B	54
ISO C0	60
ISO C1	61
ISO C2	62
ISO C3	63
ISO C4	64
ISO C5	65
ISO C6	66
ISO C7	67
ISO C8	68
ISO C9	69
ISO C10	70
ISO DL	71
JIS B0	80
JIS B1	81
JIS B2	82
JIS B3	83
JIS B4	84
JIS B5	85
JIS B6	86
JIS B7	87
JIS B8	88
JIS B	89

For	Enter
JIS B10	90
custom	98

CUSTOMIZING PDF OUTPUT

There are a number of ways you customize the PDF files you produce with the PDF Print Driver.

This topic discusses your options:

- *Producing Optimal PDF Output* on page 184
- *Reducing PDF File Sizes* on page 186
- *Setting PDF Viewer Preferences* on page 189
- *Creating Linearized PDF Files* on page 192
- *Setting Up Bookmarks* on page 193
- *Adding Hypertext Links* on page 196
- *Adding Digital Signature Placeholders* on page 196
- *Forcing the PDF Driver to Print Color Images* on page 197
- *Meeting the PDF for Archive Specification* on page 198
- *Emulating Duplex Printing from the PDF Print Driver* on page 200
- *Interfacing with Imaging Systems* on page 201
- *Creating Section 508-compliant PDF Files* on page 202
- *Examples* on page 203

PRODUCING OPTIMAL PDF OUTPUT

To produce optimal PDF output, make the following changes in your font cross-reference (FXR) file.

Note The tools used to customize a font cross-reference file are included with the full PPS system and in Documaker Server. For PPS runtime users, the people who create the libraries of form sets you use will have these tools and are responsible for setting up the FXR correctly.

- Remove font IDs built from the FORMSX font.

This is not needed in Metacode conversions and there is no equivalent built-in Acrobat font. Furthermore, the original entry does not contain character width information.

- Fix point size settings for font IDs.

Metacode fonts do not contain point size information. Therefore, a point size is approximated when a Metacode font is inserted into the FXR file. Sometimes, this approximation is incorrect. Many Metacode fonts include the point size as a part of its file name. For example, font ID 5 was built from the Metacode font AR07BP but is listed as having a point size of 8. However, AR07BP is really a 7-point font and the point size for this font ID should be changed to 7.00.

- Remove font IDs built from landscape fonts.

For landscape fonts, where the equivalent portrait fonts are included in the FXR, the font IDs for the landscape fonts should be deleted and the landscape font name should be in the Rotated Fonts field of the portrait font. For example, font ID 4 was built from the landscape font AR07BL (Arial Bold 7-point). Font ID 5 was built from the portrait font AR07BP (Arial Bold 7-point). Therefore, font ID 4 was deleted and `;AR07BL` was added to the Rotated Font Files field in the Metacode section of the Printers tab of the FXR file.

- Remove non-text fonts from the FXR file.

You may decide to leave a non-text font in the FXR if you have an equivalent TrueType or PostScript font to embed and you have enabled font downloading. If not, remove the font IDs for non-text fonts.

Note Two non-text fonts are included in the base Acrobat fonts: Symbol and Zapf Dingbats.

If the non-text font contains a signature or graphic, such as a company logo, convert the font to a graphics (LOG) file. Fonts of this style have contiguous characters and are always referenced one way. For example, the font JOHND0 might contain only the letters *A*, *B*, and *C*. When the letters *ABC* are printed together using the JOHND0 font, the signature *John Doe* is printed. When a font is always used with a single contiguous set of characters, convert the font to a LOG file.

For non-text fonts that contain characters which will be printed using a variety of combinations, you cannot use a LOG file. In this case, make the Xerox font available in the directory specified by the FontLib option in the MasterResource control group.

Examples of these types of non-text fonts include OCR, MICR, and bar code fonts. For example, a ZIP code (bar code) font produces a different picture when different ZIP codes are used. The ZIP code for 30309 looks different than the ZIP code for 49501. Using this approach, a bitmap image is created from the Xerox font using the specified characters (30309, 49501, and so on.) in the print stream. Only use this approach for fonts that are used infrequently because it results in larger PDF files which affects the time it takes to create and download a PDF file.

- Make sure color bitmaps are not saved as Comp TIFF or Comp Pack. These compression methods are not supported by PDF.

The Comp TIFF format is designed to compress monochrome bitmaps, not color ones. The Comp Pack format is only useful when the color bitmap is 16 or 256 colors. There is no reason to use Comp Pack on a full-color (24-bit) bitmap.

In most cases, you can simply leave full color bitmaps as JPEG or bitmap files and not convert them at all. The only reason to convert these files to the LOG format is to move them to a platform that does not support those file types, like MVS.

Note The PDF driver supports monochrome, monocolored, 8-bit color (256 color) and 24-bit color graphics.

- Run the FXRVALID utility to check the FXR file.

The FXRVALID utility reports missing font files, incorrect built-in Acrobat font names, and so on. Correct any errors reported by the FXRVALID utility.

- Avoid specialty fonts when mapping to built-in Acrobat fonts.

The built-in Acrobat fonts include Courier, Helvetica, and Times plus a couple of fonts containing non-text characters (Symbol and ZapfDingbats). Fonts such as Arial and Univers are pretty similar to Helvetica in terms of appearance and size and the built-in Acrobat font can often be used without any noticeable effect. Some font vendors also supply versions of the fonts where the characters are condensed (narrow) or expanded (wide). Although these fonts may have a similar character appearance, their size has been altered in a way that makes mapping to a built-in Acrobat font problematic.

REDUCING PDF FILE SIZES

There are several ways to customize the PDF Print Driver so it produces PDF files more efficiently. For instance, you can adjust the PDF compression option, make JPEG compression the default method for compressing graphics (LOG files), and use compact fonts.

Setting PDF Compression Options

You can choose from these PDF compression methods:

Choose	For
0 (zero)	no compression
1	best speed
2	default compression
3	best compression

To override the default, add the Compression option in the PrtType:PDF control group in the DAP.INI file.

```
< PrtType:PDF >  
  Compression = 3
```

You can test the various compression options to see what works best for your implementation by comparing...

- The time difference between the request to view the transaction in Acrobat Reader and when it is displayed.
- The size of the PDF file after it is retrieved.

You can also control how much compression is used for fonts embedded into your PDF files. To do this, see the FontCompression option, discussed on [169](#).

Using JPEG Compression

Depending upon the options you choose, you can see a very significant size reduction in the PDF files you produce. For instance, if you are using 24-bit color graphics, you will see the greatest reduction. Also, if you are using embedded fonts and sub-setting the fonts instead of embedding all of them, you will see the file size decrease.

```
< PrtType:PDF >  
  JPEGCompression = Yes
```

Note For more information on embedding and sub-setting fonts, see *Working With Fonts* on page 208.

If compression is turned on, and by default the PDF Print Driver uses compression method 2, 24-bit color images are compressed using the JPEG compression method. If compression is turned on, 8-bit graphics loaded using the **AddMultiPageBitmap rule** are compressed. This is a *lossy* compression method.

You can, however, use the JPEGCompression option to disable JPEG compression, as shown here:

```
< PrtType:PDF >  
    JPEGCompression = No
```

A lossy compression method is one where compressing data and then decompressing it retrieves data that may differ from the original, but is close enough to be useful. Lossy compression is typically used to compress multimedia data (audio, video, still images), especially in applications such as streaming media and internet telephony.

On the other hand, lossless compression is preferred for text and data files, such as bank records, text articles, and so on. Most lossy compression formats suffer from generation loss: repeatedly compressing and decompressing the file will cause it to progressively lose quality. This is in contrast with lossless data compression.

Lossless data compression allows the exact original data to be reconstructed from the compressed data. Use lossless compression when it is important that the original and the decompressed data be identical. For instance, you would use lossless compression for executable programs and source code. Some image file formats, like PNG or GIF, use only lossless compression, while others like TIFF and MNG use either lossless or lossy methods.

Using Compact Fonts

The PDF Print Driver supports Adobe's Compact Font Format (CFF) for embedding Type 1 fonts. Type 1 fonts, also known as PostScript fonts, are a type of scalable font created by Adobe. Several Type 1 fonts are shipped with Documaker.

CFF provides for a smaller embedded font program. The amount of space savings depends on several factors, including:

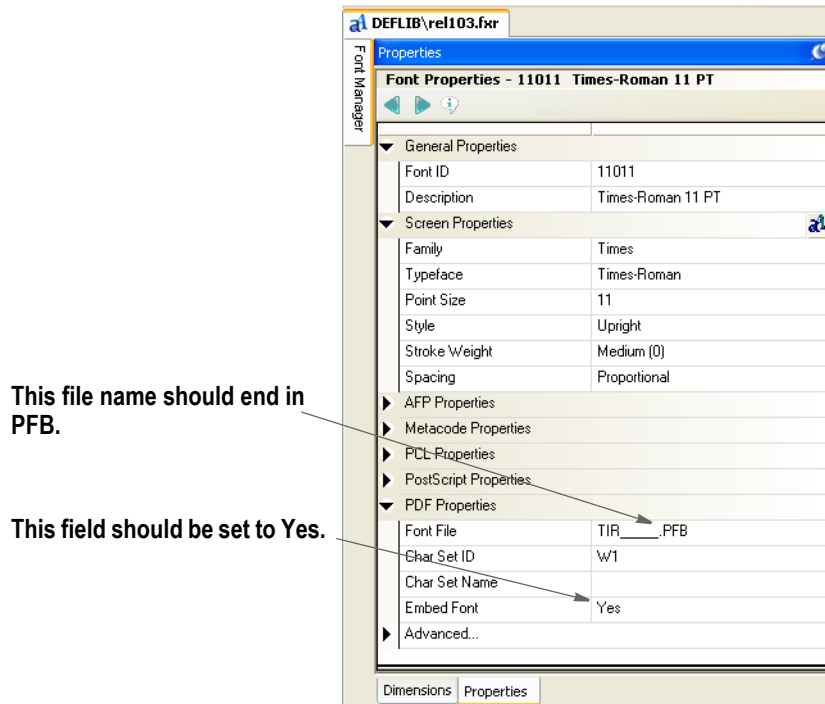
- The number of characters used by the PDF
- Whether you are subsetting fonts (the default behavior)
- Whether you are using font compression

For example, when using every character from 32 (space) to 255 (ydieresis) and default compression, the Times-Roman Type 1 font supplied with Documaker uses 48,114 bytes in the Type 1 format, but only 21,730 bytes in the Compact Font format.

To use compact fonts, first make sure font downloading is enabled with this INI setting:

```
< PrtType:PDF >  
    DownloadFonts = Yes
```

In addition, the fonts in your FXR should be set up to download Type 1 font programs when using PDF. This can be done using Studio's Font manager. Under the PDF Properties for the font, make sure the Font file ends with the extension *.PFB* (indicating that the font file to be embedded is an Adobe Type 1 font) and the Embed Font option is set to Yes.



Note that you do not have to get CFF fonts or convert the fonts you currently have. By default, the PDF Print Driver automatically converts your Type 1 fonts for you whenever a Type 1 font is requested. To change this behavior, set the UseCompactFonts option to No:

```
< PrtType:PDF >
  UseCompactFonts = No
```

Option	Description
UseCompactFonts	Enter No if you do not want to use Adobe's Compact Font Format (CFF) for embedding Type 1 (PostScript) fonts. The default is Yes. Using CFF results in smaller PDF files.

SETTING PDF VIEWER PREFERENCES

You can specify a set of viewer preferences for the PDF files you create. This includes, for instance, whether to display the tool and menu bars and defaults for printing options. To add viewer preferences to a PDF file, first add the following INI option:

```
< PrtType:PDF >
  ViewerPreferences = PDFViewerOptions
```

Option	Description
ViewerPreferences	<p>Enter the name of the INI control group that contains the PDF viewer options. Letting you choose the name of this control group gives you the option of defining multiple sets of PDF viewer options, however, you also have to remember the control group names you assign. Choose your names so they will be easy to remember and meaningful to other users.</p> <p>For instance, if you need two sets of options to handle different security settings, you could name the groups as shown here:</p> <pre>PDFViewerOptions_MinSecurity PDFViewerOptions_MaxSecurity</pre>

Note The name of your PDF printer group may differ from the one shown in the example (PrtType:PDF).

The ViewerPreferences option specifies the control group where you specify the following viewer preferences.

```
< PDFViewerOptions >
  HideToolbar =
  HideMenubar =
  HideWindowUI =
  FitWindow =
  CenterWindow =
  DisplayDocTitle =
  NonFullScreenPageMode =
  Direction =
  PrintScaling =
  Duplex =
  PickTrayByPDFSize =
  PrintPageRange =
  NumCopies =
```

The printing related options (PrintScaling, Duplex, PickTrayByPDFSize, and NumCopies) merely set defaults for the Print window. You can change these defaults at print time.

Option	Description
HideToolbar	Enter Yes if you want the PDF viewer to hide the toolbar. The default is No.
HideMenubar	Enter Yes if you want the PDF viewer to hide the menu bar. The default is No.
HideWindowUI	Enter Yes if you want the PDF viewer to hide items such as scroll bars and navigation controls. The default is No.
FitWindow	Enter Yes if you want the PDF viewer to size its window to the page contents. The default is No.

Option	Description
CenterWindow	Enter Yes if you want the PDF viewer to center its window on the screen. The default is No.
DisplayDocTitle Requires PDF version 1.4.	Enter Yes if you want the PDF viewer to display the document title. The default is No, which tells the viewer to display the PDF file name.
NonFullScreenPageMode	<p>This option tells the PDF viewer what to do when the user exits full-screen mode. This is only used if the DisplayMode option of the PDF printer group is set to <i>FullScreen</i>.</p> <p>You can choose from these options:</p> <p>UseNone - None of outline, thumbnail, or option content panes are visible</p> <p>UseOutlines - The outline pane is visible</p> <p>UseThumbs - The thumbnail image pane is visible</p> <p>UseOC - The optional content pane is visible</p> <p>The default is UseNone.</p>
Direction	Enter L2R to specify the reading order for text as left to right. Enter R2L to specify the reading order as or right to left. The default is L2R.
PrintScaling Requires PDF version 1.6.	<p>Use this option to specify the default page scaling option for the Print window. You can choose from these options:</p> <p>Requires PDF AppDefault - Use the application's current default value</p> <p>Version 1.6 None - Do not scale the page</p> <p>The default is AppDefault.</p> <p>Note: When you are creating PDF files that contain bar codes, set this option to None. Otherwise, Acrobat may stretch the output to fit, which will distort the bar code.</p> <p>If you don't use "PrintScaling=None", the print dialog will use Acrobat/Adobe Reader's current setting for scaling.</p>
Duplex Requires PDF version 1.7.	<p>Use this option to specify the default duplex options for the Print window. You can choose from:</p> <p>Simplex - Print single-sided</p> <p>DuplexFlipShortEdge - Print duplex and flip on the short edge of the sheet</p> <p>DuplexFlipLongEdge - Print duplex and flip on the long edge of the sheet</p>
PickTrayByPDFSize Requires PDF version 1.7.	<p>Enter Yes if you want to use the PDF page size to select the paper tray when printing.</p> <p>Enter No if you do not want the PDF viewer to use the PDF pages size to select the paper tray.</p>
PrintPageRange Requires PDF version 1.7.	<p>Use this option to specify the range of pages to be printed. The format is <code>firstpage-lastpage</code> where <i>firstpage</i> and <i>lastpage</i> are the beginning and ending page numbers. Here is an example:</p> <p><code>PrintPageRange = 2-7</code></p> <p>This tells the system to print pages two through seven of the document. The default varies, depending on the viewer.</p>
NumCopies Requires PDF version 1.7.	Enter a number between two and five (inclusive) to specify the number of copies to print. The default varies, depending on the viewer.

Note See also the DisplayMode option in the PrtType:PDF control group when setting viewer preferences.

Since some of these settings require specific PDF versions, you should consider the version of Acrobat or Adobe Reader needed to fully support each PDF version.

This version of Acrobat	Supports
4.0 and higher	PDF version 1.3
5.0 and higher	PDF version 1.4
6.0 and higher	PDF version 1.5
7.0 and higher	PDF version 1.6
8.0 and higher	PDF version 1.7

If you are creating documents that will be distributed widely, consider only using features that only require Acrobat 5 (PDF 1.4) or Acrobat 6 (PDF 1.5). This will help make sure all users can view and print your documents correctly.

CREATING LINEARIZED PDF FILES

Use the Linearize INI option to specify whether you want linearized or non-linearized PDF files.

With a linearized PDF file, a browser can display the first page of the PDF file before it finishes loading the entire file. This lets those who are using the PDF file start working with it sooner.

Also, if you are making the PDF files available via byte-serving web servers (servers that allow a user to request a specific range of bytes), linearization lets the user jump to any page in the PDF without having to first load all of the pages in between.

So if you are creating large PDF files which may be accessed via the internet, you probably want to linearize them.

Note Sometimes this is called *optimizing a PDF file*.

`Linearize = Yes`

Set this option to Yes to create a linearized PDF file. Setting this option to No tells the PDF Driver to produce a non-linearized PDF file.

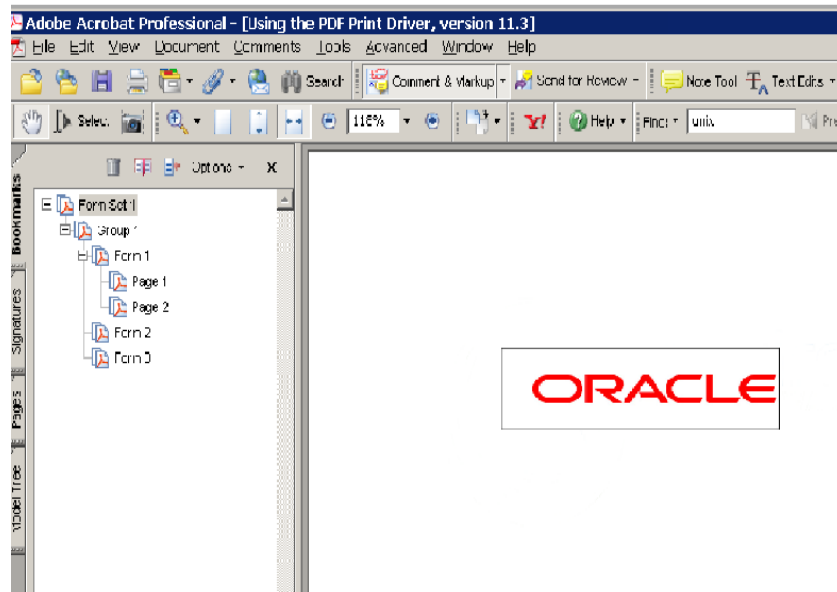
You cannot linearize a PDF file if the SortBatches option is set to Yes, as it would be if you were generating a single PDF file that contained all of your transactions. For more information, see *Generating a Single PDF File* on page 178.

SETTING UP BOOKMARKS

The PDF Print Driver sets up bookmarks at these levels by default:

- Form set - The text for this bookmark is the recipient name if applicable, otherwise it is the name of the form set.
- Group - The text for this bookmark is the name of the group.
- Form - The text for this bookmark is the description of the form. If there is no form description, the PDF Print Driver uses the name of the form.
- Page - The text for this bookmark is the name of the page.

If the text for any of the bookmarks is blank, the PDF Print Driver inserts text to describe the bookmark level, such as Form Set, Group, Form, or Page, and the index for that level. Here is an example:



Creating Custom Bookmarks

You can use custom rules that you create to make custom bookmarks in the PDF file. With a custom rule, you can use the *extra info* in the FAP objects to store custom bookmark titles. Currently, the system uses *extra 1* and *extra 2*, leaving *extra 3* for bookmark titles.

If you choose to develop a custom rule to use *extra 3* for this purpose, keep these things in mind:

- The setting for the Bookmark option remains the same.
- The maximum length for a custom bookmark title is 128 characters.
- This feature is not a callback, so all bookmark settings using *extra 3* must be finished before you send them to the PDF Print Driver.

- The PDF Print Driver expects to receive a character string for *extra 3* and the first eight characters must be *BOOKMARK*. The actual bookmark text begins with the ninth character and is NULL terminated.
- If you do not set *extra 3* (NULL handle) or the first eight characters are not *BOOKMARK*, the PDF Print Driver ignores *extra 3* and uses its original logic to create bookmarks.

Note Refer to the following discussion of the `FAPGetExtraInfo` and `FAPPutExtraInfo` functions for information on getting and setting *extra3*.

Here's how the system determines the text for a bookmark in a form set:

If you are filtering by recipient, the system...

1. Checks *extra3* of the recipient (128 character limit).
2. Checks *extra3* of the form set (128 character limit).
3. Check the recipient name (15 character limit).

If you are not filtering by recipient, the system...

1. Checks *extra3* of the form set (128 character limit).
2. Checks the form set name, which cannot exceed eight characters.

For group level bookmarks, the PDF Print Driver first checks the *extra3* of the group. If there is no information in *extra3*, it tries to use a string with this format:

GROUPNAME1, GROUPNAME2, GROUPNAME3

For form level bookmarks, the PDF Print Driver first checks the form's *extra3*. If there is no information in *extra3*, the driver uses the form description. If there is no form description, it uses the form name.

For page level bookmarks, the PDF Print Driver checks *extra3*. If there is no information in *extra3*, the driver uses information from the page name, unless the page name is...

- Of the format "Page #"
- or
- The same as the name of the first image on the page

FAPGetExtraInfo

Use this function to get the *extra* information for the given object.

```
VMMHANDLE FAPGetExtraInfo(VMMHANDLE descH,  
                           unsigned short whichOne,  
                           void* rec,  
                           size_t size)
```

Parameter	Description
-----------	-------------

descH	Handle of an input object.
-------	----------------------------

Parameter	Description
whichOne	The extra information you want, such as: <ul style="list-style-type: none"> • FAPEXTRA_ONE • FAPEXTRA_TWO • FAPEXTRA_THREE • FAPEXTRA_ATTRS
rec	A pointer to a buffer.
size	The size of the buffer.

This function returns the handle of the extra information unless it is a null handle. If it is a null handle, the function returns VMMNULLHANDLE.

Files

Include: fapdef.h

Source: FAPBASE.C

FAPPutExtraInfo

Use this function to put the extra information for the given object.

```
VMMHANDLE FAPPutExtraInfo(VMMHANDLE descH,
                          unsigned short whichOne,
                          void* rec,
                          size_t size)
```

Parameter	Description
descH	Handle of an input object.
whichOne	The extra information you want, such as: <ul style="list-style-type: none"> • FAPEXTRA_ONE • FAPEXTRA_TWO • FAPEXTRA_THREE • FAPEXTRA_ATTRS
rec	A pointer to a buffer.
size	The size of the buffer.

This function restores the extra information for the given object using the information in the input buffer. It then returns the handle of the extra information unless it is a null handle. If it is a null handle, the function returns VMMNULLHANDLE.

Files

Include: fapdef.h

Source: FAPBASE.C

ADDING HYPERTEXT LINKS

Using Documaker Studio, you can create forms which, when output from the PDF Print Driver, contains hypertext links. These links can, for instance, launch a web browser and open a specified web site. You can create hypertext links in these types of objects in Studio:

- Text labels
- Variable fields
- Graphics (LOG files)

For more information, see the Documaker Studio User Guide.

ADDING DIGITAL SIGNATURE PLACEHOLDERS

On June 30, 2000, the Electronic Signatures in Global and National Commerce (E-SIGN) Act was signed into law. This law provides for digital signatures to carry the same weight as their written counterparts.

Adobe's PDF format lets you add digital signatures to documents. Some vendors and companies are using PDF-based software solutions as a way of implementing electronic document signatures.

Oracle Insurance added support for placing empty signature fields in PDF documents to its Documaker line of products beginning with version 11.3. In Documaker software, these empty signature fields are called *PDF signature placeholders*.

When you open a PDF file that contains a signature placeholder in Adobe Acrobat, the signature field appears as shown below. When you click on the signature field, Acrobat displays a window that lets you sign the document electronically.

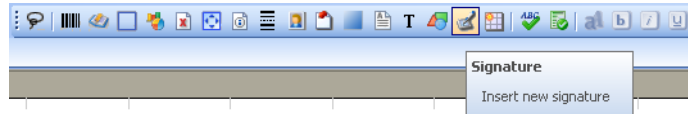


While Documaker and the PDF standard allow for any number of signatures to be placed in a document, the PDF standard does not provide for signing only sections of a document. That means the digital signatures will apply to the entire document.

Note Other, third-party signing tools, typically sold as plug-in extensions to Acrobat, may support multiple signatures in a document. The Documaker solution can be used with third-party signing tools, but Documaker itself does not manage the signing process for this type of signature.

Inserting a signature placeholder

To insert a signature placeholder into a section, first open the Studio's Section manager. Then select Signature from the Insert menu or click the Signature icon on the toolbar.



Your cursor changes to look like the one shown below. Use the cursor to draw a rectangle where you want the signature object to be located.



After drawing the signature, you can choose from these options for the signature object:

Option	Description
Name	(Optional) Enter up to 64 characters as a name for the signature object.
Type	The type of the signature object. The only type currently supported is PDF Placeholder, which is an empty signature field placed in PDF files.
Font ID	Specify the font with which the signed field will be displayed by Acrobat.
Color	Specify the color in which the signature text will be displayed by Acrobat. Make sure the Print in Color option has been checked.

No additional options are required for the PDF driver. The PDF driver automatically inserts the signature placeholder into the PDF file.

Note This feature is supported by the PDF driver on all supported platforms. Signature objects are ignored by print drivers other than PDF.

FORCING THE PDF DRIVER TO PRINT COLOR IMAGES

You can use the ForceColorBitmaps option to make sure the PDF Print Driver prints images in color, as shown here:

```
< PrtType:PDF >
    ForceColorBitmaps = Yes
```

Option	Description
ForceColorBitmaps	Enter Yes if you want the PDF Driver to print images in color even if the images are not set to print in color (the image does have to be a color image, of course). The default is No.

Note Before version 11.1, the PDF Print Driver always printed graphics (LOG files) in color, regardless of how the Print in Color option is set for the graphic. In version 11.1 this was changed so the PDF Print Driver honors the Print in Color setting for graphic.

So you can continue to produce PDF files with color graphics without having to update your FAP files to turn on the Print in Color option for graphics, this INI option was added. Essentially, this option lets you produce PDF files just as before, without having to check or reset the Print in Color option.

MEETING THE PDF FOR ARCHIVE SPECIFICATION

The PDF Print Driver complies with the requirements of the PDF for Archive (PDF/A) specification. This specification was designed by the International Standards Organization (ISO) and is intended to facilitate the long-term storage of electronic documents. The specification (ISO 19005-1) outlines the restrictions to which conforming files must adhere.

There are two sets of requirements. The PDF/A-1a set of requirements is more rigorous than the PDF/A-1b set. The Oracle Insurance PDF Print Driver implements PDF/A-1b compliance.

To set up your system to generate PDF/A-1b compliant documents, add the following INI option:

```
< PrtType:PDF >
    PDFAOptions = PDFA
```

Option	Description
PDFAOptions	Include this option if you want to produce PDF/A-1b compliant PDF files. Your entry identifies the INI control group that contains PDF/A-specific options. The default is PDFA.

If you use the default, the system automatically assumes the PDFA control group has the following options and values, so you do not have to actually include it — unless you want to use different values:

```
< PDFA >
    ICCProfile      = srgb.icc
    OutputCondition = Generic CRT Monitor
    OutputConditionID= sRGB IEC61966-2.1
```

Option	Description
ICCProfile	This specifies the name of the International Color Consortium (ICC) profile. This file describes the color attributes of a device or viewing requirement by defining a mapping between the source or target color space and a profile connection space (PCS). The file you specify must reside in the directory defined by the DefLib option in the MasterResource control group. The default is srg.icc. This file is embedded in the source code and will be used unless you include this control group and option and enter a different value in this field.
OutputCondition	A text string that describes the intended output device or production condition in human-readable form.
OutputConditionID	A text string identifier for the output condition.

Note The PDFAOptions option tells the system you want to produce PDF/A-1b compliant PDF files. The system looks at the value you enter for that option and tries to find a control group by that name. If it cannot find a control group that matches, it uses the default PDF/A values.

Keep in mind...

- The PDF file header must begin at byte offset zero (0) in the file. This is not a concern unless DAL scripts are used to place comments at the start of the file. If you enable PDF/A in a system that does emit comments at the beginning of the PDF file, the PDF driver will issue a warning but will still emit the comments.
- The document cannot be encrypted. If encryption and PDF/A support are both enabled, a warning appears and the document will not be encrypted.
- Colors must be specified in a device-independent manner. Oracle Insurance applications support the RGB scheme of specifying colors, which is device-dependent.

To implement device-independence, the PDF driver must embed an ICC profile in the file. ICC profiles are widely available for an extensive array of devices. There may be licensing requirements if color profiles are to be embedded in PDF files. Adobe offers a royalty-free license for embedding the color profiles located at this web site:

www.adobe.com/digitalimag/adobergb.html

The embedded color profile must be an RGB-based profile. The Oracle Insurance PDF Print Driver defaults to using a file named SRGB.ICC, which is a generic color profile with a wide range of colors (gamut).

Note ICC profiles vary in size from a few hundred bytes to a megabyte or more. If the system is configured to produce compressed PDF files, these are compressed as well. Be aware, however, that they add to the size of the file.

- To be PDF/A compliant, all fonts, TrueType or Postscript, must be embedded into the PDF file. This means the font files for every font used must be available on disk. Furthermore, you must have the legal right to embed all fonts used in a document. It is up to you to handle any font licensing issues that may arise.

Keep in mind that embedding fonts increases the size of the PDF file. Enabling font compression in the PDF Print Driver will minimize this to an extent.

- The document catalog must contain a metadata key referencing a valid XMP metadata stream. The metadata stream must contain 2-4 KB of padding at the end to allow for in-place updating by applications that may not be PDF-aware. The Oracle Insurance driver uses 3KB. Additionally, the metadata stream must be unfiltered — specifically, uncompressed. Therefore, metadata will increase the size of PDF files by a minimum of 3kB.

Note Metadata is generated by the driver and is not configurable.

Important Notice

1. Oracle Insurance applications may include an ICC Profile for use in producing PDF/A-compliant output. Oracle Insurance obtained the ICC Profile from Adobe Systems Incorporated ("Adobe"), but this ICC Profile and others can also be obtained directly from Adobe's Support Web site and then used with Oracle Insurance applications.
2. Oracle Insurance has not modified the Adobe ICC Profile, however, Customer acknowledges that (a) Adobe disclaims all warranties and conditions, express or implied; (b) Adobe excludes any and all liability for damages related to the use of the ICC Profile as provided by Oracle Insurance; and (c) the terms and conditions with respect to the license granted herein for the Adobe ICC Profile are offered by Oracle Insurance alone and not by Adobe. The ICC Profile is Copyrighted 2006 by Adobe Systems Incorporated.

EMULATING DUPLEX PRINTING FROM THE PDF PRINT DRIVER

You can tell the PDF Print Driver to add blank pages so it will produce output that emulates how a form set that contains both simplex and duplex forms would print on a duplex printer. You can print this PDF file from Adobe Acrobat to a Windows print driver which has duplex printing enabled. The printed document will appear to have been printed on a duplex printer. In addition, you can tell the PDF Print Driver to add FAP files onto the pages it adds.

Note You must have a printer capable of duplex printing.

Here is an example of how you would set up the INI options in your MRL INI file for the PDF Print Driver:

```
< Printer >
  PrtType           = PDF
< PrtType:PDF >
  BlankPageImage    = LeftBlank
  BookMark          = Yes, Page
  Device            = data\pdfout.pdf
  DownloadFonts     = Yes, Enabled
  EmulateDuplexPrinter= Yes
  ForceColorBitmaps = Yes
  LanguageLevel     = Level1
  Module            = PDFW32
  PageNumbers       = Yes
  PrintFunc         = PDFPrint
  SendColor         = Yes, Enabled
```

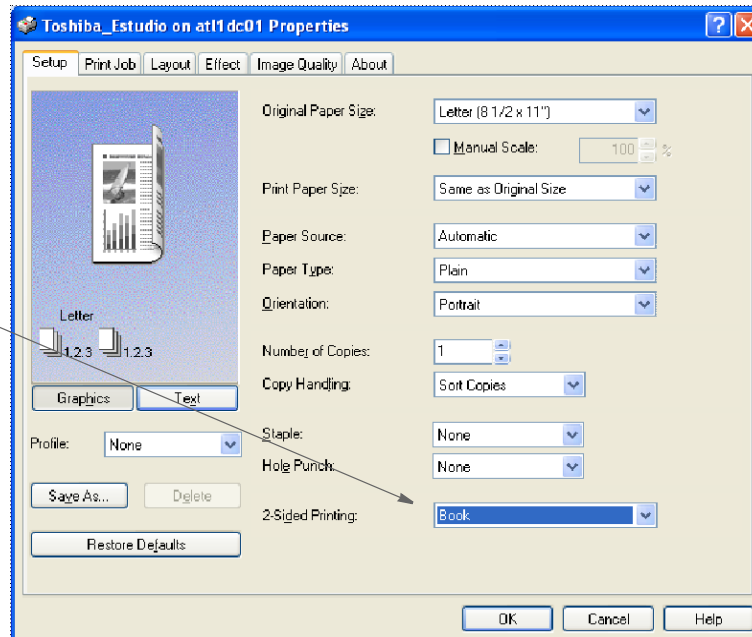
The options for this feature are discussed here:

Option	Description
BlankPageImage	(Optional) Enter the name of the section (FAP file) you want the PDF Print Driver to insert when it needs to insert a page to emulate duplex printing. For instance, you could use this to have the PDF Print Driver insert a FAP file named <i>LeftBlank</i> which contained a text label that said: This page intentionally left blank.
EmulateDuplexPrinter	(Optional) Enter Yes if you want the PDF Print Driver to emulate a duplex printer. This means the driver will insert pages as necessary in the form set. The pages are blank by default, but you can use the BlankPageImage option to specify a FAP file to insert. The default is No.

Note Do not use this feature with the AddBlankPages DAL function or the DPRAddBlankPages Documaker Bridge rule. This feature is used instead of these rules.

You can print the PDF file the print driver produces from Acrobat to a Windows print driver (printer) with duplex printing enabled. The printed document will appear to contain the same mixture of simplex and duplex forms. When you print the PDF file through Adobe, be sure to turn on 2-sided printing. This option will appear on the Properties tab of the Print window for your printer. Here is an example:

Be sure to turn on 2-sided printing



In this example the options for 2-sided printing include none, book (long binding), or tablet (short binding). These options will vary depending on your printer. Check your printer manual for more information.

Keep in mind, the PDF Print Driver...

- Does not rotate pages (turn upside down) that would have printed on the back of a short bind duplex print job. PDF files are meant to be viewed.
- Does not support a mixture of long and short bind duplex for the PDF files with the blank back pages that are added. If the entire print job is a mixture of simplex and long bind duplex, you can select the long bind duplex setting for your Windows print driver. If the entire print job is a mixture of simplex and short bind duplex, you can select the short bind duplex setting for your Windows print driver.

INTERFACING WITH IMAGING SYSTEMS

The PDF Print Driver can add free form text or data at the beginning of a batch or each form set within the batch. This can help you interface with imaging systems such as RightFax.

Use the TEXTScript option to specify the DAL script you want to run. This DAL script creates a free form data or text buffer and adds it to the print stream.

Here is an example of the DAL script:

```
* Populate the PDF stream comment with these values from RCBDFFD
faxnum = trim(GVM('FaxNumber'))
faxname = trim(GVM('FaxName'))
```

```
AddComment('<TOFAXNUM:' & faxnum & '>',1)
AddComment('<TONAME:' & faxname & '>',1)
```

```
Return
```

Notice the use of the second parameter to the AddComment DAL function. The 1 indicates the string should be an ASCII string. If you omit this parameter, the system converts the string into an EBCDIC string. You can also use the TEXTCommentOn option to tell the system to add free form text or data to the beginning of every form set or print batch.

Here is an example:

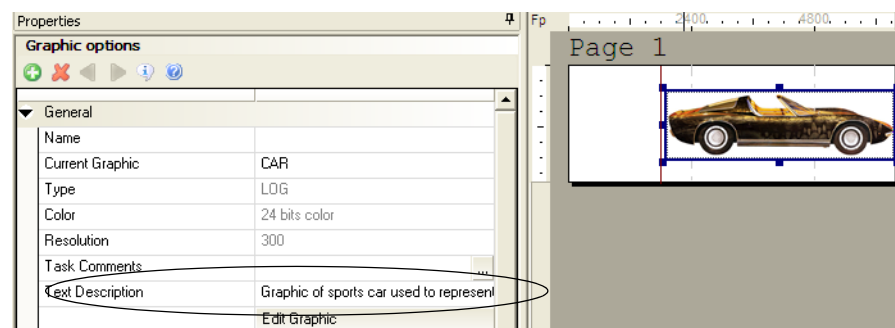
```
< PrtType:PDF >
  TEXTScript      = imaging.DAL
  TEXTCommentOn = formset
```

CREATING SECTION 508-COMPLIANT PDF FILES

You can create PDF files that are compliant with Section 508 of the Rehabilitation Act of 1973, as amended (29 U.S.C. 794d). Section 508 requires federal agencies to give disabled employees and members of the public access to information that is comparable to the access available to those without disabilities. Documaker lets you create PDF files that are as accessible to persons with disabilities as they are to those without disabilities. To enable Section 508 compliance, set the Section508 option to Yes, as shown here:

```
< PrtType:PDF >
  Section508 = Yes
```

To create content that complies with Section 508, you must add text equivalents for all non-text objects. To provide text equivalents, use the Text Description field.



You can enter up to 100 characters. The PDF Print Driver adds this description when it creates the PDF.

Also make sure any information conveyed with color is also available without color.

Note It is also generally accepted that Section 508 compliance implies the ability of assisted technology to determine the reading order of content.

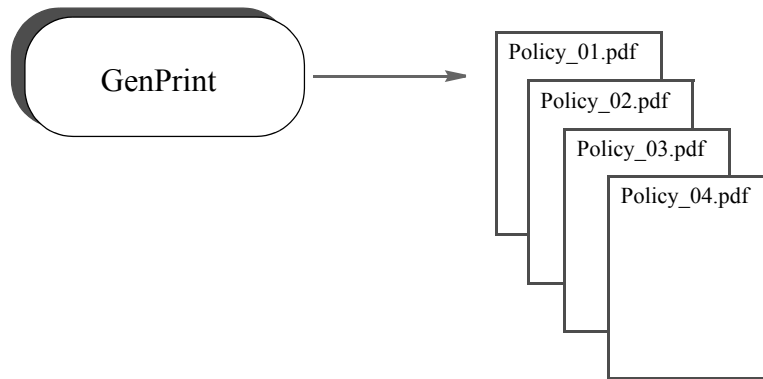
EXAMPLES

Here are some examples of how you can use the PDF Print Driver.

- *Using the KeyID field to create one PDF file per transaction on page 203*
- *Naming recipient print streams in using single-step mode on page 204*

Using the KeyID field to create one PDF file per transaction

This scenario shows how to set up your system so it will produce a single PDF file for each transaction and the PDF file will be named using the KeyID field and the PDF extension, such as: *policynumber.pdf*.



This scenario assumes:

- You are running Documaker Server in multiple step mode, with separate the GenTrn, GenData, and GenPrint programs executing separately
- The KeyID field (policy number) is unique throughout the entire batch run
- You have one recipient per transaction

To accomplish this, set up your system as described here:

1. Create a DAL library that contains the function to use to create the unique file name. Then copy the DAL library into DEFLIB. Assume the library illustrated below is named *SETPDFNM.DAL*.

Here is a sample DAL library that contains two DAL scripts, one to illustrate a simple way to accomplish this, and a more complex example that can handle more possible name collisions.

```
BEGINSUB SIMPLE_NAME
* This example presumes:
*   One occurrence per run of a given policy number
*   Only one recipient per transaction
*
  IF HAVEGVM("PolicyNum")
    RETURN("data\" & GVM("PolicyNum") & ".pdf")
  END
  COUNTER += 1
  RETURN("data\Emptyfile" & COUNTER & ".pdf")
ENDSUB
```

```
BEGINSUB COMPLEX_NAME
* This example is more complex and assumes
```

```
*   many possible factors should be considered
*   to prevent a name collision
*
COUNTER += 1
IF HAVEGVM("Company")
    FILENAME = GVM("Company") & \
               GVM("Lob") & \
               GVM("PolicyNum") & \
               GVM("TransactionType") & \
               GVM("RCBRcpCode") & \
               GVM("RunDate") & \
               COUNTER
ELSE
    FILENAME = "emptyfile" & COUNTER
END
RETURN("data\" & FILENAME & ".pdf")
ENDSUB
```

2. Enable the DAL library to be loaded. In your FSISYS.INI file, add the following option:

```
< DALLibraries >
    Lib = SETPDFNM
```

3. Enable the multifile print callback function and log file. In your FSISYS.INI file, include options similar to these:

```
< Print >
    CallbackFunc = CUSMultiFilePrint
    MultiFileLog = data\pdflog.txt
```

4. Make sure PDF print is enabled. In your FSISYS.INI file, include options similar to these:

```
< PrtType:PDF >
    Module = PDFW32
    PrintFunc = PDFPrint
< Printer >
    PrtType = PDF
```

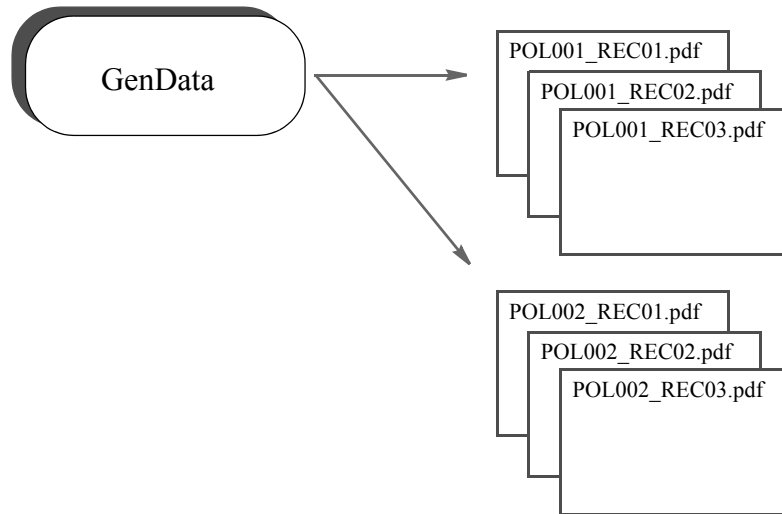
5. Make the Port option call the DAL function to get the name. In your FSISYS.INI file, change all of the Port options to something like this:

```
< Printer1 >
    Port = ~DALRUN SIMPLE_NAME
< Printer2 >
    Port = ~DALRUN SIMPLE_NAME
< Printer3 >
    Port = ~DALRUN SIMPLE_NAME
< Printer4 >
    Port = ~DALRUN SIMPLE_NAME
< Printer5 >
    Port = ~DALRUN SIMPLE_NAME
and so on ...
```

When the GenPrint program runs, it will create a name using the KeyID (policy number) field.

Naming recipient print streams in using single-step mode

This scenario shows how to set up a master resource library (MRL) so it will produce a unique file name for each recipient of each transaction when running Documaker Server in single-step mode.



This scenario assumes:

- You have multiple recipients per transaction.
- You are using the XDB for its token lookup ability. (Using XDB is not required, it was just used in this example.)

To accomplish this, set up your MRL as described here:

1. Create a DAL library file in your MRL DefLib directory that contains the user-defined function to create unique file names. Assume the library file illustrated below is named: *unique_print_names.dal*.

This file contains two DAL user-defined functions; one illustrates using an XML file extract file, and a more complex example that can handle possible name collisions using a standard extract file. The DAL scripts are not extract specific.

- * The **xml_prt_names** script assumes:
- * Three unique global variables are defined in the RCBDFDFL.DAT file.
- * The XDB has an entry called `policy_number`, whose xpath points to the policy number and each policy number is unique.
- * Only three recipients exist.
- * The PrintFormSet control group has the correct INI options defined.

```

BeginSub xml_prt_names
    prt_name = Lower(GetINIStrng(, "Printer", "PrtType"))
    pol_num1 = Trim(?("policy_number")) & ".insured." & prt_name
    pol_num2 = Trim(?("policy_number")) & ".agent." & prt_name
    pol_num3 = Trim(?("policy_number")) & ".company." & prt_name
    SetGVM("PrtName001", pol_num1, "C", 25)
    SetGVM("PrtName002", pol_num2, "C", 25)
    SetGVM("PrtName003", pol_num3, "C", 25)
EndSub

```

- * The **std_prt_names** script assumes:
- * Three unique global variables are defined in the RCBDFDFL.DAT file.
- * Only three recipients exist.
- * The PrintFormSet control group has the correct INI options defined.

```

BeginSub std_prt_names
    cnt += 1
    prt_name = Lower(GetINIStrng(, "Printer", "PrtType"))
    pol_num1 = "insured." & #cnt & prt_name
    pol_num2 = "agent." & #cnt & prt_name
    pol_num3 = "company." & #cnt & prt_name

```

```
SetGVM("PrtName001" ,pol_num1 ,,"C",25)
SetGVM("PrtName002" ,pol_num2 ,,"C",25)
SetGVM("PrtName003" ,pol_num3 ,,"C",25)
EndSub
```

2. Make sure your FSISYS.INI or FSIUSER INI file has the following control groups and options defined. These examples assume you want to create PDF output.

Enable the DAL sub-routine library to be loaded:

```
< DALibraries >
  Lib          = unique_print_names.dal
```

Enable the multiple file print function:

```
< PrintFormSet >
  MultiFilePrint= Yes
  LogFile       = .\data\pdflog.dat
  RCBDFDField   = PrtName
```

Make sure PostScript print is enabled:

```
< Printer >
  PrtType       = PDF
< PrtType:PDF >
  Module        = PDFW32
  PrintFunc     = PDFPrint
```

Define the Port options as follows:

```
< Insured >
  Printer       = Insured
  Port          = .\print\.pdf
< Agent >
  Printer       = Agent
  Port          = .\print\.pdf
< Company >
  Printer       = Company
  Port          = .\print\.pdf
```

3. Make sure the RCBDFDFL.DFD file has the following global variables defined.

```

FieldName = PRTName001
FieldName = PRTName002
FieldName = PRTName003

< Field:PRTName001 >
  Int_Type    = Char_Array
  Int_Length  = 26
  Ext_Type    = Char_Array_No_Null_Term
  Ext_Length  = 25
  Key         = N
  Required    = N
< Field:PRTName002 >
  Int_Type    = Char_Array
  Int_Length  = 26
  Ext_Type    = Char_Array_No_Null_Term
  Ext_Length  = 25
  Key         = N
  Required    = N
< Field:PRTName003 >
  Int_Type    = Char_Array
  Int_Length  = 26
  Ext_Type    = Char_Array_No_Null_Term
  Ext_Length  = 25
  Key         = N
  Required    = N

```

4. Include the following rule in your AFGJOB.DAT file to call the DAL user-defined function before each transaction is executed.

```

/* Every form set in this base uses these rules. */
<Base Form Set Rules>
;NoGenTrnTransactionProc;;;
;PrintFormset;;;
;UseXMLExtract;;;
;ResetOvFlw;;;
;BuildFormList;;;
;PreTransDAL;xml_prt_names();

```

Keep in mind you could also use this PreTransDAL rule:

```
;PreTransDAL;;std_prt_names();
```

If you want to use the *std_prt_names* DAL script.

When the GenData program runs, it creates print output files as follows. Assume the policy numbers are: *MVF 01-12-03* and *GRA 06-22-03*.

- When using the *xml_prt_names* DAL library function:

```

MVF 01-12-03.insured.pdf
MVF 01-12-03.company.pdf
GRA 06-22-03.company.pdf
...

```

- When using the *std_prt_names* DAL library function:

```

Agent.1.pdf
Insured.1.pdf
Company.1.pdf
Agent.2.pdf
...

```

WORKING WITH FONTS

The fonts you use determine how the text on the page looks. With PDF files, you can choose to simply use the base fonts distributed with Acrobat Reader or you can embed the actual fonts used into the PDF file. The latter approach makes sure the document represented by the PDF file looks just like the original.

Embedding fonts also makes for larger PDF files, so if file size is a greater consideration than fidelity, you may want to choose not to embed fonts or to design the document with fonts similar to those distributed with Acrobat Reader.

This topic discusses...

- *Using the Base Fonts* on page 208
- *Embedding Fonts* on page 210
- *Subsetting Fonts* on page 214
- *Handling Fonts with Multiple Width Tables* on page 214
- *Using Font Cross Reference Files* on page 215

USING THE BASE FONTS

Adobe includes the following fonts with Acrobat Reader. You do not have to embed these fonts in PDF files.

Fixed Pitch Fonts	Proportional Fonts
Courier	Helvetica
Courier-Bold	Helvetica-Bold
Courier-Oblique	Helvetica-Oblique
Courier-BoldOblique	Helvetica-BoldOblique
	Times-Roman
	Times-Bold
	Times-Italic
	Times-BoldItalic
	Symbol
	ZapfDingbats

USING OPENTYPE® FONTS

There are two types of OpenType fonts:

- OpenType/TTF fonts containing TrueType outlines, usually with a .TTF file extension.
- OpenType/CFF fonts containing Postscript outlines, usually with a .OTF file extension.

Currently, only OpenType/TTF fonts containing TrueType outlines (TTF) are supported.

To use OpenType fonts in Documaker, first import the font into your font cross-reference (FXR) file and perform any related tasks, depending on how you will use the font. These tasks can include the following:

To	See
Adding fonts to the FXR file	The Documaker Studio User Guide . The FXR files included with the system do not contain OpenType® fonts.
Embedding fonts	<i>Embedding Fonts</i> on page 210.
Installing the font in Windows Font Control panel	Your operating system documentation

Note The PDF Print Driver does not support font subsetting for OpenType® fonts.

EMBEDDING FONTS

The PDF Print Driver lets you embed fonts into the PDF print stream. This topic discusses when to embed fonts and when not to. It also describes how the system determines which base font to use or which custom font to embed.

Generally, you want the PDF Print Driver to reproduce your document so that it looks exactly as it did when you created it. Embedding fonts lets you accomplish this if you are using fonts that do not match the criteria discussed below.

When not to embed fonts

If you do not need to reproduce the exact look of the original document, you do not need to embed fonts. For example, the base system FXR file is not set up to embed fonts. The fonts in the base system FXR are remapped to the names of the 14 base fonts Adobe supports with Acrobat Reader.

Note For a list of the base fonts, see *Using the Base Fonts* on page 208.

The Andale Duospace (fixed pitch, sans-serif) font is mapped to the base Adobe Courier (fixed pitch, serif) font. If you prefer the sans-serif look of Andale Duospace, you should embed that font.

Adobe also uses a standard scaling algorithm. If your implementation uses fonts that scale exactly as Adobe expects, you do not need to embed fonts. The PDF Print Driver determines the fonts to use and scales them for you. See *Not Embedding Fonts* on page 211 for more information.

In summary, you do not need to embed fonts if the fonts you are using ...

- Are already scalable
- Closely match the PDF base fonts

When to embed fonts

Embedding fonts lets you control the appearance of the document by letting you specify which fonts Acrobat Reader should use and what the font width will be. When you need to reproduce the exact look of the original document and you use custom fonts that do not scale exactly as Adobe expects, you should embed fonts.

To embed fonts, you need a set of PostScript Type 1 fonts or TrueType fonts, and you need to set the DownloadFonts INI option to *Yes*. You also need to run the FXRVALID utility to prepare your FXR file. For detailed instructions, see *Using Embedded Fonts* on page 212.

Not Embedding Fonts

If you are not going to embed fonts, you must set the following INI option to No, as shown here:

```
DownloadFonts = No
```

When you use a font that is not included in the 14 base fonts distributed with Acrobat Reader, the PDF Print Driver uses the information in the following fields, in this order, to determine what to do with the font:

- The PDF Print Driver checks to see if the beginning of the setup data matches one of the system font names. If the text matches one of the fonts, the print driver checks the font attributes for weight and style (italic or not), and uses that information to complete the mapping.
- The PDF Print Driver checks the Setup Data field under PostScript Properties.
- The PDF Print Driver checks the TypeFace field under Screen Properties. The system checks this field to see if its contents matches one of the 14 base Adobe fonts or an equivalent system font name.

Note For a list of the base fonts, see *Using the Base Fonts* on page 208.

When the system matches a criteria, it then stops. If, after checking these fields, the system does not find information that matches one of the 14 base Adobe fonts or an equivalent system font, it then maps...

- Proportional fonts to the Adobe Helvetica font (normal, bold, italic, or bolditalic)
- Fixed pitch or non-proportional fonts to the Adobe Courier font (normal, bold, italic, or bolditalic)

The system then checks these fields to determine additional font attributes:

- Spacing (fixed pitch or proportional)
- Style (italic or upright)
- Stroke Weight (bold or normal)

Using Embedded Fonts

If you are going to embed fonts, you must have either PostScript Type 1 or TrueType fonts. In addition, you must set the following INI option to Yes, as shown here:

```
DownloadFonts = Yes
```

You must also have the following information set up correctly for the PDF Print Driver to embed the font. If there is an error embedding a font, the PDF file is not created.

- The Embed Font field under PDF Properties is set to Yes if the field should be embedded or No if it should not be embedded.
- The Font Name field under PDF Properties must be defined. If PDF Print Driver cannot find the file name here, it checks the Font Name field under PostScript Properties. (Postscript => .PFB TrueType => .TTF). This field contains the file name of the PostScript or TrueType font you want to embed. This file should exist in the directory specified by the FontLib setting in your master resource library.

Note The information stored in the A,R3 OTH record in the FXR appears in the font property fields in Studio's Font manager. You can edit the information there. The FXRVALID utility can also create this record. For more information about the FXRVALID utility, see the Utilities Reference.

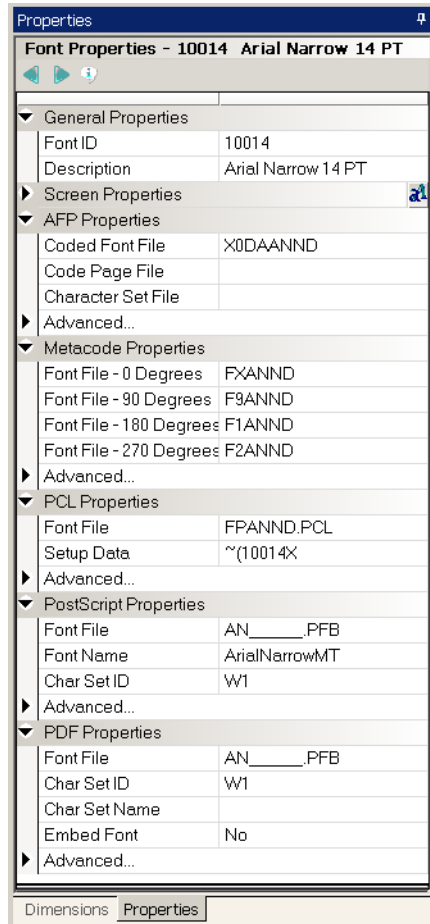
Embedding Bitmap Fonts

You can embed bitmap fonts in AFP, PCL, and Xerox Metacode (FNT) format. To embed fonts in these formats, add the FontSearchOrder option to your PDF printer control group. Here is an example:

```
< PrtType:PDF >  
  FontSearchOrder =
```

Option	Description
FontSearchOrder	List the printer types in the order in which you want them searched. You can include these printer types: AFP – AFP printers XMC - Xerox Metacode printers PCL – PCL printers Separate the printer types with commas. Here is an example: <pre>FontSearchOrder = AFP,XMC,PCL</pre>

These printer types are defined on the Font Properties tab in Studio:



If the PDF Print Driver cannot locate a scalable font file (Adobe Type 1 or TrueType) in the PDF or PostScript properties, it checks the other font properties, in the order you specified in the FontSearchOrder option. For instance, assume you set the option as shown here:

```
< PrtType:PDF >
  FontSearchOrder = PCL,XMC
```

In this example, after searching the PDF and PostScript attributes, the PDF Print Driver would see if the PCL attributes of the font contain a font file name. If they do and that file exists, the PDF Print Driver embeds that font. If not, it looks at the Xerox Metacode (XMC) attributes to see if they contain a font file name and if that file exists.

If it cannot find a font it can embed, the PDF Print Driver generates an error message and stops processing the transaction.

Note If additional font attribute types exist but are not listed in the FontSearchOrder option, the PDF Print Driver ignores those attributes.

When it checks for the existence of font files, the PDF Print Driver uses the FontLib option of the MasterResource control group. For all but Xerox Metacode fonts, the PDF Print Driver uses the file name as provided in the corresponding attributes. For Metacode fonts, it first tries to use the file name as provided. If that file does not exist and the file name as provided has no extension, the PDF Print Driver appends the extension .FNT to the file name and looks for that file.

Note There are no extensions on z/OS, so for that environment the PDF Print Driver ignores the extension when checking for the font file.

Scaling Embedded Fonts

When using Acrobat's built-in fonts, the PDF Print Driver automatically scales text to match the height and width defined by the text label's dimensions. Occasionally, this scaling can also improve the fidelity of embedded fonts.

To tell the system to scale embedded fonts, add the AdjustTextWidth option to the PDF printer group. Here is an example:

```
< PrtType:PDF >  
  AdjustTextWidth = Yes
```

Option	Description
AdjustTextWidth	Enter Yes to tell the system to scale embedded fonts to match the height and width of the text label. In some cases, this scaling can improve the fidelity of embedded fonts. The default is No. Acrobat's built-in fonts are automatically scaled.

SUBSETTING FONTS

By default, the PDF Print Driver embeds only those glyphs that appear in the document when it embeds fonts. The glyphs are typographical symbols, such as letters of the alphabet and punctuation symbols. Embedding only the glyphs used in the document results in smaller PDF files.

You can, however, tell the PDF Print Driver to include all of the glyphs — not just those used in the document — by adding the SubsetAllEmbeddedFonts option to your PDF printer control group and setting it to No, as shown in this example:

```
< PrtType:PDF >  
  SubsetAllEmbeddedFonts = No
```

Note The PDF Print Driver does not support font subsetting for OpenType® fonts.

HANDLING FONTS WITH MULTIPLE WIDTH TABLES

When embedding resources for multiple fonts, the PDF Print Driver must decide if the fonts are related and whether the various point sizes of related fonts are scalable. If the fonts are not embedded, The PDF Print Driver compares the base Adobe font names being used. If the fonts are embedded, the PDF Print Driver must consider the source of the font.

If an FXR entry is created from a bitmap font — anything except files with one of the extensions *TTF* or *PFB* — the driver assumes the font is not scalable with respect to any other font and it will get its own width table. Furthermore, when it generates PDF/A-compliant output, the width table must match the widths embedded in the font program. If there are multiple width tables, the PDF/A standard requires that there also be multiple embedded font programs.

When importing TrueType or PostScript Type 1 font files, the PDF Print Driver recognizes that all FXR entries created from a given file are scalable and only embeds a single width table.

To optimize PDF file size, it is a good idea to create forms using FXR entries generated from TrueType or PostScript fonts.

Note Because the PDF Print Driver can discriminate between scalable and bitmap fonts as well as scale text both horizontally and vertically, the SplitText option is no longer necessary or supported. Also, the Font Index field is no longer needed and is ignored.

USING FONT CROSS REFERENCE FILES

When not embedding fonts, the quality of the PDF files you create is in large part influenced by the setup information contained in the font cross-reference (FXR) file. Keep the following tips in mind when looking at your FXR file to optimize the quality of your PDF output.

PostScript font names should be present in your FXR, and all font IDs should contain one of the following PostScript font names in the Setup Data field for PostScript printing. The names of the PostScript fonts are case sensitive.

- Courier (four versions)
- Helvetica (four versions)
- Times (four versions)
- Symbol
- ZapfDingbats

The point size value should be present and should be within 33% of the font height. Font heights are measured in 2400 dots per inch while point sizes are measured in 72 dots per inch, so some conversions to equivalent units will be necessary to determine their relative values.

Note All of the fonts listed above, except for the Univers fonts, are included in Adobe's Acrobat Reader and do not have to be embedded in PDF files.

The spacing value (either fixed or proportional) should be present and accurate. Here is a list of PostScript fonts sorted by spacing value.

Fixed pitch fonts	Proportional fonts
Courier	Helvetica
Courier-Bold	Helvetica-Bold
Courier-Oblique	Helvetica-Oblique
Courier-BoldOblique	Helvetica-BoldOblique
Courier-Italic	Times-Roman
Courier-BoldItalic	Times-Bold
	Times-Italic
	Times-BoldItalic
	Univers-Medium
	Univers-Bold
	Univers-MediumItalic
	Univers-BoldItalic
	Symbol
	ZapfDingbats

The font style value (upright or italic) should be present and accurate; it should match a PostScript font with an equivalent font style.

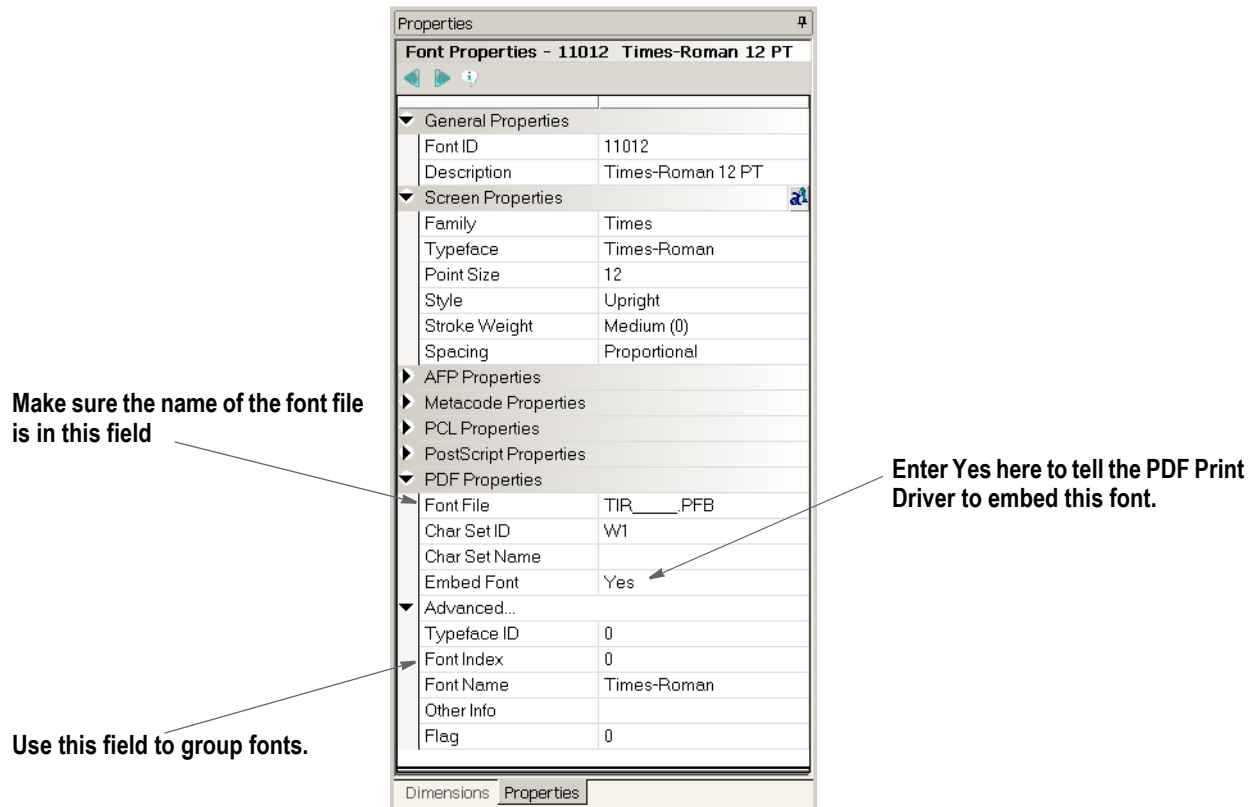
The font weight (bold or normal) should be present and accurate; it should match a PostScript font with an equivalent font weight.

How to embed fonts

Follow these instructions to embed fonts.

Note You can embed TrueType or PostScript fonts. The PDF Print Driver uses the font file extensions to distinguish between the two types of fonts. TrueType fonts have a *TTF* extension. PostScript fonts have a *PFB* extension.

1. Use a text editor to open the INI file. Set the DownloadFonts option in the PrtType:PDF control group to Yes.
2. Next, use Font manager to set up your font cross reference file (FXR). When you select Fonts in Studio the Font manager opens the font cross reference file associated with your workspace. Select the font you want to embed and click Edit.
3. On the Properties tab, make the following changes:



- Enter Yes in the Embed Font field to indicate that the font should be downloaded.
 - Use the Font Index field to group fonts.
 - Enter the name of the font file in the Font File field.
4. Set up the remaining fields as you would for a PostScript font. Keep in mind...
- If you set the DownloadFonts option to No, all fonts are mapped to one of the 14 Type 1 base fonts and no fonts are downloaded.
 - If you set the DownloadFonts option to Yes, the system downloads fonts used in the document as long as the Embed Fonts field is set to Yes.

If you set the Embed Fonts field to No, the system maps the fonts used in the document to one of the 14 Type 1 base fonts.

- If you do not define the fields on the Properties tab, the system will not download the font — even if you set DownloadFonts option to Yes. Instead, the system maps the font to one of the 14 Type 1 fonts.
- For symbol fonts, such as WingDings, make sure the Char Set ID field is set to *WD*.

Note Each font you embed increases the size of the PDF file. See *FontCompression* on page 169 for information on compressing fonts.

ADDING SECURITY TO PDF FILES

You can make your PDF documents more secure by encrypting them and by adding security settings. Secure PDF documents may be required for electronic bills, confidential documents (like medical records), and other documents containing sensitive material (bank statements, loan applications, and so on).

Security settings let you assign passwords for opening and modifying the document and control access to printing, editing, and annotating the document.

A document can have both an open password and an owner password and these passwords can consist of up to 32 characters. You can view a document with either type of password, but you must enter the owner password to change the document or its security settings.

If you use the security settings to restrict access to certain features, all toolbar and menu items related to those features are dimmed in Adobe Reader or Acrobat.

To enforce these restrictions, the content of the document is encrypted using a 40- or 128-bit algorithm specified by Adobe.

This topic provides information on the following:

- *Configuring the Security Features* on page 218
- *Using the PDFKEY Utility* on page 226
- *Using the PDFKEYGEN Function* on page 227
- *Using AES Encryption* on page 227
- *Example Security Settings* on page 228
- *Tips* on page 230

CONFIGURING THE SECURITY FEATURES

The PDF security features are built into the PDFLIB library so there are no files to install.

Configuring the INI Files

In each PrtType:XXX control group for which you want security features, add this INI option:

```
< PrtType:PDF >  
    Encrypt = Yes
```

This option tells the PDF library to encrypt PDF files. Encrypting the PDF file changes the file so it is no longer easy to read when transmitted over the network. It also means you cannot use a text editor to alter the file without destroying it.

Since, however, no passwords or permissions have been specified, this option provides only a minimal amount of security. If someone gets a copy of the file, there is no way to prevent that person from viewing the file in Acrobat Reader or altering it with the full Acrobat product.

You can, however, create more secure documents by including additional options in the PDF printer control group of your INI file.

In the same control group as the Encrypt option, you can add the SecurityGroup option to specify a control group which will contain the permissions, passwords, and encryption strength. Here is an example:

```
< PrtType:PDF >
    SecurityGroup = PDF_Encryption
    Encrypt       = Yes
< PDF_Encryption >
    .
    .
```

Note While the only way to specify passwords is through INI options, the PDFKEYGEN built-in function lets you create a custom built-in that supplies the actual passwords. This way, the only thing that the INI file contains is

```
OwnerKey = ~PDFKEYGEN ~CUSTOMPASSWORDRULE
```

Keep in mind the INI files are generally inside a firewall and the passwords are stored in encrypted form.

If you have a custom password rule, you can just specify:

```
Owner = ~CUSTOMPASSWORDRULE
```

Setting Up a Security Control Group

The SecurityGroup option specifies the control group where permissions, passwords, and encryption strength are set. The permissions let you control if users can...

- Print the document. (You can also control the print quality.)
- Modify the document.
- Copy text or graphics to the clipboard.
- Add or update annotations.
- Fill in form fields.
- Access the document with accessibility tools, such as text to speech applications.
- Add navigational aids to the document.

This table shows the INI options you can set. All AllowXXX options default to Yes, meaning the permission is granted. See also *Understanding permissions* on page 221.

Option	Description
AllowPrinting	Lets the user print the document.
AllowModify	Lets the user modify the document.
AllowCopy	Lets the user copy data to the clipboard.
AllowAnnotate	Lets the user add or update annotations.
AllowFormFields	Lets the user fill in form fields. You must set the KeyLength to 128 to use this option.

Option	Description
AllowAccessibility	Lets accessibility tools, such as text to speech applications, access the document. You must set the KeyLength to 128 to use this option.
AllowAssembly	Lets the user add navigational elements to the document. You must set the KeyLength to 128 to use this option.
AllowHighQualityPrinting	If you set AllowPrinting to Yes, set this option to Yes to let the user generate a high quality hard copy of the document. If you set AllowPrinting to Yes and this option to No, users can only print draft quality hard copy. You must set the KeyLength to 128 to use this option.
OwnerKey	Specifies the password required to change the document or its security settings. The password takes the form of an encrypted 64-byte hexadecimal encoded string. See <i>Choosing passwords</i> on page 220 for more information. For additional security, instead of specifying encrypted data in the INI file, you can use custom rules to provide the passwords directly. Here is an example: <pre>Owner = ~CUSTOMPASSWORDRULE User = ~CUSTOMERUSERPASSRULE</pre> For less security, you can use settings similar to these to specify plain text passwords: <pre>Owner = pdfowner User = pdfuser</pre>
UserKey	Specifies the password required to open the document. The password takes the form of an encrypted 64-byte hexadecimal encoded string. See <i>Choosing passwords</i> on page 220 for more information.
KeyLength	Specifies the encryption strength, in bits, either 40 or 128. The default is 128. The key length must match the length provided to the PDFKey tool. The choice of key length is a primary factor in determining how secure the PDF file will be. Computers are fast enough that 40-bit encryption is susceptible to attacks in which every possible encryption key is tried. The 128-bit encryption is much more secure and allows finer control of permissions, but PDF files encrypted with a key length of 128 bits can only be viewed using Adobe Acrobat and Acrobat Reader 5.0 or later. Because Acrobat Reader is available at no cost, this restriction is generally no more than a minor inconvenience.

Choosing passwords

An encrypted version of the open and owner passwords as well as permission information is kept in the security control group. These take the form of 64-byte hexadecimal encoded strings. Here is an example:

```
6d6f62d768bc143cefa30fc4fd3cc00eb1f638157f1d985dd5fe8ebfaa7c8317
```

There are two ways to generate these keys:

- The PDFKey tool generates these keys (see *Using the PDFKEY Utility* on page 226 for more information). Because the permissions are encoded in the keys, the permissions in the Security control group *must* match those provided to the PDFKey tool.
- You can also use the PDFKEYGEN built-in function to generate the OwnerKey and UserKey information. See *Using the PDFKEYGEN Function* on page 227 for more information.

You can set passwords any way you like and they can consist of up to 32 characters. Choosing the password is up to you. You can use special characters in passwords but, when using the PDFKey utility, you have to enclose the entire password in quotation marks (“”) just as you would if the password contains spaces. Here is an example:

```
pdfkw32 /U="~!@#$$%^&*()_+" /O="?<>{ } [] - / \ | "
```

This table describes the levels of security you get based on the passwords you set up:

Add this password

Owner	Open	To provide this level of security
Yes	Yes	You must enter the owner or open password to view the document. Users are bound by the assigned permissions You cannot change the security settings without the full Acrobat product and the owner password.
Yes	No	No password is required to view the document. Users are bound by the assigned permissions. You cannot change the security settings without the full Acrobat product and the owner password.
No	Yes	You must enter the open password to view the document Anyone with the full Acrobat product and the open password can view and change the PDF file, including the file's security settings.
No	No	No password is required to view the document. Users are bound by the assigned permissions. Anyone with the full Acrobat product can change the security settings — so it makes little sense to set passwords in this manner.

Note If you forget a password, there is no way to retrieve it from the document. Be sure to store passwords in a secure location in case you forget them.

Understanding permissions

This table provides a general overview of how permissions are related.

If this is set to Yes Attempts to set the following to No are ignored

AllowModify	AllowAnnotate, AllowFormFields, and AllowAssembly
AllowAnnotate	AllowFormFields and AllowAssembly
AllowCopy	AllowAccessibility
AllowFormFields	AllowAssembly

Note PDF files with security settings are called *secure* files. You cannot insert a secure PDF file into another PDF file. You can, however, insert a nonsecure PDF file into a secure PDF file. In this case, the nonsecure PDF file inherits the secure PDF file's security settings.

These tables show all permitted combinations of permissions for Adobe PDF files. If you set permissions differently than those shown below, Adobe changes them based on the information shown in these tables.

KeyLength	Modify	Annotations	FormFields	Assembly
40	Allowed	Allowed	Allowed	Allowed
40	Not allowed	Allowed	Allowed	Allowed
40	Not allowed	Not allowed	Allowed	Allowed
40	Not allowed	Not allowed	Not allowed	Allowed
128	Not allowed	Not allowed	Not allowed	Not allowed
128	Allowed	Allowed	Allowed	Allowed
128	Not allowed	Allowed	Allowed	Allowed
128	Not allowed	Not allowed	Allowed	Allowed
128	Not allowed	Not allowed	Not allowed	Allowed
128	Not allowed	Not allowed	Not allowed	Not allowed

KeyLength	Copy	Read Access
40	Allowed	Allowed
40	Not allowed	Allowed
40	Not allowed	Not allowed
128	Allowed	Allowed
128	Not allowed	Allowed
128	Not allowed	Not allowed

KeyLength	Print
40	High resolution
40	None
128	High resolution
128	Low resolution
128	None

While testing the security feature of the PDF Print Driver, Oracle Insurance has noted an issue with Adobe's products in which permissions set correctly in the PDF Print Driver are displayed incorrectly on the Adobe security settings window. It appears the permissions work as expected, regardless of whether those settings display correctly. This problem may be corrected in later updates to Acrobat and Reader.

There are, however, a couple of caveats that do not follow these permitted combinations:

For a 40-bit encryption

Adobe lets you set Annotation to *Not allowed*, even though Modify was set to *Allowed*. Below are the permission settings for this caveat.

Permission	Setting
Encryption	Yes
KeyLength	40
Print	High resolution
Modify	Allowed
Annotation	Not allowed
Form Fields	Allowed
Assembly	Allowed
Copy	Not allowed
Accessibility	Not allowed

Note Keep in mind these are the permission settings you would see in Adobe Reader or Adobe Acrobat, not PDF Print Driver INI options.

In addition...

- When you set AllowModify to *Allowed*, AllowAnnotations remains *Not allowed*, although it should be *Allowed*.
- When you set AllowAnnotations to *Allowed*, AllowAssembly remains *Not allowed*, although it should be *Allowed*.
- When you set AllowFormFields to *Allowed*, AllowFormFields and AllowAssembly both remain *Not allowed*, although they should be *Allowed*.
- When you set AllowAccessibility to *Allowed*, it remains *Not allowed*.
- When you set AllowAssembly to *Allowed*, it remains *Not allowed*.

For a 128-bit encryption

Adobe lets you set Modify to *Allowed*, even though Annotation and Form Fields were set to *Not allowed*. Below are the permission settings for this caveat.

Permission	Setting
Encryption	Yes
KeyLength	128
Print	Low resolution
Modify	Allowed
Annotation	Not allowed
Form Fields	Not allowed
Assembly	Allowed
Copy	Not allowed
Accessibility	Not allowed

Note Keep in mind these are the permission settings you would see in Adobe Reader or Adobe Acrobat, not PDF Print Driver INI options.

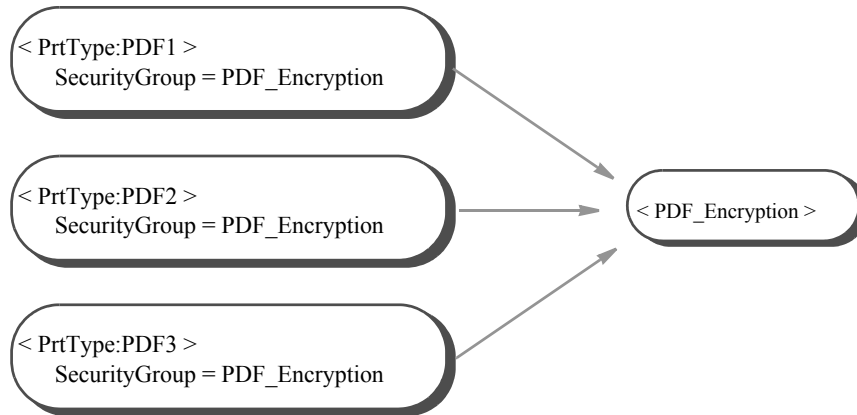
In addition...

- When you set AllowModify to *Allowed*, AllowAnnotations and AllowFormFields both remain *Not allowed*, although they be *Allowed*.
- When you set AllowCopy to *Allowed*, AllowAccessibility remains *Not allowed*.
- When you set AllowAnnotations to *Allowed*, AllowAssembly remains *Not allowed*, although it should be *Allowed*.
- When you set AllowFormFields to *Allowed*, AllowAssembly remains *Not allowed*, although it should be *Allowed*.

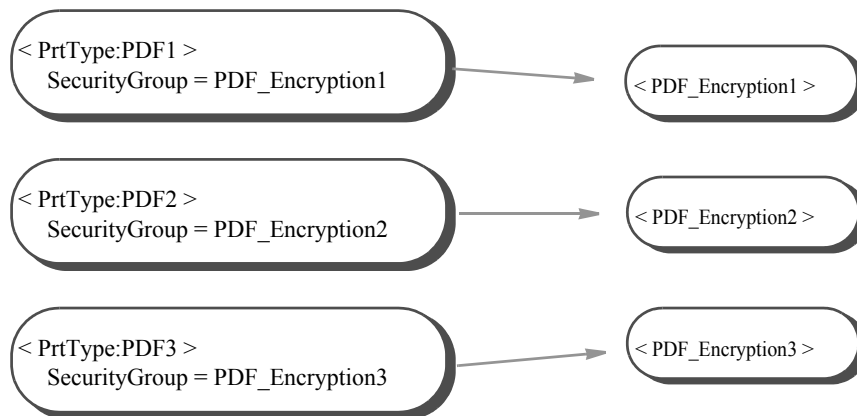
Note If you encounter any other unusual combinations, you may want to contact Adobe.

Setting up multiple security groups

You can set up your PDF PrtType:XXX control groups to share a common set of security settings or have security settings tailored to each control group. This illustration shows how you would set your printer options to use the same security settings:



Or, you can have custom security settings for each printer control group:



Choose the approach that works best in your situation.

Setting up document-level security

You can implement unique passwords at the document level. All you have to do is provide a custom *~PDFPASSWORD* function (the name can vary).

For each recipient in each transaction in each batch, the GenPrint program calls the PDFPrint function in the PDFW32.DLL file. The PDFPrint function checks the Encrypt option in the PrtType:PDF control group to see if encryption is desired.

If the Encrypt option is set to Yes, the PDFPrint function calls the PDFCryptInit function to create an encryption *context*. If no passwords are passed to the PDFCryptInit function, as is the case with the PDF Print Driver, it checks the OwnerKey option:

```
< PDF_Encryption >
  OwnerKey =
```

Note Owner and User can also be used to specify passwords directly rather than specifying keys.

Keep in mind that the SecurityGroup control group must be specified in this option:

```
< PrtType:PDF>
  SecurityGroup = PDF_Encryption
```

If the OwnerKey option is set to `~PDFKEYGEN ~PDFPASSWORD`, the INI built-in function calls the `~PDFPASSWORD` function, then passes the result to the `~PDFKEYGEN` function.

Note The `~PDFPASSWORD` function varies for each implementation and is usually part of CUSLIB.

USING THE PDFKEY UTILITY

Use the PDFKEY utility to generate the encrypted passwords used in the security control group. This control group specifies where permissions, passwords, and encryption strength are set.

Program names

Windows	PDFKW32.EXE
MVS	See the Documaker Installation Guide

Place the executable file (PDFKW32.EXE) in the directory which contains the other Oracle Documaker binary files.

Syntax `pdfkw32 /U /O /K /P /F /M /C /N /R /A /? /H`

Parameter	Description
/U	Enter the password required to open the document. You can enter up to 32 characters. Passwords are case sensitive.
/O	Enter the password required to modify the document or its security settings. You can enter up to 32 characters. Passwords are case sensitive.
/K	Enter either 40 or 128 to specify the length of the encryption key. The default is 128.
/P	Enter No to prevent users from printing the file, L to permit low quality printing, or H to permit high quality printing.
/F	Enter No to prevent users from entering form fields. The default is Yes.
/M	Enter No to prevent users from modifying the document. The default is Yes.
/C	Enter No to prevent users from copying text from the document to the clipboard. The default is Yes.
/N	Enter No to prevent users from annotating the document. The default is Yes.
/R	Enter No to prevent users from using reader accessibility tools to view the document. The default is Yes.
/A	Enter No to prevent users from adding navigation aids, such as bookmarks. The default is Yes.

Passwords can contain spaces. Simply enclose the entire password parameter in quotation marks, as shown here:

```
pdfkw32 "/O=Password With Spaces"
```

For more information about passwords see *Choosing passwords* on page 220.

USING THE PDFKEYGEN FUNCTION

You can use the PDFKEYGEN built-in function to generate the OwnerKey and UserKey information. When you include a string in this form, the built-in function generates the OwnerKey and the UserKey values:

```
OWNERPASS/USERPASS
```

The OwnerKey is returned via the function call. The UserKey is added to the INI context. To omit one or both of the passwords, modify the string as shown here:

```
"/USERPASS"  
"OWNERPASS/"  
"/"
```

To use the PDFKEYGEN built-in, first create a custom built-in that returns the password string in the format described above.

To call PDFKEYGEN (assuming the custom built-in function is registered as PDFPASSWORD), specify the OwnerKey INI option as shown here:

```
OwnerKey = ~PDFKEYGEN ~PDFPASSWORD
```

You can omit the UserKey option since it will be supplied via the above call.

You can avoid using the PDFKEYGEN built-in by specifying these options:

```
Owner =  
User =
```

But keep in mind that the custom password functions must return only the single password rather than the combination owner/user the PDFKEYGEN built-in expects.

USING AES ENCRYPTION

The PDF Print Driver can generate files that use the Advanced Encryption Standard (AES). AES is supported by Adobe's PDF standard beginning with version 1.6 (Acrobat 7 and higher).

Note To view AES-encrypted PDF files, you must have Acrobat Reader version 7.0 or higher. The current version of Acrobat Reader is 9.0 and can be downloaded for free at Adobe's web site.

AES is more secure than the RC4 encryption used in prior versions of Acrobat. To enable AES encryption, add this setting in your INI file:

```
< PDF_Encryption >  
  AESEncryption = Yes
```

Option	Description
AESEncryption	Enter Yes to create that use the Advanced Encryption Standard (AES). The default is No.

Note When you run the PDFKEY utility, it creates the INI options and settings you need to add to your INI file. For example, the utility can either write the output to the terminal session or you can send it to a file name and extension you specify by adding a parameter similar to this to the end of the command you use to run the PDFKEY utility:

```
> filename.extension
```

For best results, copy these INI options and settings from that text file and paste them into your INI file. For more information on this utility, see *Using the PDFKEY Utility* on page 226.

The actual control group name may vary. The name of the control group should match the value of the SecurityGroup option in the PDF printer INI control group.

Keep in mind:

- AES encryption causes a small increase in the size of the output file. For example, in one test, the increase was about 1.9%, from 16593 bytes to 16901 bytes. For larger files, the size increase (as a percentage) should be smaller. The AES algorithm was developed to be very efficient. No performance impact is expected.
- AES encryption requires a 128-bit key length. This will be enforced by the PDFKEY utility and the PDF Print Driver.
- To generate keys for AES encryption, include the /AES parameter when you run the PDFKEY utility.

EXAMPLE SECURITY SETTINGS

Here are some examples of how you can set up your PDF security configuration. The examples include:

- Choosing permissions, passwords, and encryption strengths
- Generating keys with the PDFKey tool
- Setting the appropriate INI options

Example 1

This example shows how to set up:

- 128-bit encryption security
- An owner password of *skywire*
- No password required to open the document
- Permissions that prevent readers from modifying information or copying information to the clipboard

Here is how you would run the PDFKey tool:

```
PDFKW32 /O=skywire /K=128 /M=N /C=N
```

The output of the PDFKey tool is:

```
OwnerKey: e551db056412e608eeab3e5f96619ac9296b49e419467fdb8879f75f95c2a816
UserKey: 605e0bb040c9ce39d650e718b3127f9b10000000fc000000000000000000000000
```

Here are the additions you would make to your INI file, assuming that the printer type control group is called PrtType:PDF.

```
< PrtType:PDF >
  Encrypt          = Yes
  SecurityGroup    = PDF_Encryption_Example_1
< PDF_Encryption_Example_1 >
  KeyLength        = 128
  OwnerKey = e551db056412e608eeab3e5f96619ac9296b49e419467fdb8879f75f95c2a816
  UserKey  = 605e0bb040c9ce39d650e718b3127f9b10000000fc0000000000000000000000
  AllowModify      = No
  AllowCopy        = No
```

You do not have to enter a password to view documents generated with these INI options. To modify a document, however, or to copy information from it to the clipboard, you must enter the owner password. Also, because 128-bit encryption option is used, you must have Acrobat 5.0 or later to view these documents.

Example 2

This example shows how to set up:

- 40-bit encryption security
- An owner password of *skywire*
- A open password of *EncryptionIsFun*
- No permission restrictions

Here is how you would run the PDFKey tool:

```
pdfkw32 /o=skywire /u=EncryptionIsFun /k=40
```

The output of the PDFKey tool is:

```
OwnerKey: 90ed4c70598767459de9523a9ce7e77ac51f4459257401fbae1936b6b1bbc7fd
UserKey: f6ac1d4b26000000000000000000000002000000058000000000000000000000
```

Here are the additions you would make to your INI file:

```
< PrtType:PDF >
  Encrypt          = Yes
  SecurityGroup= PDF_Encryption_Example_2
< PDF_Encryption_Example_2 >
  KeyLength        = 40
  OwnerKey = 90ed4c70598767459de9523a9ce7e77ac51f4459257401fbae1936b6b1bbc7fd
  UserKey  = f6ac1d4b26000000000000000000000002000000058000000000000000000000
```

To view documents generated with these settings, you must enter the password *EncryptionIsFun* when prompted by Acrobat. Once the document opens, there are no restrictions.

To change the security settings, you must enter the owner password *skywire*. Because 40-bit encryption is used, you only need Acrobat 4.0 or later to view documents created with these settings.

Example 3

This example shows how to set up:

- 128-bit encryption security
- No owner password

- A open password of *AnythingGoes*
- No permission restrictions

Here is how you would run the PDFKey tool:

```
PDFKW32 /u=AnythingGoes
```

Since the key length defaults to 128, the keys are:

```
OwnerKey: 87958ea2053c6e89302ba926dfd78a2b3d0213d8e9569734d3045a8be297370e
UserKey: 75a73844c09078ad1a587957d4328e34100000008300000000000000000000000
```

Here are the additions you would make to your INI file:

```
< PrtType:PDF >
  Encrypt      = Yes
SecurityGroup  = PDF_Encryption_Example_3
< PDF_Encryption_Example_3 >
  KeyLength    = 128
  OwnerKey     = 87958ea2053c6e89302ba926dfd78a2b3d0213d8e9569734d3045a8be297370e
  UserKey      = 75a73844c09078ad1a587957d4328e341000000083000000000000000000000
```

To view documents created with these settings, you must enter the password *AnythingGoes* when prompted. You then have unrestricted use of the document. Since there is no owner password, you can even modify security options. Because you are using 128-bit encryption, you must have Acrobat 5.0 or later to open the document.

TIPS

In case you run into problems, keep in mind...

- If you cannot open a password-protected PDF file after supplying the correct password, this probably indicates you have errors in your setup for producing secure PDF files.

The command line parameters used with the PDFKey tool, which produces the OwnerKey and UserKey hex strings, must match the secure PDF INI settings used when the PDF file is produced. Check your secure PDF INI settings for problems such as misspellings and INI settings that do not match the parameters used when running PDFKey. For example, specifying a 40-bit key length to PDFKey (/K=40) and using a KeyLength=128 INI setting will produce a PDF file that will not open in Acrobat.

- Using passwords instead of keys lessens the possibility errors.
- If you set the Print option (/P) to No in the PDFKey tool, you cannot set the AllowHighQualityPrinting option to No in the INI file. If you do, the result is an error, such as the one described above.
- When generating the hex key using the PDFKey tool, the system prints the output in INI format, as shown here:

```
D:\rel103>pdfkw32
< PDF_Encryption >
  KeyLength = 128
  OwnerKey  = 36451bd39d753b7c1d10922c28e6665aa4f3353fb0348b536893e3b1db5c579b
  UserKey   = 7880927481fd184b32c0efb547ef5adb100000006c00000000000000000000000
```

The output is formatted this way so you can copy and paste these settings into your INI file.

- When configuring PDF encryption for use with IDS, you must modify two INI files: DAP.INI and the INI file for the MRL. The DAP.INI file contains all of the SecurityGroup names and SecurityGroups for all of the MRLs for which encryption is needed. The MRL's INI file contains only the SecurityGroup name and settings used by that MRL.

You can also set up multiple SecurityGroups within one MRL. Both the DAP.INI and MRL's INI file would contain the names of all SecurityGroups in the PrtType:PDF control group as well as the individual SecurityGroups.

- If you add this INI option when templating fields, the PDF Print Driver creates annotation objects that display information about the field.

```
< PrtType:PDF >  
    FieldTemplateAnnotations = Yes
```

- The PDF Print Driver can embed its own INI control group within the PDF metadata for support purposes. This is enabled by adding the EmbeddedINI option, as shown here:

```
< PrtType:PDF >  
    EmbeddedINI = Yes
```


Chapter 11

Using the PostScript Print Driver

This chapter provides information about Documaker's PostScript Print Driver. PostScript was developed by Adobe Systems, Inc.

This chapter includes the following topics:

- *Overview* on page 234
- *Setting PostScript INI Options* on page 235
- *Using PostScript Printer Resources* on page 243

OVERVIEW

Adobe Systems created the PostScript language. It is an interpretive programming language with powerful graphics capabilities. For the most part, system-produced PostScript output will run on any printer that supports PostScript Level 2.

Note The PostScript Print Driver supports monochrome, 4-bit, 8-bit, and 24-bit color bitmaps. If your printer does not support color, the print driver will automatically convert the color graphics into monochrome graphics. Keep in mind that for the best performance you should avoid color graphics.

SETTING POSTSCRIPT INI OPTIONS

You must define the necessary printer related options for the GenPrint program to produce PostScript output. These options specify PostScript output and are located in a `PrtType:xxx` control group, such as `PrtType:PST`. Common PostScript printer options are shown below, with default values in bold:

Option	Values	Description
Device	any file or device name	The name of the file or device (LPT1) where the PCL print stream should be written. This setting is ignored by the GenPrint program but is used by Documaker Studio, Documaker Desktop, and other system programs.
Module	PSTW32	The name of the program module which contains the PostScript print driver. See also the Class option. See also <i>Using Defaults for the Module and PrintFunc Options</i> on page 237.
PrintFunc	PSTPrint	The name of the program function that is the main entry point into the PostScript print driver. See also <i>Using Defaults for the Module and PrintFunc Options</i> on page 237.
Resolution	300	The dots per inch resolution of the printer which will receive the PostScript data stream.
SendOverlays	Yes/No	Set to Yes if you have created PostScript overlays for each FAP file. See also <i>Creating Smaller PostScript Output</i> on page 238.
DSCHeaderComment		Use to specify PostScript Document Structure Convention (DSC) comments you want added to the header portion of the generated PostScript print stream. You can include as many DSCHeaderComment options as are necessary. See <i>Adding DSC Comments</i> on page 239 for more information.
OverlayPath	any directory	Set to the directory which contains the PostScript overlays for each FAP file. The default is the FormLib option of the MasterResource control group. Instead of using the above control groups and options, you could use the following options: <pre>< MasterResource > OverlayPath = <CONFIG:Batch Processing> OverlayPath = < CONFIG:Batch Processing > OverlayPath = .\PstOvl\</pre> The default is the FormLib directory pointed to by the FormLib option in the MasterResource control group., as shown here: <pre>< MasterResource > FormLib = <CONFIG:Batch Processing> FormLib = < CONFIG:Batch Processing > FormLib = ./forms/</pre>

Option	Values	Description
OverlayExt	any file extension (OVL)	The file extension of the PostScript overlays.
PageNumbers	Yes/No	Set to Yes to enable form or form set page numbering.
SendColor	Yes/No Enabled/ Disabled/Hidden	Set to Yes to enable color printing. Enabled = Option appears in the Print window and is active (available to be checked). Disabled = Option appears in the Print window but is grayed out (not available to be checked). Hidden = Option does not appear in the Print window
DownloadFonts	Yes/No	Set to Yes to enable downloading of PostScript fonts. See also <i>Creating Smaller PostScript Output</i> on page 238.
PrinterModel	file name (omit extension)	Contains the name of the PostScript Printer Definition (PPD) file. This file contains information about printer-specific features. This file must be in the directory specified by the DefLib option of the FMRES control group.
TemplateFields	Yes/No	Set to Yes to test print Xs in variable fields
FitToWidth	Yes/No	Not supported by the PostScript print driver
PrintViewOnly	Yes/No	If set to Yes, the view only sections will print. This does not apply to entry only sections, which are never printed. Entry only sections are usually worksheets. If the section is marked as hidden and view only, it will not print.
PrePrintedPaper	Yes,Disabled	Determines if the check box which lets you print or not print pre-printed objects appears on the Print window. Also determines the default for this check box—checked or unchecked. You must add this option to the INI file if you want the check box to appear on the Print window. The default for this option includes the check box on the Print window and leaves it unchecked. All objects except fields can be designated as pre-printed on the object's Properties window.
Class	(first three characters of the Module option)	Specifies the printer classification, such as AFP, PCL, XER, PST, or GDI. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the Class option to make sure the correct printer is available before continuing.
LanguageLevel	Level1 Level2	Level2 is the default setting and is required for complex printing tasks, such as duplexing, tray selection, and so on. Only use Level1 if your printer only supports PostScript Level 1 language features.
StapleOn StapleOff	see description	These options work in a similar fashion to the Tray# options which let you specify PostScript commands directly as a quoted string or to look up the PostScript commands to use in your printer's PPD file. For detailed information, see <i>Stapling Forms</i> on page 240.

Option	Values	Description
SelectRecipients	Yes/No Enabled/ Disabled/Hidden	Enabled = Option appears in the Print window and is active (available to be checked). Disabled = Option appears in the Print window but is grayed out (not available to be checked). Hidden = Option does not appear in the Print window.
SetOverprint		Enter Yes if you are using a highlight color printer, such as the Xerox DocuTech/DocuPrint 180 Highlight Color printer, and you want to remove the white outline that appears around black letters printed on a highlight color background. If you are using pre-compiled overlays, be sure to re-create the overlays after you set this option to Yes. If you still see a small white outline around the characters in your printed output, your printer may need to be re-calibrated. Contact your printer vendor to fine tune your printer calibration.

Using Defaults for the Module and PrintFunc Options

Default values for the Module and PrintFunc options in the PrtType:xxx control group are provided when you use a standard print type name or print class, such as AFP, PCL, PDF, PST, VPP, XER, XMP, or GDI.

These defaults keep you from having to enter the Module and PrintFunc names in your INI file. For example, if you want to generate PST print files, you can specify these INI options:

```
< Printer >
  PrtType   = MYPST
< PrtType:MYAFP >
  Class     = PST
```

And the system will default these options for you:

```
< PrtType:MYAFP >
  Module     = PSTPRT
  PrintFunc  = PSTPrint
```

Avoiding a White Outline Around Letters

On some highlight color printers, such as the Xerox DocuTech/DocuPrint 180 Highlight Color printer, if you print black text on a colored shaded area, the black text is printed with a white outline around the letters. To eliminate the white outline, add the SetOverprint option to your PostScript printer INI control group and set it to Yes.

PRINTING UNDER WINDOWS

Windows XP/2000 does not recognize printer ports such as LPT1. Change the PrtType control group in the FSIUSER.INI file to reflect the print server name and print device. Here is an example:

```
< PrtType:PST >
  Device = \\FSISRV03\\OPTRA1
```

GENERATING POSTSCRIPT FILES ON z/OS

You can generate PostScript output files on z/OS systems with an updated (version 11.0 or later) PSTLIB. Be sure to include these settings in your FSISYS.INI file to print PostScript on z/OS:

```
< Printer >
  PrtType      = PST
< PrtType:PST >
  Module       = PSTW32
  Printfunc    = PSTPrint
  SendOverlays = (Yes or No)
  SendColor    = (Yes or No)
  DownloadFonts = (Yes or No)
```

CREATING SMALLER POSTSCRIPT OUTPUT

The PostScript print driver automatically downloads (embeds) only the fonts that are needed. This results in smaller output files.

Note To produce a PostScript print stream that only downloads (embeds) the minimum set of fonts required by the PostScript print stream, you *cannot* use overlays.

All PostScript fonts referenced in the FXR file are downloaded if the SendOverlays option is set to Yes because the system does not know which fonts are used by the overlays.

You must set these PostScript INI options as shown to tell the PostScript print driver to download the minimum set of fonts required by a print stream:

```
< PrtType:PST >
  DownloadFonts = Yes
  SendOverlays = No
```

If you are running the GenPrint program, you will need to tell GenPrint to load the FAP files (instead of overlays) by using the DownloadFAP option:

```
< RunMode >
  DownloadFAP = Yes
```

Bitmap Compression

The PostScript print driver supports bitmap compression. Compression is enabled by default. To disable compression, add this option to the PostScript printer control group:

```
< PrtType:XXX >
  Compression = No
```

Color bitmaps are compressed in JPEG format.

Monocolor bitmaps are compressed using Run Length Encoding (RLE) compression. If compression or color is disabled, 4-bit and 8-bit color bitmaps are printed as monocolor bitmaps. For compatibility with previous releases, 24-bit color bitmaps are printed in color when compression is disabled and color is enabled.

PostScript print streams with bitmap compression are often smaller and may be produced faster than PostScript print streams without bitmap compression.

PostScript print streams with compressed multi color bitmaps will see the greatest reduction in terms of file size and time to produce.

The 4-bit and 8-bit color bitmaps printed in color with compression will likely produce larger print streams than 4-bit and 8-bit color bitmaps which have been converted to monocolour (black and white) bitmaps.

Keep in mind:

- For any bitmap to print in color, you must make sure the bitmap (LOG) is marked as *Print in Color* in the FAP file. Also make sure you set the SendColor option to Yes in the PCL or PostScript printer control group before printing.
- When using Forms Integrity Manager (FIM) to compare a version 11.2 or later PostScript print stream with bitmap compression against an older PostScript print stream without bitmap compression, FIM will report that some bitmaps are not identical. Older PostScript print streams without bitmap compression generated the bitmap data in multiple streams while the newer compressed bitmaps are always generated within a single stream. In this case, FIM will report the older print streams contains multiple *Overlay Images* entries while the new print streams contain a single *Overlay Images* entry. Also, FIM may report differences in some attributes (height, width, raster size, and so on) of *Overlay Images* and *Variable Images* due to differences in how bitmaps are emitted.

ADDING DSC COMMENTS

Use the DSCHeaderComment option to specify the PostScript Document Structure Convention (DSC) comments you want added to the header portion of the generated print stream. You can include as many DSCHeaderComment options as are necessary.

This example shows how, in addition to specifying PostScript commands in the Tray# options, you can also include DSC comments you want added to the header portion of the generated PostScript print stream:

```
< PrtType:PST >
  Device           = test.ps
  DownloadFonts    = Yes,Enabled
  DSCHeaderComment = %%DocumentMedia:Media1 612 792 75 (White)
(Tray1)
  DSCHeaderComment = %%+ Media2 612 792 75 (White) (Tray2)
  DSCHeaderComment = %%+ Media3 612 792 75 (White) (Tray3)
  DSCHeaderComment = %%+ Media4 612 792 75 (White) (Tray4)
  LanguageLevel    = Level2
  Module            = PSTW32
  PageNumbers       = Yes
  PrinterModel      = XDP92C2
  PrintFunc         = PSTPrint
  Resolution        = 300
  SendColor         = No,Enabled
  Tray1             = "<< /MediaType (Tray1)/MediaColor(White) /
MediaWeight 75>>setpagedevice"
  Tray2             = "<< /MediaType (Tray2)/MediaColor(White) /
MediaWeight 75>>setpagedevice"
  Tray3             = "<< /MediaType (Tray3)/MediaColor(White) /
MediaWeight 75>>setpagedevice"
  Tray4             = "<< /MediaType (Tray4)/MediaColor(White) /
MediaWeight 75>>setpagedevice"
  SendOverlays      = Yes,Enabled
```

The DSC header comments are added at the beginning of the generated PostScript print stream as shown here:

```

%!PS-Adobe-3.0
%%Title: INSUREDS COPY
%%Creator: FormMaker PostScript Driver
%%CreationDate: Thu Apr 04 17:50:57 2002
%%For: INSURED
%%Pages: (atend)
%%DocumentData: Clean7Bit
%%DocumentSuppliedResources: font (atend)
%%DocumentMedia:Media1 612 792 75 (White) (Tray1)
%%+ Media2 612 792 75 (White) (Tray2)
%%+ Media3 612 792 75 (White) (Tray3)
%%+ Media4 612 792 75 (White) (Tray4)
%%EndComments

```

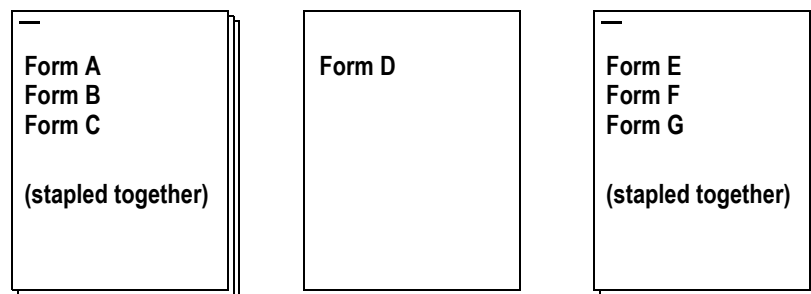
STAPLING FORMS

Use the StapleOn and StapleOff INI options in the PostScript printer control group to control staple support. These options work in a similar fashion to the Tray# INI options which let you specify PostScript commands directly as a quoted string or to look up the PostScript commands to use in your printer's PPD file.

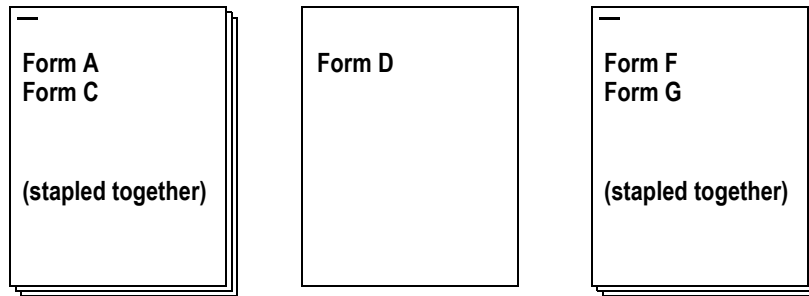
Here is an example. Suppose you have seven forms in the form set and all but one (Form D) are to be stapled. There are two recipients who are to receive these forms as shown in this table:

Form	Staple?	Recipients
A	Yes	INSURED, AGENT
B	Yes	INSURED
C	Yes	INSURED, AGENT
D	No	INSURED, AGENT
E	Yes	INSURED
F	Yes	INSURED, AGENT
G	Yes	INSURED, AGENT

The INSURED recipient's forms print as:



The AGENT recipient's forms print as:



By default, the PostScript print driver will use these commands:

```
< PrtType:PST >
...
  StapleOn  = "<</Staple 3 >> setpagedevice"
  StapleOff = "<</Staple 0 >> setpagedevice"
```

You can override PostScript staple commands by providing an alternate PostScript command to use via the StapleOn and StapleOff options in your PostScript printer control group.

You can issue PostScript staple commands in these forms:

- A quoted string containing the PostScript commands. The quoted string should contain the appropriate PostScript commands for turning stapling on or off. Here is an example:

```
StapleOn = "1 dict dup /Staple 0 put setpagedevice"
```

- A UI keyword from a PPD file. UI keywords represent features that commonly appear in a user interface (UI). They provide the code to invoke a user-selectable feature within the context of a print job, such as the selection of an input tray or manual feed. The entries of UI keywords are surrounded by these structure keywords:

```
*OpenUI/*CloseUI or *JCLOpenUI/*JCLCloseUI
```

Here is an example of an OpenUI structure for XRXFinishing:

```
*OpenUI *XRXFinishing/Finishing: PickOne
*OrderDependency: 60.0 AnySetup *XRXFinishing
*DefaultXRXFinishing: None

*XRXFinishing None/None: "
1 dict dup /Staple 0 put setpagedevice"
*End

*XRXFinishing Single_Portrait_Staple/Single Portrait Staple: "
2 dict dup /Staple 3 put
  dup /StapleDetails 2 dict dup /Type 1 put dup /StapleLocation
(SinglePortrait) put
  put setpagedevice"
*End

*XRXFinishing Single_Landscape_Staple/Single Landscape Staple: "
2 dict dup /Staple 3 put
  dup /StapleDetails 2 dict dup /Type 1 put dup /StapleLocation
(SingleLandscape) put
  put setpagedevice"
*End

*XRXFinishing Dual_Portrait_Staple/Dual Portrait Staple: "
```

```
2 dict dup /Staple 3 put
  dup /StapleDetails 2 dict dup /Type 1 put dup /StapleLocation
  (DualPortrait) put
  put setpagedevice"
*End
```

```
*XRXFinishing Dual_Staple/Dual Landscape Staple: "
2 dict dup /Staple 3 put
  dup /StapleDetails 2 dict dup /Type 1 put dup /StapleLocation
  (DualLandscape) put
  put setpagedevice"
*End
```

```
*?XRXFinishing: "(Unknown) = flush"
```

```
*CloseUI: *XRXFinishing
```

A PostScript Printer Definition (PPD) file is supplied with a PostScript printer. This file contains information about printer-specific features. You specify the PPD file you want to use in the PrinterModel option in your PostScript printer control group (just the file name, no drive, path, or file extension). If the PrinterModel option contains the name of a PPD file, this file must be in the directory specified in the DefLib option in the FMRes control group.

This example shows a PostScript printer group that uses a PPD file for a DocuPrint 65 printer (XRD60651.PPD) and specifies StapleOn and StapleOff options using keyword settings from the PPD file:

```
< PrtType:PST >
...
PrinterModel = XRD60651
StapleOn = *XRXFinishing Single_Portrait_Staple/Single Portrait
Staple:
  StapleOff = *XRXFinishing None/None:
```

USING POSTSCRIPT PRINTER RESOURCES

A number of resources participate in the total printing process. They reside in directories specified in the MasterResource control group.

Fonts

The system supports PostScript Type 1 fonts. These fonts must reside in the directory specified in the FontLib option in the MasterResource control group when the DownloadFonts option is set to *Yes*.

Overlays

Use the OVLCOMP utility to create PostScript overlays from FAP files. These overlays must reside in the directory specified in the OverlayPath option in the MasterResource control group when the SendOverlays option is set to *Yes*.

True Type fonts within a Postscript print stream

TrueType fonts can be used within a Postscript print stream by converting TrueType fonts into PostScript Type 42 fonts. The Type 42 font format is a PostScript wrapper around a TrueType font, allowing PostScript-capable printers containing a TrueType rasterizer to print TrueType fonts.

Note The TrueType font must contain a "POST" table that contains the additional information needed to use the TrueType font with a PostScript printer. Most of the TrueType fonts used in REL121.FXR can be converted into Type 42 fonts. However, the Andale Duospace TrueType font cannot be converted into the Type 42 format and the TrueType fonts for Chinese, Japanese, and Korean (Albany WorldType) cannot be converted into Type 42 fonts.

For detailed information on using True Type fonts within a Postscript print stream, see [Documaker Desktop User Guide](#).

PostScript Printer Definition (PPD) Files

A PostScript Printer Definition (PPD) file is supplied with a PostScript printer. This file contains information about printer-specific features. If the PrinterModel option contains the name of a PPD file, this file must be in the directory specified in the DefLib option in the FMRES control group.

Chapter 12

Using the RTF Print Driver

This chapter provides information about Documaker's RTF Print Driver. Use this print driver to create RTF (Rich Text Format) files from Documaker form sets.

This chapter includes the following topics:

- *Overview* on page 246
- *Setting RTF INI Options* on page 247
- *Generating Separate Files* on page 251
- *Adding or Removing Frames* on page 252
- *Creating Form Fields* on page 253
- *Setting Margins* on page 254
- *Removing the Contents of Headers and Footers* on page 255
- *Working with the Documaker Add-In* on page 256

OVERVIEW

The RTF Print Driver lets you create a medium-fidelity export of the contents of a form set in a format you view or edit with most popular word processors.

The EPT Print Driver uses this capability to email form sets. See *Using the EPT Print Driver* on page 77 for more information.

SETTING RTF INI OPTIONS

You can use these INI options with the RTF Print Driver:

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

Option	Description
--------	-------------

PrtType	Enter RTF This option lets the system know that RTFLIB is a print driver so it will include it on the Print window when you print from Documaker Desktop.
---------	---

Use these options in the PrtType:RTF control group to further customize the RTF Print Driver.

```
< PrtType:RTF >
  Class                      =
  Device                     =
  Module                     = RTFW32
  PrintFunc                  = RTFPrint
  InitFunc                   =
  TermFunc                   =
  AllowInput                 =
  BarCode                    =
  Bitmap                     =
  BMSUB                      =
  BMSUBChar                  =
  Box                         =
  Chart                      =
  EmptyFooters               =
  EmptyHeaders               =
  Field                      =
  GenerateAddInXML           =
  MinBottomMargin            =
  MinLeftMargin              =
  MinRightMargin             =
  MinTopMargin               =
  OverlayExt                 =
  PageNumbers                =
  PrePrintedPaper            =
  ReplaceFAPHeadFoot         =
  RTFCompatibilityOptions    =
  SendColor                  =
  StreamBufferSize           =
  TemplateFields             =
  Text                       =
  Vector                     =
  WriteFrames                =
```

Note You also need to specify an output device name on the Print window.

Option	Description
--------	-------------

Class	Use to specific the printer classification, such as AFP, EPT, PCL, XER, PST, or GDI. For the TRF Print Driver, enter RTF. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. These internal functions check the Class option to make sure the correct printer is available before continuing. The default is the first three characters of the Module option settings.
-------	---

Option	Description
Device	This setting is ignored by the GenPrint program but is used when printing from GUI Documaker applications like Documaker Studio and Documaker Desktop.
Module	This option tells the system to load the module which contains the print driver. The default is RTFW32.
PrintFunc	This option tells the system which print function to use. The default is RTFPrint.
InitFunc	Enter RTFInit. This tells the system to use a special initialization function called RTFInit which is located in RTFW32.DLL.
TermFunc	This option tells the system to use a specified termination function. The default is RTFTerm, which is located in RTFW32.DLL.
AllowInput	Enter Yes to enable form fields in the output file. The default is No. This option is used by Studio and the FAP2RTFutility to print or convert a FAP file with variable fields into an RTF file with form fields. The form fields can then be used as entry fields in a word processor such as Microsoft Word. If the variable field is a date or time format field, you can use the WordDateFormats and WordTimeFormats control groups to convert the Documaker date or time format into a Word date or time format.
BarCode	Enter No if you want to omit these kinds of objects from the RTF file. The default is Yes.
Bitmap	Enter No if you want to omit these kinds of objects from the RTF file. The default is Yes.
BmSub	Enter No if you <i>do not</i> want to replace invalid characters. The default is Yes. The names of the variable fields in the FAP file are stored in the Bookmark field in the RTF file. If you have spaces in the variable field names, set the BmSub option to Yes to avoid the following message when entering data into form fields: <code>The bookmark name is not valid</code>
BmSubChar	If you entered Yes in the BmSub option, enter the character you want to use when invalid characters are substituted. The default is the underscore (_).
Box	Enter No if you want to omit these kinds of objects from the RTF file. The default is Yes.
Chart	Enter No if you want to omit these kinds of objects from the RTF file. The default is Yes.
EmptyFooters	Enter Yes if you want the system to remove the contents, both text and graphics, from the footers. The resulting RTF file will have empty footers. The default is No.
EmptyHeaders	Enter Yes if you want the system to remove the contents, both text and graphics, from the headers. The resulting RTF file will have empty headers. The default is No.
Field	Enter No if you want to omit these kinds of objects from the RTF file. The default is Yes.

Option	Description
GenerateAddInXML	<p>Enter Yes when creating RTF output for the Documaker Add-In for Microsoft Word if you want the output to contain embedded XML and RTF bookmarks which will identify the document elements in the document. When you open the output in Word, the Documaker Add-In converts these document elements into content controls.</p> <p>The default is No.</p>
MinBottomMargin MinLeftMargin MinRightMargin MinTopMargin	<p>Use these options to specify the minimum required bottom, top, left, and right margins in FAP units (2400 per inch). The default is 400 FAP units, or 1/6 of an inch.</p> <p>The margin values you specify in these options override those set in the FAP file if the page margins in the FAP file are smaller.</p> <p>This example sets the margins at 1/2 inch:</p> <pre>< PrtType:RTF > MinTopMargin = 1200 MinLeftMargin = 1200 MinRightMargin = 1200 MinBottomMargin = 1200</pre>
OverlayExt	Enter the file extension for the RTF overlays. The default is OVL.
PageNumbers	Enter Yes to have page numbers printed in the "Page X of Y" format. The default is No.
PrePrintedPaper	<p>Enter Enabled or Hidden to specify whether the check box which lets you print or not print pre-printed objects appears on the Print window. Enter Yes or No to specify whether the check box is checked.</p> <p>The default is Yes,Enabled, which includes the check box on the Print window and leaves it checked. All objects except fields can be designated as pre-printed on the object's Properties window in Studio.</p>
ReplaceFAPHeadFoot	Enter Yes to override the size of the headers and footers from the original form when importing an RTF file. The default is No, which retains the size of the headers and footers in the original form.
RTFCompatibilityOptions	<p>Use this option to insert RTF compatibility option commands into the RTF output file. The RTF compatibility options are documented in the RTF Specification manuals, available on Microsoft's web site.</p> <p>The default is an empty string.</p>
SendColor	Enter Yes to enable color printing. The default is No.
StreamBufferSize	<p>Enter the number of bytes to use for buffering. You may can use this option for performance tuning.</p> <p>The default is zero (0), which means to use the default buffer size.</p>
TemplateFields	<p>The default, Yes, tells the system to test print Xs in variable fields. If you also include Enabled (Yes,Enabled), the Template Variable Fields field in Studio is checked.</p> <p>The default varies, depending on the system:</p> <ul style="list-style-type: none"> • No,Enabled in Documaker Studio • No,Hidden in Documaker Desktop • No in Documaker Server
Text	Enter No if you want to omit these kinds of objects from the RTF file. The default is Yes.

Option	Description
Vector	Enter No if you want to omit these kinds of objects from the RTF file. The default is Yes.
WriteFrames	Enter No if frames are not required. The default is Yes.

Use the WordDateFormats and WordTimeFormats control groups to convert the Documaker date or time format into a Word date or time.

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

To the left of the equal sign, you list the Documaker format used on the section. To the right, you list the Word format you want to use. If not specified, the system converts date formats to MM/DD/YY and time formats to hh:mm:ss.

For more information, see *Field Formats* on page 287.

GENERATING SEPARATE FILES

You can generate separate files for each transaction when you choose RTF (or PDF) from WIP or batch print.

The name of the files will have a rolling number appended to the end of the name that starts the process and is filled in on the Print window. This is automatically handled and you do not have to set INI options to get the WIP or batch print to work as long as your PrtType name is PrtType:RTF.

There are several INI options you can use to override the naming process and also name other print drivers that require this unique handling.

```
< BatchPrint >
    NoBatchSupport= RTF
    PreLoadRequired= RTF
```

These are the default settings and cannot be overridden. However, you can specify other PrtType print driver definitions you want to fall into these same categories.

Option	Description
NoBatchSupport	Indicates that the named PrtType items, separated by semicolon, do not really support batch transactions and require special handling.
PreLoadRequired	Lets you specify all the PrtType items, separated by semicolon, that should be forced to load the form set prior to the starting print. Most print drivers don't require this special requirement, but some, such as PDF do.

Also, you can name PrtType specific items under the BatchPrint control group to override the normal Device naming option. Here is an example:

```
< BatchPrint >
    PDF = ~HEXTIME .PDF
    RTF = ~HEXTIME -~KeyID .RTF
```

Any batch print sent to PrtType:PDF (picking PDF on the Print window) will override the name and store the current hexadecimal date and time, such as BCF09CA4.PDF, which is an eight-character name, as the name of each transaction's output.

Also, you can combine INI built-in calls as shown in the RTF example. Here any WIP or batch print sent to RTF will name the files using the HEXTIME and the KeyID from the WIP transaction. This will result in names similar to this:

```
BCF099A4-123456.RTF
```

Note that you must leave a space after the built-in INI function name for it to work properly. That space will not appear in the resulting output name.

ADDING OR REMOVING FRAMES

By default, the RTF print driver uses frames to replicate the look of a document. If you do not want the frames, which print as boxes around the various document objects, to appear, set the WriteFrames option to No.

```
< PrtType:RTF >  
  WriteFrames =
```

This option is turned on by default and causes the resulting output to maintain layout fidelity — what you see is what you get. *Frames* are a Word (and RTF) method of defining fixed object locations. Essentially, this creates a box or frame for each object to live within in the resulting document. The result is that the output looks very much like the original form set does in WIP or Archive.

For instance, you can use the RTF print driver to print form sets to an RTF file. Once the RTF file is created, you can then open it in a word processor. To avoid having frames in the file, you would set this option to No.

If you set this option to No, the resulting output will contain the data, but may not format in the same way, or in the same places, as your original document when it was opened in Word.

CREATING FORM FIELDS

You can use the RTF print driver to convert variable fields into RTF form fields. For example, a variable address field is converted into an RTF form field. The format of the field is retained. If, for example, the address field contained all uppercase characters, this would be reflected in the corresponding RTF form field.

To print form fields, include this INI option:

```
< PrtType:RTF >  
    AllowInput = Yes
```

Note This works with print types RTF and RTF_NoFrame.

You may also need to include the WordTimeFormats and WordDateFormats control groups. You can use these control groups in case you are using a time or date format that has no equivalent in Word. The following groups and options let you map a Documaker format to a Word format.

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

To the left of the equals sign, you list the Documaker format used on the section. To the right, you list the Word format you want to use.

SETTING MARGINS

The RTF print driver produces margins by calculating what is required and putting the result in the RTF output. You can, however, set minimum required margins using the RTF print type control group.

You must set the minimum required margins in FAP units (2400 dots per inch). Here are the default settings:

```
< PrtType:RTF >
  MinTopMargin      = 400
  MinLeftMargin     = 600
  MinRightMargin    = 600
  MinBottomMargin   = 400
```

Margin values specified in the INI file override those set in the FAP file if the page margins in the FAP file are smaller.

Note The changes in the margins are noticeable when you open the document in an application such as Microsoft Word. You will see the left and right margins shifting based on what you specified in the INI file. The top and bottom margins (seen on the left side of the page) will also vary based on what you specified in the INI file.

REMOVING THE CONTENTS OF HEADERS AND FOOTERS

Use these options to remove the contents, including graphics and text, from headers and footers when creating RTF files:

```
< PrtType:RTF >  
  EmptyFooters = Yes  
  EmptyHeaders = Yes
```

Option	Description
EmptyHeaders	Enter Yes to remove the contents from any headers in the file. This includes both text and graphics. The default is No.
EmptyFooters	Enter Yes to remove the contents from any footers in the file. This includes both text and graphics. The default is No.

WORKING WITH THE DOCUMAKER ADD-IN

Use these options to generate enhanced RTF files for the Documaker Add-In for Microsoft Word. You can then open this enhanced RTF output in Word (with the Documaker Add-In) to view information about the Documaker resources used to make up the form. For example, this lets you...

- See where the data fields and Documaker graphics are located in the document
- See which Documaker forms and sections were used and where they are in the document
- Identify any incorrect sections, forms, graphics, or data fields in the document
- Identify sections that may be candidates for change
- Identify sections that could be used as the basis for a new section for the same type or new type of form or product line

To generate these enhanced RTF files, you must set the `GenerateAddInXML` option to Yes.

```
< PrtType:RTF >  
  GenerateAddInXML          = Yes  
  RTFCompatibilityOptions = \dntblnsbdb
```

Option	Description
GenerateAddInXML	Enter Yes when creating RTF output for the Documaker Add-In for Microsoft Word if you want the output to contain embedded XML and RTF bookmarks which will identify the document elements in the document. When you open the output in Word, the Documaker Add-In converts these document elements into content controls. The default is No.
RTFCompatibilityOptions	Use this option to insert RTF compatibility option commands into the RTF output file. The RTF compatibility options are documented in the RTF Specification manuals, available on Microsoft's web site. The default is an empty string.

Note Use the `RTFCompatibilityOptions` option to target Microsoft Word 2007 and higher. The RTF option that targets Word 2007 is `\dntblnsbdb`. The Documaker enhanced RTF output will have fewer line wrapping differences when this option is on.

Chapter 13

Using the VIPP Print Driver

This chapter provides information about Documaker's VIPP (Variable Data Intelligent PostScript PrintWare) Print Driver.

This chapter includes the following topics:

- *Overview* on page 258
- *Using VIPP Resource Files* on page 260
- *Managing VIPP Resources* on page 264
- *Setting VIPP INI Options* on page 267
- *VIPP Limitations* on page 274
- *VIPP Troubleshooting* on page 275

OVERVIEW

Variable Data Intelligent PostScript PrintWare (VIPP) was created by Xerox in the early 1990s to enable high-performance variable data printing on PostScript devices. VIPP is based on PostScript and works by extending the PostScript programming language. VIPP can be used on any PostScript compatible printer, including Xerox and third-party network, workgroup, and production devices that have been licensed for VIPP.

VIPP is supported on these devices:

- DocuPrint NPS (monochrome and color)
- DocuPrint N-series
- DocuSP (Document Services Platform) controllers, including iGen3
- DocuColor, EFI, and Creo controllers, (including iGen3)

The Documaker VIPP print driver requires that you have VIPP version 5.3 or later installed on your printer's controller.

Note Contact your Xerox representative to see if your specific printer supports VIPP and to obtain VIPP licensing and installation of the latest VIPP version. To use the Documaker VIPP print driver, any supported device must have a local file system you can access to transfer resource files. Check with your Xerox representative for any limitations or considerations when using VIPP on your specific printer. For example, DocuColor systems may have limited or no support for stapling, duplexing, and paper tray (media) selection. In addition, older models of DocuTech and DocuPrint printers may have limited or no support for caching resource files.

The Documaker VIPP print driver produces native mode VIPP output. Native mode refers to files composed solely of VIPP commands. VIPP commands are used to place text, lines, boxes, shades, and graphics directly on the page. Native mode is the default VIPP mode.

A VIPP print job can refer to external resource files such as fonts, TIFF and JPEG graphics files, and page overlays (segments).

VIPP provides a mechanism called VIPP Projects that lets you manage all of the resources needed for a VIPP print job.

VIPP Projects allow you to organize the resources of a job under a single name (the project) and group the jobs by family (the folder).

A folder is a collection of projects that share some common features. For example, you may decide to create one folder for each customer, each division, or each line of business. Within each folder, you could define multiple projects. A folder can contain common resources (company logo, standard boilerplate page segments, and so on) that are shared by the projects within the folder. The projects will contain resources that are unique to the project. You can also have resources that are global across all projects and folders.

Having multiple folders and projects provide a great deal of flexibility in how you organize and share your resources. Folders and projects can even provide the logical grouping of the physical resources used by the job at one or more steps during in the job life cycle (development, testing, production, and so on).

This is a sample structure:

Folder A - Dallas Division

Project 1

Project 2

Project 3

Folder B - Atlanta Division

Project 1

Project 2

Project 3

Folder C - Silver Springs Division

Project 1

Project 2

Project 3

USING VIPP RESOURCE FILES

The resource files referenced by a Documaker VIPP job are:

- Pictures (images) in TIFF or JPEG format
- Overlays (segments) in VIPP format
- PostScript fonts
- Font encoding tables

Note All VIPP resource files stored on the VIPP console that are referenced by a Documaker VIPP job must have lower case file names.

CONVERTING BITMAPS INTO VIPP IMAGE FILES

VIPP supports bitmap files in TIFF and JPEG format. The Documaker VIPP print driver assumes that mono-color (1 bit per pixel) graphics have been converted into TIFF format and multicolor (more than 1 bit per pixel) graphics have been converted into JPEG format.

Scanned images are usually converted into multicolor graphics even though the images can appear to be black and white. There are a number of ways to convert your graphics into TIFF and JPEG files as expected by the VIPP print driver.

- Use the Conversion wizard in Documaker Studio. Choose the Manage, Conversion option from the main menu. Select *VIPP image files* as the Final Conversion File Type. Selecting VIPP image files tells the system to create a TIFF file or a JPEG file, based on the number of colors used in the graphic.
- Use the LOG2VIPP utility. The utility creates a TIFF file or a JPEG file based on the number of colors used in the graphic. See the Utilities Reference for details.

Note All VIPP resource files stored on the VIPP console that are referenced by a Documaker VIPP job must have lower case file names. It is usually easier to make sure the resource file names are lower case before they are transferred to the UNIX workstation console attached to the VIPP printer.

CONVERTING FAP FILES INTO VIPP SEGMENT FILES

VIPP supports pre-compiled printer overlays (called segments). A segment is a VIPP native mode or a PostScript fragment intended to be reproduced once or several times at specific locations on one or more pages. You can use the OVLCOMP utility to convert Documaker FAP files into VIPP segment files.

Here is an example of the syntax for this utility. For more information, see the [Utilities Reference](#):

```
OVLCOMP /I=fapfile /X=fxrfile /L=VPPW32 /F=VPPPrint /U=VPP /C
```

Parameter	Description
-----------	-------------

/I	Enter the name of the FAP file. Omit the extension.
----	---

Parameter	Description
/X	Enter the name of the FXR file. Omit the extension.
/L	For the VIPP print driver, enter VPPW32 .
/F	For the VIPP print driver, enter VPPPrint . Case is important when using this parameter, therefore, you must enter it exactly as shown here: /F=VPPPrint
/U	(Optional) Enter the name of your VIPP printer group. Here is an example: /U=VPP
/C	(Optional) Include this parameter if you want to use color.

You will need a FSISYS.INI file in the directory that you run the OVLCOMP utility from. Within the FSISYS.INI file, you should have a VIPP printer group defined. For example, below is a subset of the INI settings you might find in a VIPP printer group.

```
< PrtType:VPP >
  Module           = VPPW32
  OverlayExt       = .seg
  PrintFunc        = VPPPrint
  SendOverlays     = Yes,Enabled
```

You can specify the overlay (segment) extension you want to use by including the OverlayExt option in your VIPP printer control group and telling OVLCOMP the name of your VIPP printer group (/U=VPP). Use the same OverlayExt setting in your VIPP printer control group when producing a VIPP print stream that uses overlays (segments). If you omit the OverlayExt option, the default file extension for an overlay is *OVL*.

Another way to create VIPP overlays (segments) is to use the Conversion wizard in Documaker Studio. Select the Compile Sections (FAP files) to Print Files option and choose Section to VIPP as the conversion type.

Note All VIPP resource files stored on the VIPP console that are referenced by a Documaker VIPP job must have lower case file names. It is usually easier to make sure the resource file names are lower case before they are transferred to the UNIX workstation console attached to the VIPP printer.

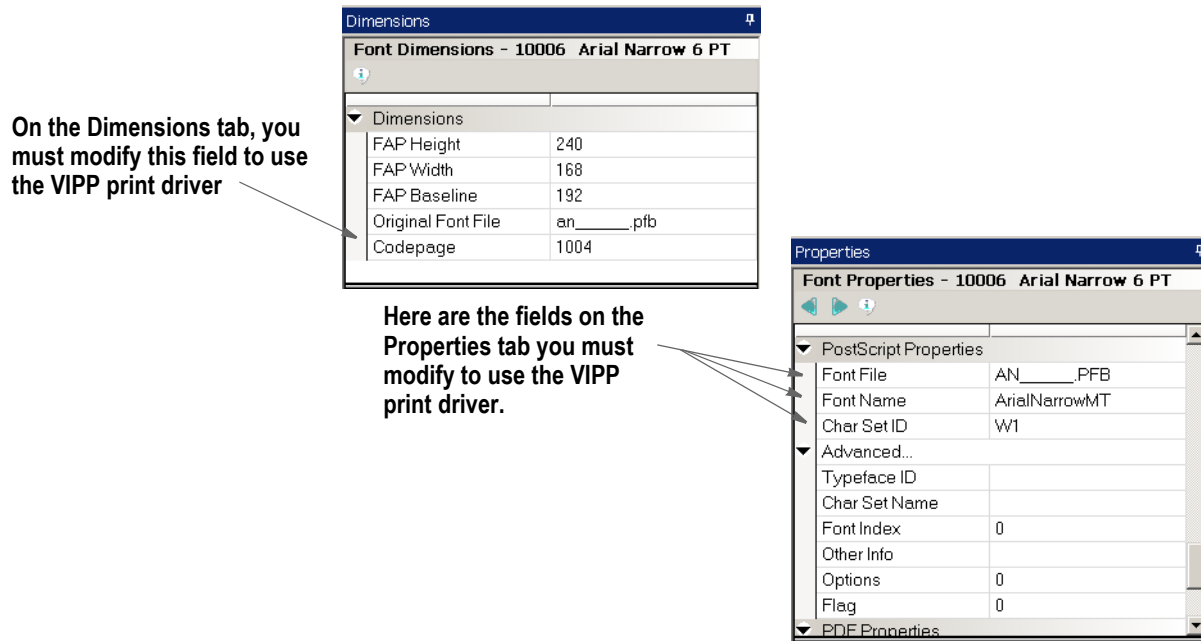
VIPP FONTS

VIPP supports PostScript fonts as VIPP resources. While VIPP supports any font type (Type 1, Type 3, and composite) supported by the PostScript interpreter, Documaker only supports Type 1 PostScript fonts. The PostScript fonts you use must be defined in your font cross-reference (FXR) file.

If you are using a base FXR file, like REL121.FXR, the base PostScript fonts are already set up for you in the FXR file. The same PostScript fonts used for printing with the Documaker PostScript print driver are also used with the Documaker VIPP print driver.

If you are using a custom FXR file and you have not set up your FXR file for printing PostScript, then you will need to add the PostScript fonts to your FXR file. You can use the Import option for the Font manager to import PostScript fonts into your FXR file. The primary fields used by the PostScript and VIPP print drivers are the CodePage field on the Dimensions tab, and the Font File, Font Name, and Char Set ID fields in the PostScript section of the Properties tab.

Here are examples of the Dimensions and Properties tabs in Documaker Studio for a font record in your FXR file:



Field	Enter...
CodePage	Under Windows, the system uses the ANSI code page. Normally, this field is set to 1004 or is left blank.
Font File	The PostScript Type 1 font file name, including the <i>PFB</i> extension. Font manager fills this field when you insert a PostScript font.
Font Name	The full font name, such as Times-Roman. Font manager fills this field when you insert a PostScript font.
Char Set ID	A character set (also known as a symbol set) identifies the set of symbols provided by the font. It is used by PostScript printing to build an internal code page. Use W1 for the fonts that use the standard Windows ANSI code page. The character set ID and code page values should match those specified in the CODEPAGE.INI file. Code page 1004 and Char Set ID W1 are used for fonts that use the standard Windows ANSI code page.

The Handling Fonts chapter in the [Documaker Studio User Guide](#) contains more information on how to add PostScript fonts to your FXR file.

Note All VIPP resource files stored on the VIPP console that are referenced by a Documaker VIPP job must have lower case file names. It is usually easier to make sure the resource file names are lower case before they are transferred to the UNIX workstation console attached to the VIPP printer.

VIPP FONT ENCODING FILES

A PostScript font is a collection of characters. Each character in a PostScript font has a PostScript-assigned name. For example, the dollar sign (\$) character has a PostScript name of `/$dollar`. While PostScript fonts use PostScript-assigned names for each character, PostScript (and VIPP) print streams use a byte value to represent each character. For example, the dollar sign (\$) is usually represented by a value of 24 hex. An encoding table is used to match a byte value (24 hex) with the character name (`/$dollar`) contained within a PostScript font.

This table shows the relationship between the hex byte value, the equivalent decimal value, the PostScript character name, and the actual printed character using the standard ASCII encoding table.

Hex value	Decimal value	PostScript name	Character
20	32	/space	
24	36	/\$dollar	\$
2A	42	/asterisk	*
30	48	/zero	0
41	65	/A	A
61	97	/a	a
7A	122	/z	z

VIPP font encoding files serve a similar purpose as the Documaker CODEPAGE.INI file and the CodePage and Char Set ID option settings in the font cross-reference file. The Documaker VIPP print driver uses the CodePage setting for each font in the font cross-reference to determine the name of the encoding file to use. The Documaker VIPP print driver appends the letters *cp* to the value of the code page setting for each font in the font cross-reference to determine the name of the VIPP font encoding file. Therefore, if a font has a CodePage setting of *1004*, then the Documaker VIPP print driver will use a VIPP font encoding file called *cp1004*.

These VIPP encoding files are provided to correspond to the code pages used by the base Documaker font cross-reference files:

File	Description
cp1004	The VIPP encoding file used for fonts that use the standard Windows ANSI code page. Most text fonts will use this.

Note All VIPP resource files stored on the VIPP console that are referenced by a Documaker VIPP job must have lower case file names. It is usually easier to make sure the resource file names are lower case before they are transferred to the UNIX workstation console attached to the VIPP printer.

MANAGING VIPP RESOURCES

Documaker VIPP print jobs use external resources for VIPP images, segments, fonts, and encoding files. By using external resources, the amount of time needed to produce a VIPP print stream is greatly reduced (as well as the size of the print job). Because the resources are not part of the job, the VIPP resources must be deployed to the controller (often a Sun workstation) that houses the VIPP software and ultimately drives the printer.

You will need some means of transferring VIPP resource files to the controller for the VIPP printer such as:

- Windows FTP command line utility
- Third- party FTP file transfer utility
- VIPP Manage (contact Xerox for more information)

You will need to log on with root access onto the controller. For some controllers, you can use the following user ID and password.

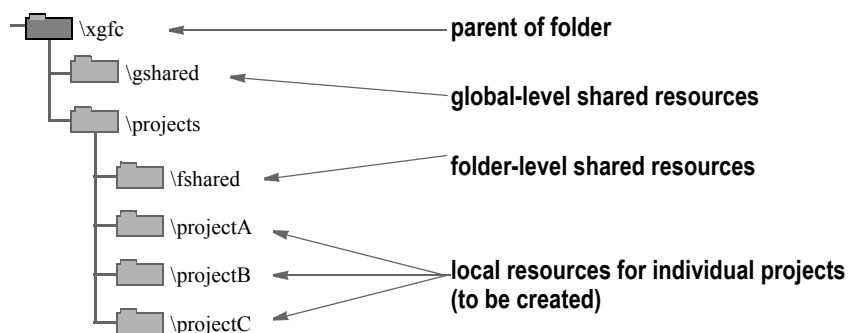
User ID: **root**
Password: **service!**

Contact your Xerox representative if you need help logging onto the controller for your VIPP printer.

As mentioned earlier, VIPP lets you organize the resources required by a VIPP job under a hierarchy of folders and projects. A folder is a collection of projects that share some common features. For example, you can decide to create one folder for each customer, each division, or each line of business. Within each folder, you could define multiple projects. A folder can contain common resources (company logo, standard boilerplate page segments, and so on) that are shared by the projects within the folder. The projects will contain resources that are unique to the project. You can also have resources that global across all projects and folders.

When VIPP is installed on the controller for your printer, VIPP is configured with a file called **xgfunix.run** (stored in the /usr/xgf/src directory). The **xgfunix.run** file contains VIPP commands that determine the VIPP resource directories.

By default, VIPP is configured with the following VIPP projects repository (collection of VIPP resources and projects):



The root path for xgfc will be /usr/xgfc on controllers that use UNIX systems.

In the `xgfunix.run` file, you might see a `SETPPATH` (VIPP command) that looks like this:

```
[ (/usr/xgfc/$$FOLDER./$$PROJECT./)    % project local paths
(/usr/xgfc/$$FOLDER./fshared/)        % project folder shared paths
(/usr/xgfc/gshared/)                  % project global shared paths
(/usr/xgfc/fontlib/)                  % project access to font lib
(/usr/xgf/encoding/)                  % project access to standard encoding
(/usr/xgf/gshared/)                    % project global shared path
(/opt/XXnps/resources/ps/mislib/)      % project access to DocuSP
resource list
] SETPPATH
```

The `$$FOLDER.` and `$$PROJECT.` strings are placeholders for project folders and project names. In the example listed earlier, `$$FOLDER` would be represented by the projects folder and `$$PROJECT` could be represented by `projectA`, `projectB`, or `projectC`.

Project paths are divided into these three levels of hierarchy or scope:

- Local scope — paths that contain both `$$FOLDER` and `$$PROJECT`. These libraries will hold resources that pertain only to the project. In the example listed earlier, `usr/xgfc/projects/projectA` would have a local scope.
- Folder scope — paths that contain only `$$FOLDER`. These libraries will hold project libraries and resources shared by projects belonging to the same folder. In the example listed earlier, `usr/xgfc/projects/fshared` would have a folder scope.
- Global scope — paths that contain neither `$$FOLDER` nor `$$PROJECT`. These libraries will hold resources shared by all projects. In the example listed earlier, `usr/xgfc/gshared` would have a global scope.

When a resource is present with the same name in more than one folder (scope), VIPP uses the following order of precedence to determine which resource file to use:

- Local scope folder
- Folder scope
- Global scope

Even the simple default VIPP repository gives you a lot of flexibility in how you manage your VIPP resources.

As you recall, Documaker LOG files are converted to VIPP image files (TIFF or JPEG files). Let's say that some of your LOG files are unique to `projectA` while others are shared by `projectA`, `projectB`, and `projectC`.

The TIFF or JPEG files that are unique to `projectA` could be stored in a local scope folder such as `usr/xgfc/projects/projectA`.

The TIFF or JPEG files that are shared between `projectA`, `projectB`, and `projectC` could be stored in a folder scope folder such as `usr/xgfc/projects/fshared`.

Similarly, Documaker FAP files are converted to VIPP segment files. Again, some of your FAP files are unique to `projectA` while others are shared by `projectA`, `projectB`, and `projectC`.

Like the VIPP image files, the VIPP segment files that are unique to projectA could be stored in a local scope folder such as `usr/xgfc/projects/projectA` while the VIPP segment files that are shared between projectA, projectB, and projectC could be stored in a folder scope folder such as `usr/xgfc/projects/fshared`.

Finally, you have the PostScript fonts and the font encoding resources to consider. Perhaps your company has established standards on the use of the PostScript fonts and font encoding. As a result, you only need one set of PostScript fonts and font encoding files for all projects to use. In that case, you could place your PostScript fonts and font encoding files in a global scope folder such as `usr/xgfc/gshared`.

In the section entitled VIPP INI Settings, you will see how you can define the folder name (“`$$FOLDER.`”) and project name (“`$$PROJECT.`”) used to represent the directories containing the VIPP resources required by the VIPP print streams produced from the Documaker VIPP print driver. You also see how to set up your own list of libraries containing VIPP resources.

Note All VIPP resource files stored on the VIPP console that are referenced by a Documaker VIPP job must have lower case file names. It is usually easier to make sure the resource file names are lower case before they are transferred to the UNIX workstation console attached to the VIPP printer.

SETTING VIPP INI OPTIONS

Here are the INI options and settings commonly-used with the VIPP print driver:

Option	Values	Description
Device	any file or device name	The name of the file or device (LPT1) where the VIPP print stream should be written. This setting is ignored by the GenPrint program but is used by Documaker Studio, Documaker Desktop, and other system programs. The default is the first three letters of the entry for the Module option, such as VPP.
Module	VPPW32	The name of the program module that contains the VIPP print driver. See also the Class option. The default is PCLW32, but you must enter VPPW32 to use the VIPP print driver.
PrintFunc	VPPPrint	The name of the program function that is the main entry point into the VIPP print driver. The default is PCLPrint, but you must enter VPPPrint . Case is important when using this option, therefore, you must enter it exactly as shown here: VPPPrint
Resolution	300	The dots per inch resolution of the printer that will receive the PostScript data stream. The default is zero (0) which tells the system to let the print driver to determine the resolution. The VIPP print driver defaults to 300 dpi.
SendOverlays	Yes/No	Set to Yes if you have created VIPP overlays (segments) for each FAP file.
CacheFiles	any number, zero or higher	Set to enable the caching of VIPP segments and images. The first x number of VIPP segments and images in the print job are cached. The default is zero (0).
CacheLogos	Yes/No	Set to enable the caching of VIPP images if CacheFiles is also enabled. The default is No.
DSCHeaderComment		Use to specify PostScript Document Structure Convention (DSC) comments you want added to the header portion of the generated VIPP print stream. You can include as many DSCHeaderComment options as are necessary. See <i>Adding DSC Comments</i> on page 272 for more information.
OverlayExt	any file extension	The file extension of the VIPP overlays (segments). The default is OVL.
PageNumbers	Yes/No	Set to Yes to enable form or form set page numbering. The default is No.

Some default settings are determined by the program performing the print operation. The defaults in this table refer to printing from the GenPrint program. The defaults when printing from other applications, such as Documaker Desktop, may differ.

Option	Values	Description
SendColor	Yes/No Enabled/ Disabled/ Hidden	Set to Yes to enable color printing. Enabled = Option appears in the Print window and is active (available to be checked). Disabled = Option appears in the Print window but is grayed out (not available to be checked). Hidden = Option does not appear in the Print window.
HighlightColor	Yes/No	Set to Yes to enable highlight color support. The default is No. If you set this option to Yes, you must also set the SendColor option to Yes.
DownloadFonts	Yes/No	Set to Yes to embed (download) PostScript fonts within the VIPP print stream. Set to No if you have loaded the PostScript fonts onto the VIPP controller. The default is Yes but you will get better performance if you set this option to No.
TemplateFields	Yes/No	Set to Yes to test print Xs in variable fields
Class	(first three characters of the Module option)	Specifies the printer classification, such as AFP, PCL, XER, PST, GDI, or VPP. If you omit this option, the system defaults to the first three letters from the Module option. Some internal functions expect a certain type of printer. For instance, all 2-up functions require an AFP printer. The internal functions check the Class option to make sure the correct printer is available before continuing.
SelectRecipients	Yes/No Enabled/ Disabled/ Hidden	This only applies to the Documaker Desktop. Enabled = Option appears in the Print window and is active (available to be checked). Disabled = Option appears in the Print window but is grayed out (not available to be checked). Hidden = Option does not appear in the Print window.
Tray# (where # is a number from 1 to 9)	Media string	Specifies a media string in the form of: <i>Media Type:Media Color:Media Weight</i> See <i>Setting up Paper Trays</i> on page 271 for more details.
Folder	Directory name	Name of the high level directory (folder) under which a project may appear. See <i>Setting Up Folders and Projects</i> on page 269 for more details.
Project	Directory Name	Name of the directory where local resources for a project will reside. See <i>Setting Up Folders and Projects</i> on page 269 for more details.

Some default settings are determined by the program performing the print operation. The defaults in this table refer to printing from the GenPrint program. The defaults when printing from other applications, such as Documaker Desktop, may differ.

Option	Values	Description
ProjectPath	Fully qualified directory path	Each ProjectPath setting defines a path that will be used to define a SETPPATH command that overrides the one found in the <i>xgfunix.run</i> file found on the VIPP controller. The path defined by the first ProjectPath setting will be the first directory searched for VIPP resources. If the resource is not found, the path defined by the second ProjectPath will be searched next (and so on). See <i>Overriding the List of Libraries for Projects</i> on page 270 for more information.

Some default settings are determined by the program performing the print operation. The defaults in this table refer to printing from the GenPrint program. The defaults when printing from other applications, such as Documaker Desktop, may differ.

SETTING UP FOLDERS AND PROJECTS

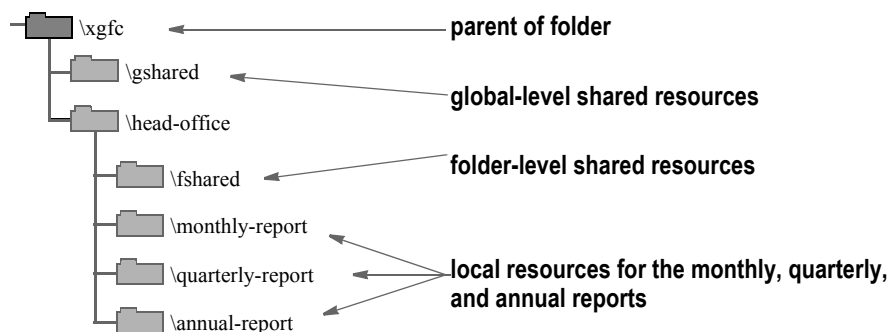
VIPP uses a configuration file named *xgfunix.run* (stored in the */usr/xgf/src* directory) to define a list of libraries (directories) for projects. In the *xgfunix.run* file, you might see a SETPPATH (VIPP command) that looks like this:

```
[ (/usr/xgfc/$$FOLDER./$$PROJECT./) % project local paths
  (/usr/xgfc/$$FOLDER./fshared/) % project folder shared paths
  (/usr/xgfc/gshared/) % project global shared paths
  (/usr/xgfc/fontlib/) % project access to font lib
  (/usr/xgf/encoding/) % project access to standard encoding
  (/usr/xgf/gshared/) % project global shared path
  (/opt/XRXnps/resources/ps/mislib/) % project access to DocuSP
resource list
] SETPPATH
```

SETPPATH is a VIPP command that defines a list of libraries (directories) for projects. The *\$\$FOLDER.* and *\$\$PROJECT.* strings are placeholders for project folders and project names.

You can use the projects directory for your main folder or create your folder directory. The name of the directory for your local project resources can be anything you wish.

Let's say you wanted to create a series of projects for the head office. Your VIPP projects repository might look like this:



Of course, you would need to create the *head-office* directory along with the subdirectories for the *fshared*, *monthly-report*, *quarterly-report*, and *annual-report* on the VIPP controller. And you would need to transfer the VIPP resource files (images, segments, fonts, and so on) into the appropriate directories.

However, before you can produce one of the reports for the head office, you will need to tell the Documaker VIPP print driver which VIPP folder and project names this report will use. You do this by specifying the Folder and Project options in your VIPP printer control group.

Option	Description
Folder	The Folder option contains the name of the high level directory (folder) under which a project may appear. The value set in the Folder option is substituted automatically as the \$\$FOLDER string in the SETPPATH statement found in the <i>xgfunix.run</i> file on the VIPP controller.
Project	The Project option contains the name of the directory where local resources for a project will reside. The value set in the Project option is substituted automatically as the \$\$PROJECT string in the SETPPATH statement found in the <i>xgfunix.run</i> file on the VIPP controller.

Using the example described earlier, let's say you want to produce a monthly report for the head office. In that case, you would use the following Folder and Project settings:

```
< PRTType:VPP >
  Folder = head-office
  Project= monthly-report
```

OVERRIDING THE LIST OF LIBRARIES FOR PROJECTS

As mentioned before, VIPP uses a configuration file called *xgfunix.run* (stored in the /usr/xgf/src directory) to define a list of libraries (directories) for projects.

In the *xgfunix.run* file, you might see a SETPPATH (VIPP command) that looks like this:

```
[ (/usr/xgfc/$$FOLDER./$$PROJECT./)    % project local paths
  (/usr/xgfc/$$FOLDER./fshared/)        % project folder shared paths
  (/usr/xgfc/gshared/)                  % project global shared paths
  (/usr/xgfc/fontlib/)                  % project access to font lib
  (/usr/xgf/encoding/)                  % project access to standard encoding
  (/usr/xgf/gshared/)                   % project global shared path
  (/opt/XXnps/resources/ps/mislib/)     % project access to DocuSP
resource list
] SETPPATH
```

If you wanted to override the list of project paths with a different set, you can do so by using a series of ProjectPath INI options. Each ProjectPath option defines a path that will be used to define a SETPPATH command that overrides the one found in the *xgfunix.run* file found on the VIPP controller. The path defined by the first ProjectPath option will be the first directory searched for VIPP resources. If the resource is not found, the path defined by the second ProjectPath will be searched next (and so on).

The following ProjectPath settings would produce the same list of paths as described earlier:

```
< PRTType:VPP >
  ProjectPath = /usr/xgfc/$$FOLDER./$$PROJECT./
  ProjectPath = /usr/xgfc/$$FOLDER./fshared/
  ProjectPath = /usr/xgfc/gshared/
  ProjectPath = /usr/xgfc/fontlib/
```

```
ProjectPath = /usr/xgf/encoding/  
ProjectPath = /usr/xgf/gshared/  
ProjectPath = /opt/XXnps/resources/ps/mislib/
```

When defining your own list of project paths, keep in mind:

- In the Local scope category, \$\$PROJECT must immediately follow \$\$FOLDER.
- A path containing \$\$PROJECT without \$\$FOLDER is not allowed.
- If present, \$\$FOLDER and \$\$PROJECT must appear only once in each path.
- No additional path components are allowed after \$\$PROJECT.
- A path ending by \$\$FOLDER is invalid.
- There must be at least one path for each category.
- There may be several paths in each category but they must be defined and grouped by category (local, folder, global) in the SETPPATH list.
- A folder or project name must appear only once in the trees of directories covered by SETPPATH.
- When a resource is present with the same name in more than one scope, the order of precedence is: local, folder, global.
- To improve cross-platform portability, Xerox recommends that FOLDER and PROJECT names do not contain more than 32 characters, and only use the characters “a” to “z”, “0” to “9”, “.” (dot), “-” (dash) and “_” (underscore).

SETTING UP PAPER TRAYS

The type of media (paper) stored in each paper tray needs to be defined in terms of its MediaType, MediaColor, and MediaWeight.

The MediaType can be named Plain, Transparency, Drilled, and so on

The MediaColor can be any color such as White, Green, Blue, GoldenRod, and so on

The MediaWeight is measured in grams per square meter. Usually, the media weight is set to 75 g/m² (equivalent to 20 lb. paper).

When designing your form set, you may have specified that certain forms use a specific paper tray to make sure the proper paper (pre-printed forms, colored paper, perforated paper, and so on) was used.

To make sure these forms print on the desired type of paper, you must define a unique MediaType, MediaColor, and MediaWeight combination for the paper tray. This information must be set up on both the printer and in the TRAY# INI settings in your VIPP printer control group.

For example, let's say that on your printer, you defined a type of paper will be stored in TRAY1 as having a MediaType of Plain, a MediaColor of Green, and a MediaWeight of 75 g/m².

For your form set to print from that paper tray, you would add the following INI option to your VIPP printer control group:

```
< PrtType:VPP >
```

```
Tray1 = Plain:Green:75
```

The Tray# INI settings expect a string in the form of:

```
MediaType:MediaColor:MediaWeight
```

You can specify any of the media attributes as null or omit them. When any of the media attributes are omitted or specified as null, those attributes are ignored in the following media selections. This example ignores *MediaType*.

```
Tray1 = null:Green:75
```

If the trailing media attributes are omitted, you can omit the trailing colon (:), as shown in this example:

```
Tray2 = Plain::
```

or

```
Tray2 = Plain:
```

or

```
Tray2 = Plain
```

When any of the media attributes such as type, color, or weight are omitted, the last specification or the default value for that attribute remains in effect. Because it may be difficult for you to know the value of the attribute that remains in effect, omitting or media attributes as null should be used with caution.

Finally, the TRAY# INI settings can also be specified with just a tray number from 1 to 9. For example, Tray5=1 maps output for tray 5 to tray 1. The system checks the INI option for overriding Tray1 before it checks the setting for Tray2 and so on.

Because of this, do not specify a tray number *less than* the tray you are overriding. For example, you should not use a setting of Tray5=6.

```
< PrtType:VPP >
  Tray1 = Plain:White:75
  Tray2 = Plain:Yellow:75
  Tray3 = Plain:Pink:75
  Tray4 = Drilled:White:75
  Tray5 = 1
  Tray6 = 1
```

ADDING DSC COMMENTS

For paper tray selection to work properly on DocuPrint NPS printers, it may be necessary to also include some DSC comments at the beginning of your VIPP print stream.

Use the DSCHeaderComment INI option to specify PostScript Document Structure Convention (DSC) comments you want added to the header portion of the generated VIPP print stream. You can include as many DSCHeaderComment options as are necessary.

This example shows how, in addition to specifying media commands in the Tray# options, you can also include DSC comments you want added to the header portion of the generated VIPP print stream:

```
< PrtType:VPP >
  DSCHeaderComment = %%DocumentMedia:Media1 612 792 75 (White)
  (Plain)
  DSCHeaderComment = %%+ Media2 612 792 75 (Yellow) (Plain)
  DSCHeaderComment = %%+ Media3 612 792 75 (Pink) (Plain)
```



```

DSCHeaderComment = %+ Media4 612 792 75 (White) (Drilled)
Tray1              = Plain:White:75
Tray2              = Plain:Yellow:75
Tray3              = Plain:Pink:75
Tray4              = Drilled:White:75

```

The form of the DocumentMedia DSC comment is:

% Key: <Tag Name> <Width> <Height> <Weight> <Color> <Type>

Item	Description
Tag Name	Any unique name, ignored by VIPP
Width	The width of paper stock, measured in 1/72" units
Height	The height of paper stock, measured in 1/72" units
Color	The color of paper stock. You can enter any alphanumeric string.
Type	The type of paper stock. You can enter any alphanumeric string.

The DSC header comments are added at the beginning of the generated VIPP print stream, as shown here:

```

%!
%%Title: INSURED
%%Creator: Documaker VIPP Driver
%%CreationDate: Wed Jul 13 11:55:34 2005

%%DocumentMedia:Media1 612 792 75 (White) (Plain)
%%+ Media2 612 792 75 (Yellow) (Plain)
%%+ Media3 612 792 75 (Pink) (Plain)
%%+ Media4 612 792 75 (White) (Drilled)
%%EndComments

```

VIPP LIMITATIONS

Here are some limitations of the VIPP language:

- The VIPP language does not support Unicode. As a result, the VIPP print driver can not be used as a Unicode print driver.

Here are some known problems with VIPP version 5.3:

- When caching is used in a VIPP print job, some VIPP segments and images may not print in the correct location or at all, or may cause a fatal system error on the printer. This is a known issue on some printers, such as older model DocuTech and DocuPrint printers. You can remove the CacheFiles INI option and reproduce your print job without using caching.

Or, you can open a console window on the printer's workstation, login with root access and type (or ask your Xerox analyst or engineer to do so):

```
/opt/XXnps/bin/setimagepath -f 0
```

This will disable VIPP caching for all print jobs.

- There is a VIPP bug when using a vector object to draw a circle and the line width exceeds a certain size (noticeable at 1/6 inch or higher). The outside edge of the circle does not draw completely around the border of the circle. The Xerox says it will be fixed in the next VIPP release (after version 5.3).
- There is a problem when using Univers Condensed Bold and Italic fonts on DocuPrint or DocuTech 65 printers. When printing a line of text using the Univers Condensed Bold font followed by a second line of text using the Univers Condensed Italic font, some of characters in the second line may print using the Univers Condensed Bold font (instead of the Univers Condensed Italic font). This bug reported to Xerox but will not be fixed.

Note The SPAR problem was analyzed by Xerox's VIPP and DocuSP development staffs who determined the problem lies in the Adobe PS decomposer. The problem was tested against the latest DT/DP75/90 product release and the fonts printed correctly, indicating the problem has been corrected by Adobe. Unfortunately, the DT65 is, according to Xerox, at its end of life and no further software support will be provided for this product.

VIPP TROUBLESHOOTING

Here are some troubleshooting scenarios:

Scenario 1

A VIPP job stops printing before the last page with the following error message:

```
ERROR: VIPP_unable_to_locate; OFFENDING COMMAND: filename.ext  
Flushing: rest of job (to end-of-file) will be ignored
```

Where *filename.ext* is the name of a VIPP resource file.

This error occurs if the VIPP print job references a VIPP resource file (PostScript font, font encoding table, VIPP segment overlay, VIPP bitmap image) that cannot be found.

Make sure you have loaded the missing file onto the VIPP controller and placed it in a folder defined for your VIPP project. See *Managing VIPP Resources* on page 264 for more information.

Scenario 2

A VIPP job stops printing before the last page, usually with the following error message:

```
ERROR: undefined  
OFFENDING COMMAND: Selected pages 0 n
```

Where *n* is the page volume limit for that device.

If VIPP is installed without a production license file, then the VIPP program will run in demonstration mode. Demonstration mode is a full-featured version of the VIPP software, however page volume limitations are imposed. The page volume limits are device-dependant and varies between 10 and 200 pages.

On some DocuColor printers, the error does not appear. Instead, jobs simply stop when the demonstration limit is reached. The limit is 57 or 200 pages and depends upon the DocuColor printer model.

Contact your Xerox representative about getting a VIPP license to run VIPP in full production mode.

Scenario 3

If you are not getting the correct characters printing, check the code page setting in the FXR file for the font. For most fonts that use the Windows code page, the code page setting in the font record should be set to 1004.

Chapter 14

Creating HTML Files

This chapter discusses creating HTML output. You can create HTML files by simply printing to the HTML print driver. The HTML print driver includes support for:

- Boxes – solid and shaded colors only
- Bar codes
- Charts
- Vectors – solid and shaded colors only
- Logos – converted to JPG files by the driver
- Lines – solid and shaded colors only. Dashed lines are supported but do not take the line characteristics as specified. The spacing and length of dashed lines are defaulted by the HTML 4 specification.
- Shaded areas – solid and shaded colors only
- Text areas
- Text
- Variable fields

For more information on creating HTML files, see these topics:

- *Setting Up the HTML Print Driver* on page 278
- *Creating Separate HTML Files for each Transaction* on page 281
- *Producing Table Information for TextMerge Paragraphs* on page 283

SETTING UP THE HTML PRINT DRIVER

Use these INI options to set up the HTML print driver:

```
< PrtType:HTML >
Device           = Sample.htm
DirLinks         = Yes
CollapsePage     = Yes
PageBreaks       = Yes
HR               = Size=2 Width=100% Color=Black
AllowInput       = No
DownloadFonts    = Yes, Enabled
TemplateFields   = No, Enabled
Module           = HTMW32
PrintFunc        = HTMLPrint
JavaScript       = format.js
JavaScript       = help.js
JavaScript       = data.js
ColorSheet       = iecolor.css
MultiPage        = No
ImagePath        = e:\rpex1
ImagePathCreate  = e:\rpex1\new_images
PageNumbers      = Yes
SendColor        = Yes, Enabled
AllowColorSheetLink = Yes
CreateScriptFile = Yes
DumpScript       = Yes
HiddenFieldScript = textMergeP(this);
ScriptPath       = e:\rpex1\deflib
ScriptPathCreate = e:\scripts
EntryBackColor   = #BBE6E6
EntryFontColor   = #FFFFFF
SplitText        = -1
BmSub            = Yes
BmSubChar        = '_'
IMG_ZIndex       = 100
```

Option	Description
Device	Sample.htm is the name of the HTML file generated when printed from Studio or Documaker Desktop.
DirLinks	Enter Yes to include Next and Prev links on pages. The default is No.
CollapsePage	Enter Yes to remove white space at the bottom of page. The default is No.
PageBreaks	Forces a page break between pages during HTML print. The default is Yes.
HR	(Header Rule) Displays a line between pages. You can configure the size, color and width. See example to see how to configure the rule.
AllowInput	Enter Yes if you are running iPPS or iDocumaker. The default is No.
DownloadFonts	Enter Yes to download of PCL fonts. Include <i>Enabled</i> to tell the system to display a check box the user can use when printing from Studio or Documaker Desktop.
TemplateFields	Enter No to omit the Xs from variable fields.
Module	Enter the name of the print driver DLL file.
PrintFunc	Enter the name of function within the print driver for print.

Option	Description
JavaScript	Specify any script files which contain functions you want the system to use.
ColorSheet	Specify the style sheet that contains the color information. The default is WSCOLOR.CSS. The WSCOLOR.CSS file contains web safe colors. The IECOLOR.CSS file contains Internet Explorer colors.
AllowColorSheetLink	Use this option to tell the HTML print driver if a color style sheet link exists. The default is Yes. If the color information is inline in the HTML page, omit the link to an external color style sheet.
MultiPage	Enter Yes only when running Studio and you want to print multiple page FAP files with each page as a separate HTML file. This feature is not supported when printing from Documaker and Documaker Desktop.
ImagePath	Enter the path for JPG files.
ImagePathCreate	Enter a path to indicate where you want the images created.
PageNumbers	Enter Yes to print page numbers. The default is No.
SendColor	Enter Yes,Enabled to print in color. The default is No.
CreateScriptFile	<p>Enter Yes is if you want the HTML driver to generate script files. The script files correspond to the name of the FAP. For example, if the Q1ADDR.FAP file contained either an inline script or a call to a function in an external script file, the generated script file is named Q1ADDR.JS.</p> <p>The corresponding function for a function that occurred in an external DAL script file would be called. For example:</p> <pre>_q1addr_dal_1ST NAME ZIP(obj)</pre> <p>where the <i>q1addr</i> represents the name of the FAP file, <i>_dal</i> implies it was contained in an external script file, and <i>_1ST NAME ZIP</i> is the name of the field. If the DAL script is inline and not in an external file, the example would look like this:</p> <pre>_q1addr_1ST NAME ZIP(obj) .</pre>
DumpScript	<p>Enter Yes to have the inline script written to the Java script file as a comment. For example if the inline script for an effective date field existed it would be written to the js file as:</p> <pre>function _samplefap_EFFDTE(obj) ; { /*HOLDDTE = DATEADD(@("EFFDTE"),,,,1); SETFLD (HOLDDTE, "EXPDTE");*/ } </pre> <p>If you enter No for this option, the inline script is not written as a comment. Instead, it is handled as if it were in an external file. The corresponding function would be:</p> <pre>_samplefap_EFFDTE(obj) ; { /*alert(EFFDTE code goes here);*/ }</pre>
HiddenFieldScript	<p>This tells the system to display a box the user can use to enter data onto the form. In this example,</p> <pre>textMergeP(this);</pre> <p>The hidden field has a length of one.</p>
ScriptPath	Enter the path to the script (.js) files generated by the HTML print driver. This is used for script files references in the HTML files.

Option	Description
ScriptPathCreate	Enter the path that points to where the driver creates the script files.
EntryBackColor	This is the background color for the data entry fields. This can be overridden in formatting script functions. The default is #B0E0E6.
EntryFontColor	Enter the color for the font used in data entry fields. The default is #FFD2D2.
SplitText	The default (-1) tells the system to split the text so a single word appears on each HTML line. Enter zero (0) to tell the system not to split text in the HTML file.
BmSub	Enter No if you do not want the system to substitute invalid characters with the character you specify in the BmSubChar option. The default is Yes. For instance, if you omit this option but specify X in the BmSubChar option, the system substitutes Xs for invalid characters.
BmSubChar	Enter the character you want the system to use to replace invalid characters. The default is an underscore (_).
IMG_ZIndex	The z-index indicates the stacking order of objects based on the order in which those objects appear in the HTML file. Higher values place objects closer to the front while lower values place them further to the back. Objects with the same value are stacked based on the order in which they appear in the HTML source. For instance, a positive value positions an object above text that has no defined z-index. A negative value would place the object below the same text. If you omit this option or leave it blank, the system will not layer objects.

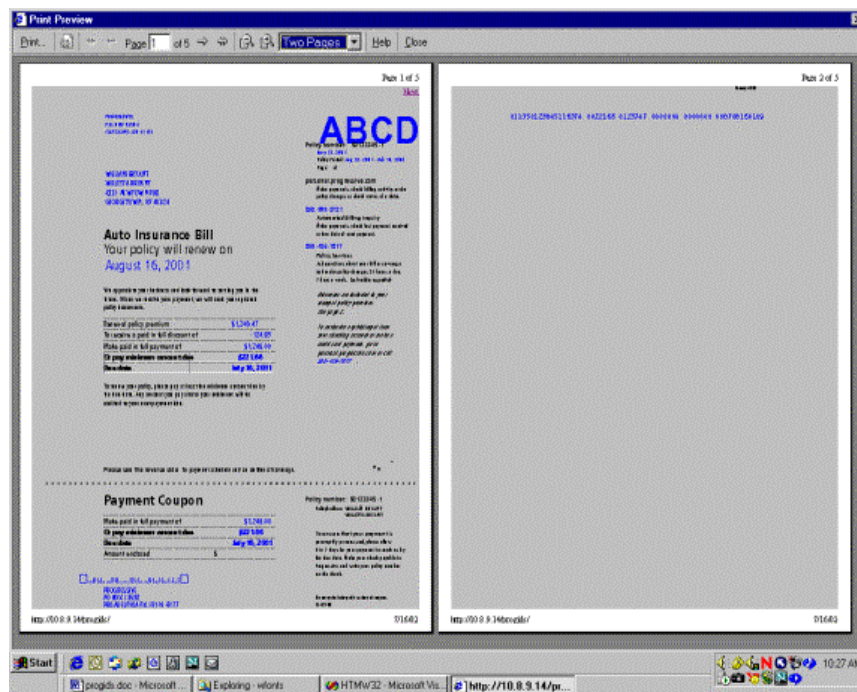
CREATING SEPARATE HTML FILES FOR EACH TRANSACTION

You can run the GenData program in single step mode and create separate files for each transaction. Each transaction will have its own HTML file if you include INI options similar to these:

```
< PrintFormSet >
    MultiFilePrint= Yes
    LogFileType    = XML
    LogFile        = d:\fap\mstrres\rpex1\data\jlog
< Printer >
    PrtType        = HTML
; Printers is for the GUI dropdown selection (uncomment many)...
< Printers >
    PrtType        = AFP
    PrtType        = PCL
    PrtType        = XER
    PrtType        = HTML
```

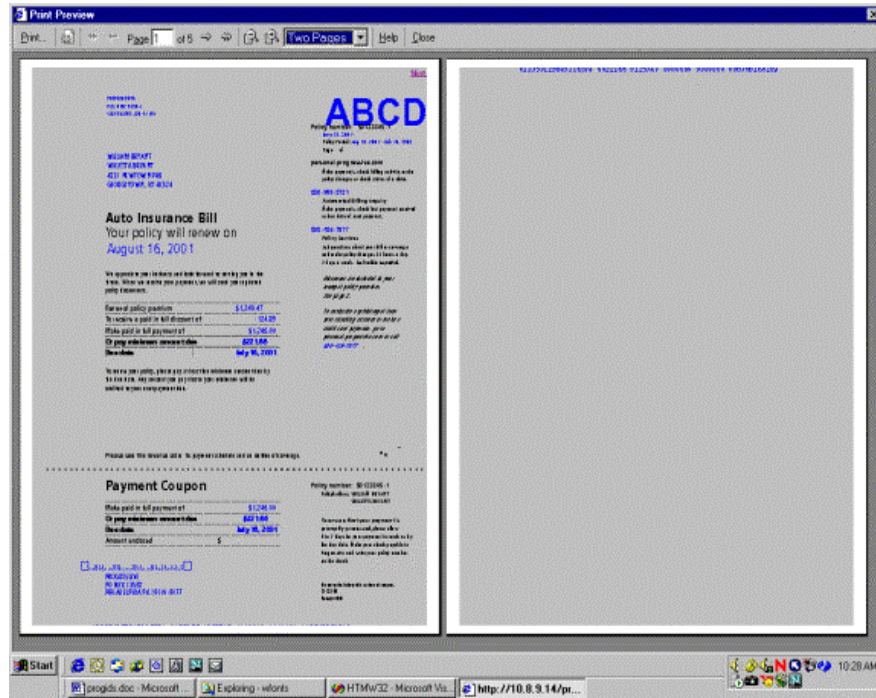
Note To make sure the page looks the same to all users, only use fonts which all users have. Otherwise, fonts are substituted.

The look of the printed HTML file depends on the settings in affect for the user's browser. For example, the following shows printed output when the page setup includes a header and footer with a top and bottom margin of 0.166":



Note how the last line of the first page has overflowed onto a new page.

The following example shows the printed output when the margins remain at 0.166" but there are no header and footers:



Note how the last line of the page has moved up but still has run over onto a second page.

Note The company logo appears as *ABCD* in this example. This can indicate a font problem. For instance, Internet Explorer can display HTML that uses raster fonts as long as the raster font is on the user's machine. Not all Windows print drivers, however, support printing with raster fonts. In this example, if this happened the output would print as *ABCD*.

PRODUCING TABLE INFORMATION FOR TEXTMERGE PARAGRAPHS

The system can produce table information for HTML documents generated from FAP documents which include text areas that have the TerSubstitute Pre-Edit procedure enabled.

The following attribute information is now produced for the hidden input tag that corresponds to an <iframe> element for a text merge paragraph:

```
attribType = tersub
TABLEID = ID value of the table.
TABLEFILE = The file value of the table.
```

Here is an example:

```
<IFRAME ID="PSELECTION" NAME="PSELECTION"
ONFOCUS="window.frames.editBar.displayToolbar(this,true);"
FRAMEBORDER="0" STYLE="BORDER-BOTTOM: buttonface 1px solid; BORDER-
LEFT: buttonface 1px solid; BORDER-RIGHT: button face 1px
solid;BORDER-TOP: button face 1px solid;height: 0.80 in; width: 6.10
in; " TABINDEX="3"></IFRAME>
<INPUT TYPE="hidden" NAME="PSELECTION" ACCESSKEY="G" TABLEID="Denial
Reasons" TABLEFILE="LTRENTY" attribType="tersub"/>
```

You can use this information to call the DPRTblLookUp rule in another request to retrieve a list of paragraphs for insertion into the text merge area. This information can in turn be used to call the DPRFap2Html rule to retrieve an HTML representation of the appropriate TerSub paragraph so it can be inserted into the <iframe> element.

This lets iPPS query real-time TERSUB information from IDS.

Appendix A

Choosing a Paper Size

The system supports a variety of paper sizes including US and international sizes. The following tables show the paper sizes you can choose from:

- *US Standard Sizes* on page 287
- *ISO Sizes* on page 288
- *Japanese Standard Sizes* on page 291

You can also find the following related information in this topic:

- *Printer Support for Paper Sizes* on page 292
- *Paper Sizes for AFP Printers* on page 295

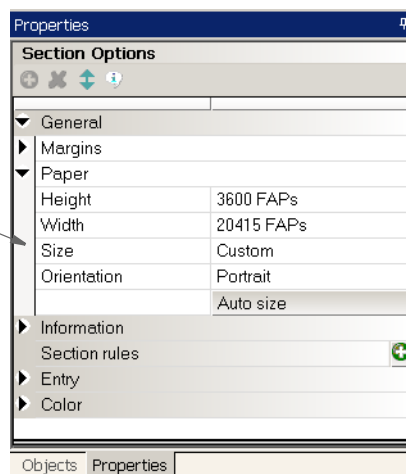
Note Please note that the NA file stores the actual section height and width for custom sized sections. This information is stored in the SIZE entry in the NAFILE.DAT file. Here is an example:

```
\NA=q1snam, LN=1, DUP=LB, SIZE=3360x18600, TRAY=U, X=600, Y=600 . . .
```

The height and width are in FAP units (2400 per inch).

In Studio you use the Size property to specify the page size for a section. There is also a Size property at the form level.

For a section, you can choose from the available standard page sizes or choose Custom here.



The screenshot shows the 'Properties' dialog box with the 'Section Options' tab selected. The 'Paper' section is expanded, showing the following properties:

Property	Value
Height	3600 FAPs
Width	20415 FAPs
Size	Custom
Orientation	Portrait
Auto size	Auto size

The 'Information' section is also expanded, showing the following properties:

Property	Value
Section rules	
Entry	
Color	

If, for a section, you choose Custom, the system defaults to the size of paper that will best contain the custom section, but you must tell it what paper is installed on your printer. For sections small enough to fit on letter size paper, the system defaults to letter.

Note	This affects section printing from Documaker Studio but has no effect on form (FOR) definitions.
-------------	--

US STANDARD SIZES

These paper sizes are commonly used in the United States and Canada. The height and width are in FAP units (2400 per inch), millimeters, and inches. The inch dimensions are approximate.

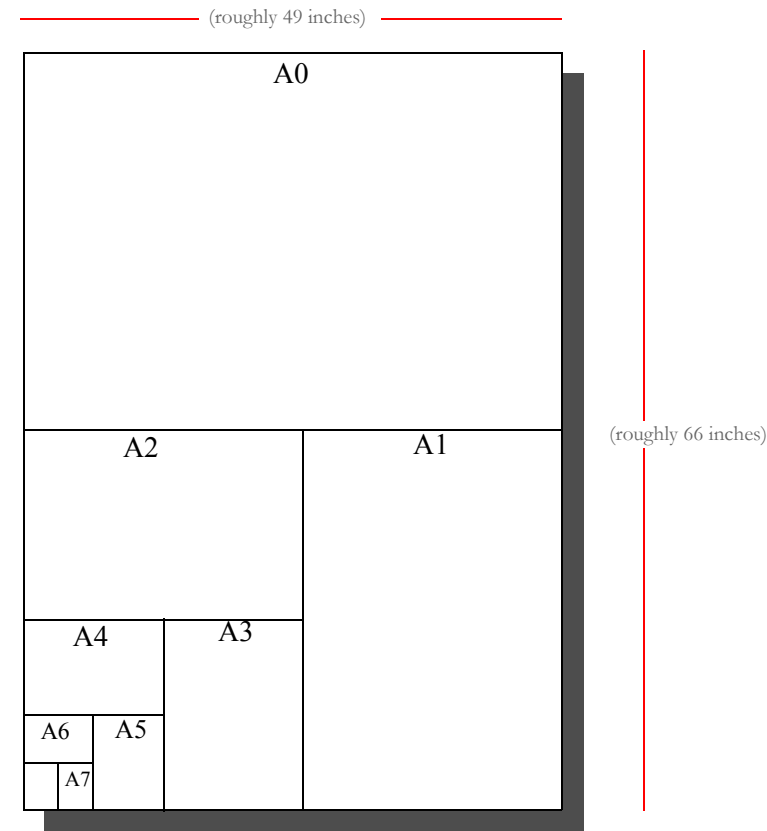
Name	Code	Width x Height		
		FAP units	Millimeters	Inches (approximate)
US letter	0	20400 x 26400	216 × 279	8½ x 11
US legal	1	20400 x 33600	216 × 356	8½ x 14
US executive	3	17400 x 25200	190 × 254	7¼ 10½
US ledger	4	40800 x 26400	432 x 279	17 x 11
US tabloid	5	26400 x 40800	279 × 432	11 x 17
US statement	6	13200 x 20400	140 x 216	5½ x 8½
US folio	7	20400 x 31200	216 x 330	8½ x 13
US fanfold	8	35700 x 26400	378 x 279	14⅞ x 11
Custom	98	any x any	any x any	any x any

ISO SIZES

The International Organization for Standardization (ISO) paper sizes, which are based on the earlier Deutsche Industrie Norm (DIN) sizes, are used throughout the world except in Canada, the United States, and Japan. There are three main series of paper sizes: A, B, and C.

ISO A Sizes

The A series of sizes are typically used for correspondence, books, brochures, and other printed materials. This diagram shows most of the various A sizes. The height and width are in FAP units (2400 per inch), millimeters, and inches. The inch dimensions are approximate.



Width x Height				
Name	Code	FAP units	Millimeters	Inches (approximate)
ISO A0	20	79464 x 112345	841 x 1189	33 ¹ / ₈ x 46 ¹ / ₄
ISO A1	21	56125 x 79464	594 x 841	23 ³ / ₈ x 33 ¹ / ₈
ISO A2	22	39685 x 56125	420 x 594	16 ¹ / ₂ x 23 ³ / ₈
ISO A3	23	28063 x 39685	297 x 420	11 ³ / ₄ x 16 ¹ / ₂

Name	Code	Width x Height		
		FAP units	Millimeters	Inches (approximate)
ISO A4	2	19842 x 28063	210 x 297	8¼ x 11¾
ISO A5	25	13984 x 19842	148 x 210	5⅞ x 8¼
ISO A6	26	9921 x 13984	105 x 148	4⅛ x 5⅞
ISO A7	27	6992 x 9921	74 x 105	2⅞ x 4⅛
ISO A8	28	4913 x 6992	52 x 74	2 x 2⅞
ISO A9	29	3496 x 4913	37 x 52	1½ x 2
ISO A10	30	2457 x 3496	26 x 37	1 x 1½
ISO 2A	32	112345 x 158927	1189 x 1682	46¾ x 66¼
ISO 4A	34	158927 x 224690	1682 x 2378	66¼ x 93⅝

ISO B Sizes

The B series of sizes are designed primarily for posters, wall charts, and similar items where the difference between each A size represents too large a jump. The height and width are in FAP units (2400 per inch), millimeters, and inches. The inch dimensions are approximate.

Name	Code	Width x Height		
		FAP units	Millimeters	Inches (approximate)
ISO B0	40	94487 x 133605	1000 x 1414	39⅞ x 55⅞
ISO B1	41	66802 x 94487	707 x 1000	27⅞ x 39⅞
ISO B2	42	47244 x 66802	500 x 707	19⅝ x 27⅞
ISO B3	43	33354 x 47244	353 x 500	13⅞ x 19⅝
ISO B4	44	23622 x 33354	250 x 353	9⅞ x 13⅞
ISO B5	45	16630 x 23622	176 x 250	7 x 9⅞
ISO B6	46	11811 x 16630	125 x 176	5 x 7
ISO B7	47	8315 x 11811	88 x 125	3½ x 5
ISO B8	48	5858 x 8315	62 x 88	2½ x 3½
ISO B9	49	4157 x 5858	44 x 62	1¾ x 2½
ISO B10	50	2929 x 4157	31 x 44	1¼ x 1¾
ISO 2B	52	133605 x 188974	1414 x 2000	55¾ x 78¾
ISO 4B	54	188974 x 267209	2000 x 2828	78¾ x 111¼

ISO C Sizes

The C series of sizes are designed for making envelopes and folders to take the A series of sizes. The height and width are in FAP units (2400 per inch), millimeters, and inches. The inch dimensions are approximate.

Name	Code	Width x Height		
		FAP units	Millimeters	Inches (approximate)
ISO C0	60	86645 x 122550	917 x 1297	36 $\frac{1}{8}$ x 51
ISO C1	61	61228 x 86645	648 x 917	25 $\frac{1}{2}$ x 36
ISO C2	62	43275 x 61228	458 x 648	18 x 25 $\frac{1}{2}$
ISO C3	63	30614 x 43275	324 x 458	12 $\frac{3}{4}$ x 18
ISO C4	64	21638 x 30614	229 x 324	9 x 12 $\frac{3}{4}$
ISO C5	65	15307 x 21638	162 x 229	6 $\frac{3}{8}$ x 9
ISO C6	66	10772 x 15307	114 x 162	4 $\frac{1}{2}$ x 6 $\frac{3}{8}$
ISO C7	67	7653 x 10772	81 x 114	3 $\frac{1}{4}$ x 4 $\frac{1}{2}$
ISO C8	68	5386 x 7653	57 x 81	2 $\frac{1}{4}$ x 3 $\frac{1}{4}$
ISO C9	69	3779 x 5386	40 x 57	1 $\frac{5}{8}$ x 2 $\frac{1}{4}$
ISO C10	70	2646 x 3779	28 x 40	1 $\frac{1}{8}$ x 1 $\frac{5}{8}$
ISO DL	71	10394 x 20787	110 x 220	4 $\frac{1}{3}$ x 8 $\frac{2}{3}$

The DL size is for a sheet 1/3 of the A4 size. This is the most common size of envelope.

JAPANESE STANDARD SIZES

Japan has its own standard paper sizes, called the Japan Industrial Standard (JIS). The JIS A series is identical in size to the ISO A series. The JIS B series, however, does not match the ISO B series. There is no equivalent to the ISO C series. This table shows the JIS paper sizes. The height and width are in JAP units (2400 per inch), millimeters, and inches. The inch dimensions are approximate.

Name	Code	Width x Height		
		JAP units	Millimeters	Inches (approximate)
JIS B0	80	97322 x 137573	1030 x 1456	40½ x 57¼
JIS B1	81	68787 x 97322	728 x 1030	28¾ x 40½
JIS B2	82	48661 x 68787	515 x 728	20¼ x 28¾
JIS B3	83	34393 x 48661	364 x 515	14¼ x 20¼
JIS B4	84	24283 x 34393	257 x 364	10⅞ x 14¼
JIS B5	85	17197 x 24283	182 x 257	7¼ x 10⅞
JIS B6	86	12094 x 17197	128 x 182	5 x 7¼
JIS B7	87	8598 x 12094	91 x 128	3½ x 5
JIS B8	88	6047 x 8598	64 x 91	2½ x 3½
JIS B	89	4252 x 6047	45 x 64	1¾ x 2½
JIS B10	90	3024 x 4252	32 x 45	1¼ x 1¾

PRINTER SUPPORT FOR PAPER SIZES

This table outlines the various paper sizes supported by the different print drivers. The table includes information for the PDF, RTF, HTML, Metacode, PCL 5, PCL 6, GDI, PostScript, and AFP print drivers. The PDF, RTF, HTML, and Metacode print drivers support all paper sizes.

Paper size	PDF, RTF, HTML, and Metacode	PXL ¹	PCL ²	GDI ²	PST ³	AFP ⁴
US letter	X	X	X	X	X	X
US Legal	X	X	X	X	X	X
US executive	X	X	X	X	X	X
US ledger	X	X	X	X	X	X
US tabloid	X	Y	US letter	X	X	X
US statement	X	JIS B5	US executive	X	X	X
US folio	X	US legal	US legal	X	X	X
US fanfold	X	US ledger	US ledger	X	X	X
ISO 4A	X	Y	US letter	US letter	US letter	C
ISO 2A	X	Y	US letter	US letter	US letter	C
ISO A0	X	Y	US letter	US letter	X	C
ISO A1	X	Y	US letter	US letter	X	C
ISO A2	X	Y	US letter	US letter	X	C
ISO A3	X	X	X	X	X	X
ISO A4	X	X	X	X	X	X
ISO A5	X	X	X	X	X	X
ISO A6	X	X	X	X	X	X

Sizes marked with an X are fully supported by the corresponding driver.

Sizes marked with a Y are supported by sending the paper dimensions in millimeters to the printer.

Sizes that refer to another size substitute the referred size when *paper size matching* is turned on. If paper size matching is not turned on, the behavior depends upon the specific driver. To turn on paper size matching, use this INI option:

```
< PRTType:XXX >
  PaperSizeMatching = Yes
```

¹ When paper size matching is not turned on, the PCL 6 (PXL) driver sends the paper dimensions in millimeters to the printer.

² When paper size matching is not turned on, these drivers substitute US letter.

³ This driver does not use paper size matching. US letter is substituted for the unsupported paper sizes

⁴ Sizes marked with a C are supported, but are commented out of the AFP formdef source file called F1FMMST.DAT. See *Paper Sizes for AFP Printers* on page 295 for more information.

Paper size	PDF, RTF, HTML, and Metacode	PXL ¹	PCL ²	GDI ²	PST ³	AFP ⁴
ISO A7	X	ISO A6	ISO C5	ISO A6	X	C
ISO A8	X	ISO A6	ISO C5	ISO A6	X	C
ISO A9	X	ISO A6	ISO C5	ISO A6	X	C
ISO A10	X	ISO A6	ISO C5	ISO A6	X	C
ISO 4B	X	Y	US letter	US letter	US letter	C
ISO 2B	X	Y	US letter	US letter	US letter	C
ISO B0	X	Y	US letter	US letter	X	C
ISO B1	X	Y	US letter	US letter	X	C
ISO B2	X	Y	US letter	US letter	X	C
ISO B3	X	Y	US letter	US letter	X	C
ISO B4	X	JIS B4	US ledger	X	X	X
ISO B5	X	JIS B5	X	X	X	X
ISO B6	X	JIS B6	ISO C5	X	X	X
ISO B7	X	ISO A6	ISO C5	ISO A6	X	C
ISO B8	X	ISO A6	ISO C5	ISO A6	X	C
ISO B9	X	ISO A6	ISO C5	ISO A6	X	C
ISO B10	X	ISO A6	ISO C5	ISO A6	X	C
ISO C0	X	Y	US letter	US letter	X	C
ISO C1	X	Y	US letter	US letter	X	C
ISO C2	X	Y	US letter	US letter	X	C
ISO C3	X	Y	US letter	X	X	C
ISO C4	X	JIS B4	US ledger	X	X	C

Sizes marked with an X are fully supported by the corresponding driver.

Sizes marked with a Y are supported by sending the paper dimensions in millimeters to the printer.

Sizes that refer to another size substitute the referred size when *paper size matching* is turned on. If paper size matching is not turned on, the behavior depends upon the specific driver. To turn on paper size matching, use this INI option:

```
< PrtType:XXX >
    PaperSizeMatching = Yes
```

¹ When paper size matching is not turned on, the PCL 6 (PXL) driver sends the paper dimensions in millimeters to the printer.

² When paper size matching is not turned on, these drivers substitute US letter.

³ This driver does not use paper size matching. US letter is substituted for the unsupported paper sizes

⁴ Sizes marked with a C are supported, but are commented out of the AFP formdef source file called F1FMMST.DAT, See *Paper Sizes for AFP Printers* on page 295 for more information.

Paper size	PDF, RTF, HTML, and Metacode	PXL ¹	PCL ²	GDI ²	PST ³	AFP ⁴
ISO C5	X	X	X	X	X	C
ISO C6	X	JIS B6	ISO C5	X	X	C
ISO C7	X	ISO A6	ISO C5	ISO A6	X	C
ISO C8	X	ISO A6	ISO C5	ISO A6	US letter	C
ISO C9	X	ISO A6	ISO C5	ISO A6	US letter	C
ISO C10	X	ISO A6	ISO C5	ISO A6	US letter	C
ISO DL	X	X	X	X	X	X
JIS B0	X	Y	US letter	US letter	X	C
JIS B1	X	Y	US letter	US letter	X	C
JIS B2	X	Y	US letter	US letter	X	C
JIS B3	X	Y	US letter	US letter	X	C
JIS B4	X	X	X	US fanfold	X	X
JIS B5	X	X	X	X	X	X
JIS B6	X	X	X	X	X	X
JIS B7	X	ISO A6	ISO C5	ISO A6	X	C
JIS B8	X	ISO A6	ISO C5	ISO A6	X	C
JIS B9	X	ISO A6	ISO C5	ISO A6	X	C
JIS B10	X	ISO A6	ISO C5	ISO A6	X	C

Sizes marked with an X are fully supported by the corresponding driver.

Sizes marked with a Y are supported by sending the paper dimensions in millimeters to the printer.

Sizes that refer to another size substitute the referred size when *paper size matching* is turned on. If paper size matching is not turned on, the behavior depends upon the specific driver. To turn on paper size matching, use this INI option:

```
< PRTType:XXX >
    PaperSizeMatching = Yes
```

¹ When paper size matching is not turned on, the PCL 6 (PXL) driver sends the paper dimensions in millimeters to the printer.

² When paper size matching is not turned on, these drivers substitute US letter.

³ This driver does not use paper size matching. US letter is substituted for the unsupported paper sizes

⁴ Sizes marked with a C are supported, but are commented out of the AFP formdef source file called F1FMMST.DAT, See *Paper Sizes for AFP Printers* on page 295 for more information.

PAPER SIZES FOR AFP PRINTERS

The AFP formdef source file (F1FMMST.DAT) contains support for the following paper sizes, but since this file contains support for so many paper sizes, its size could affect printer performance. To limit the effect, some of the paper sizes are commented out, as shown in this table:

Size	Commented out?
Letter	No
Legal	No
Executive	No
Ledger	Yes
Tabloid	Yes
Statement	Yes
Folio	Yes
Fanfold	Yes
ISO A3	Yes
ISO A4	No
ISO A5	Yes
ISO A6	Yes
ISO B4	Yes
ISO B5	Yes
ISO B6	Yes
ISO DL	Yes
JIS B4	Yes
JIS B5	Yes
JIS B6	Yes

Note The F1FMMST.DAT and F1FMMST.FDF files can be found in the FMRES master resource library (MRL).

The commented source line begins with an asterisk (*). To add support for another paper size, you open the F1FMMST.DAT file and delete the asterisk at the beginning of each line that references the paper size you want to add.

Because the AFP formdef is composed on medium map names that specify page orientation, paper size, tray selection, and duplex settings, there are 31 groups of medium map settings. Each of these groups contains the 57 possible paper sizes. So, for each paper size you add, there are 31 sources lines you must *uncomment* to fully support a paper size for all orientations, trays, and duplex settings.

After you uncomment the lines that reference the paper size you want to add, run the AFPFMDEF utility to rebuild your AFP formdef file with the new information. For more information on this utility, see the [Utilities Reference](#).

Appendix B

Miscellaneous Print Topics

This appendix contains the following print-related topics:

- *Using Pass-through Printing* on page 298
- *Printing with Missing Graphics* on page 300
- *Creating Print Streams for Docusave* on page 301
- *Adding TLE Records* on page 305
- *Handling Multiple Paper Trays* on page 306
- *Spot Color Support* on page 310

USING PASS-THROUGH PRINTING

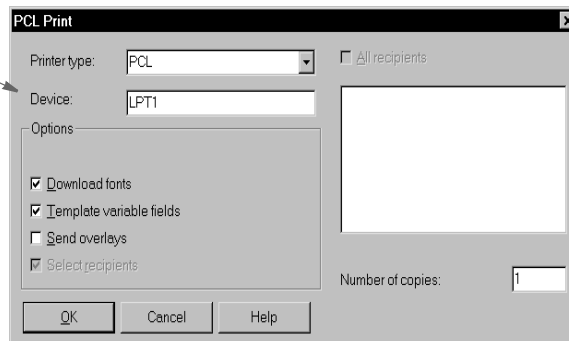
There are some problems which occur when you print to LPT1 on Windows platforms. One problem occurs if you run Netware Client 32 for Windows 95. Although you can open LPT1 from the system, you may receive errors when printing large amounts of data. Downloading PCL fonts usually causes this.

Another problem occurs when a print queue adds additional printer commands to system-created print jobs. This causes invalid output to be sent to the printer. The HP 5si print driver can cause this problem. Another problem affects other software which redirects printers and expects all print output to use the Windows GDI mechanism.

Documaker's GDI print driver uses Windows-native calls for printing, which is how most applications print under Windows. However, the Windows system print drivers have problems handling some system printing requirements, such as enhanced font selection, the ability to combine duplexing with landscape forms, and so on.

To solve these problems, Documaker print drivers can produce the commands for controlling the printer while still using an installed Windows printer device. To use this feature, leave the Device field blank on the Print window, where you select the printer driver you want to use.

Leave this field blank



Normally, the Device field contains the name of the device (LPT1) or the name of the file (D:\OUTPUT.PCL) the system should print to. When you leave this field blank, you tell the print driver you want to print through an installed printer device. After you click Ok, the Windows Print window appears so you can select which printer device to send the print job.

This printer device must be associated with a printer supported by the system's print driver. If you have a printer device available that is associated with a printer not supported by the system's print driver, the results are unpredictable. For example, if you select PCL as the system printer type (print driver), but choose a printer device associated with an AFP printer, the AFP printer will not understand the PCL output and will print garbage.

Unlike the GDI driver, Documaker print drivers control the printed output. The Windows Print window is the standard print window provided by Windows. Documaker applications cannot control or change this window. In addition, since a Documaker print driver is controlling the printer, most of the options on the Windows Print window will be ignored. The only options you can use are:

- Select a printer device.
- Select the Cancel button and the print process is canceled.
- Check the Print to File field and the system will print the document to the file you specify.

Note Not all Windows print drivers support pass-through printing. If you receive an error while printing in this manner, you are probably using a Windows print driver that does not support pass-through printing.

PRINTING WITH MISSING GRAPHICS

If you archive a form set which includes referenced external graphics and those graphics become unavailable, the system will not retrieve or print that form set. By default, if a graphic referenced by a form is unavailable, the system issues the following message and does not generate the print stream:

```
FAPLoadGlobBitmap <0> <0> Cannot load bitmap on this or later pages  
bitmap_name
```

To suppress this message and tell the system to generate the print stream without the graphic, add this option to your INI file:

```
< Control >  
    IgnoreExternalBitmapError = Yes
```

Option	Description
IgnoreExternalBitmapError	Enter Yes to tell the system to suppress and ignore errors normally generated when an external graphics file is missing. The default is No

Note You would typically only use this option if the rest of the archived form set is still valid and you need to retrieve and view or print it.

CREATING PRINT STREAMS FOR DOCUSAVE

Docusave can archive AFP, Metacode, and PCL print streams that are in a Docusave-compatible format and contain special records used to index the archive.

For AFP and Metacode, you use the OutMode option in the PrtType:AFP or XER control group to tell the GenPrint program to create a Metacode or AFP print stream in a Docusave-compatible record format. You can choose between these Docusave-compatible formats: JES2 and MRG4.

For PCL, the process is similar but there is not OutMode option to set. You include comment records in the print streams to index the archive. You can use a DAL script to add those comment records.

For details, see...

- *Archiving AFP Print Streams* on page 301
- *Archiving Metacode Print Streams* on page 302
- *Archiving PCL Print Streams* on page 303

ARCHIVING AFP PRINT STREAMS

Set the OutMode option to MRG4 to produce a print stream for Docusave from non-z/OS platforms.

Here is an example:

```
< PrtType:AFP >
  OutMode = MRG4
```

When you set the OutMode option to MRG4, the GenPrint program creates print stream records with a 4-byte sequence that precedes them. This sequence defines the record lengths. Records are grouped into blocks with one or more records in each block. Both records and blocks have a 4-byte sequence that precedes them, defining their length.

These length indicators are formed by taking the high-order byte of length followed by the low-order byte of length followed by two bytes of zeros.

The maximum number that can be displayed is a 16-bit quantity. The value in each includes the length of the structure itself. A one-byte data record in its own block would have five for the record length and nine for the block length. This table shows what a 3-byte record would look like:

Byte offset	Value (Hex)	Meaning
0	00	Block length high-order
1	0B	Block length low-order
2	00	Always 0
3	00	Always 0
4	00	Record length high-order

Byte offset	Value (Hex)	Meaning
5	07	Record length low-order
6	00	Always 0
7	00	Always 0
8	31	'1'
9	32	'2'
10	33	'3'

In addition to using the OutMode option, you must include comment records in the print streams to index the archive. You can use a DAL script to add comment records into the print stream. Use the DocusaveScript option in the PrtType:AFP control group to have the system execute a DAL script at the times when Docusave comments can be added to the print streams.

To add Docusave comments to an AFP print stream, you must add the DocusaveScript option and the name of a DAL script to execute. The DAL script should call the AddDocusaveComment function to add a string as a Docusave comment record. Here is an example:

```
< PrtType:AFP >
  DocusaveScript = Docusave.DAL
  OutMode = MRG4
```

ARCHIVING METACODE PRINT STREAMS

Set the OutMode option to JES2 to produce print streams under z/OS. Here is an example:

```
< PrtType:XER >
  OutMode = JES2
```

When you set the OutMode option to JES2, the GenPrint program creates print stream records that are native to a mainframe environment.

Also include comment records in the print streams to index the archive. You can use a DAL script to add comment records into the print stream. Use the DocusaveScript option in the PrtType:XER control group to have the system execute a DAL script at the times when Docusave comments can be added to the print streams.

To add Docusave comments to a Metacode AFP print stream, add the DocusaveScript option and the name of a DAL script to execute. The DAL script should call the AddDocusaveComment function to add a string as a Docusave comment record. Here is an example:

```
< PrtType:XER >
  DocusaveScript = Docusave.DAL
  OutMode = JES2
```

ARCHIVING PCL PRINT STREAMS

Note Docusave supports the archiving of PCL 5 print streams using the DSPJBpcl and DSPDCPCL processing libraries. Docusave expects one document per job. The index information in the PCL is EBCDIC text inside a PCL graphic comment. You can only store the PCL, no transformations.

Documaker version 10.2 and later can produce PCL 5 print streams with the necessary Docusave comment information.

You must include comment records in the print streams to index the archive. You can use a DAL script to add comment records into the print stream. Use the DocusaveScript option in the PrtType:PCL control group to have the system execute a DAL script when Docusave comments can be added to the print stream.

To add Docusave comments to an PCL print stream, add the DocusaveScript option and the name of a DAL script to execute. The DAL script should call the AddDocusaveComment function to add a string as a Docusave comment record.

Here is an example:

```
< PrtType:PCL >
  DocusaveScript = DOCUSAVE.DAL
```

Here is an example of what the DOCUSAVE.DAL file might look like:

```
* Add DocuSave Comment - use default: APPIDX record!
COMMENT = AppIdxRec()
PRINT_IT(COMMENT)
ADDDOCUSAVECOMMENT(COMMENT)
RETURN('FINISHED!')
```

Note PCL 6 print streams cannot be archived into Docusave.

USING DAL FUNCTIONS

For all types of print streams, you can use these DAL functions to create archive keys to use with Docusave.

Function	Description
AddDocusaveComment	Adds a Docusave comment string to the print stream
AddComent	Adds a comment string to the print stream
AppIdxRec	Gets an archive record based on APPIDX.DFD and Trigger2Archive INI settings
HaveGVM	Verifies if a GVM variable exists
SetGVM	Updates the contents of a GVM variable
GVM	Gets the contents of a GVM variable
MajorVersion	Gets the system's major version number
MinorVersion	Gets the system's minor version number
PrinterClass	Gets the type of print being produced

For more information on these functions, see the DAL Reference.

Function	Description
PrinterGroup	Gets the name of the print group being used
Print_It	Debug tool to print a string to the console

For more information on these functions, see the DAL Reference.

ADDING TLE RECORDS

You can add TLE (Tag Logical Element) records into AFP print streams which can be used by some 3rd-party archive systems to archive AFP print streams in a manner similar to archiving AFP or Metacode print streams in Docusave.

You must include comment records in the print streams to index the archive. You can use a DAL script to add comment records into the print stream. Use the TLEScript option in the PrtType:AFP control group to name the DAL script to execute when TLE records can be added into the print stream. The DAL script should call the AddComment function to add a string as a TLE comment record.

The TLE comment string must include a key and a value. Separate these components with a special character. This character can be any printable character as long as it is a unique character not found in the key or value portion of the comment string.

For example, you might build a comment string using a colon (:) as a separator as in the following example:

```
PolicyNum:7SAMPCO
```

The key portion of the string is *PolicyNum*, the value portion of the string is *7SAMPCO*, and the separator character is a colon (:).

Here is an example of what TLE DAL script might look like:

```
cidlabel = 'PolicyNum'
clientid = GVM("PolicyNum")
colon = ':'
AddComment (cidlabel & colon & clientid);
RETURN('FINISHED!')
```

Notice that the key portion remains constant (PolicyNum) while the value portion changes based on the contents of the GVM variable, PolicyNum.

Add these options to the PrtType:AFP control group to enable TLE record support:

```
< PrtType:AFP >
  TLEScript      = TLE.DAL
  TLEEveryPage   = No
  TLESeparator   = :
```

Option	Description
TLEScript	Enter the name of the DAL script to execute.
TLESeparator	Enter the character you want to use to separate the key and value portions of the TLE comment string.
TLEEveryPage	(Optional) If you enter Yes, the TLE DAL script will be executed at the start of every page. If you enter No, the TLE DAL script will be executed at the start of every form set. The default is No.

HANDLING MULTIPLE PAPER TRAYS

You can set up PCL, PostScript, GDI, AFP, and Metacode print drivers to support up to nine paper trays. Setting up nine tray printer support for the various types of printers is outlined below.

Note You can also use Documaker Studio to specify tray settings. See the [Documaker Studio User Guide](#) for more information.

For PCL Printers

You can override PCL tray commands by providing an alternate PCL command to use. Here are the default PCL INI settings:

```
< PrtType:PCL >
  Tray1 = ~&l1H
  Tray2 = ~&l4H
  Tray3 = ~&l5H
  Tray4 = ~&l20H
  Tray5 = ~&l21H
  Tray6 = ~&l22H
  Tray7 = ~&l23H
  Tray8 = ~&l24H
  Tray9 = ~&l25H
```

When writing PCL commands as an INI setting, the tilde (~) is used as a substitute for the PCL escape character (x1B).

For PostScript Printers

You can override PostScript tray commands by providing an alternate PostScript command to use. You issue PostScript tray commands in these forms:

- A quoted string containing the PostScript commands. The quoted string should contain the appropriate PostScript commands for selecting a paper tray. Here is an example:

```
Tray1 = "statusdict /lettertray get exec"
```

- A tray number from 1 to 9. You can use tray numbers to map non-existent trays. For example, Tray5=1 maps output for tray 5 to tray 1. The system checks the INI setting for overriding Tray1 before it checks the setting for Tray2 and so on. Because of this, do not specify a tray number *less than* the tray you are overriding. For example, you should not use a setting of Tray5=6.
- A UI keyword from a PPD file. UI keywords represent features that commonly appear in a user interface (UI). They provide the code to invoke a user-selectable feature within the context of a print job, such as the selection of an input tray or manual feed. The entries of UI keywords are surrounded by these structure keywords:

```
*OpenUI/*CloseUI or *JCLOpenUI/*JCLCloseUI
```

Here is an example of an OpenUI structure for MediaColor:

```
*OpenUI *MediaColor: PickOne
*OrderDependency: 30 AnySetup *MediaColor
*DefaultMediaColor: white
*MediaColor white: "1 dict dup /MediaColor (white) put setpagedevice"
*MediaColor clear: "1 dict dup /MediaColor (clear) put setpagedevice"
*MediaColor blue: "1 dict dup /MediaColor (blue) put setpagedevice"
*MediaColor buff: "1 dict dup /MediaColor (buff) put setpagedevice"
*MediaColor green: "1 dict dup /MediaColor (green) put setpagedevice"
*MediaColor goldenrod: "1 dict dup /MediaColor (goldenrod) put
setpagedevice"
*MediaColor pink: "1 dict dup /MediaColor (pink) put setpagedevice"
*MediaColor yellow: "1 dict dup /MediaColor (yellow) put
setpagedevice"
*?MediaColor: "
    save
        currentpagedevice /MediaColor
        {get} stopped
        {pop pop (white)} {dup null eq {pop (white)} if} ifelse
        = flush
    restore
"
*End
*CloseUI: *MediaColor
```

Input media (paper trays) are often selected on PostScript printers by specifying PageSize, MediaColor, MediaWeight, and MediaType. In the above example, media (paper) colors were defined for white, clear, blue, and so on. If you wanted to specify that the paper assigned to tray 5 uses blue paper, you could use one of these INI settings:

```
Tray5 = *MediaColor blue:
```

or

```
Tray5 = "1 dict dup /MediaColor (blue) put setpagedevice"
```

The first uses the UI keyword in the PPD file while the second uses the actual PostScript commands in a quoted string. When you use the UI keyword in an INI setting, always include the beginning asterisk (*) and the terminating colon (:).

Here are the default PostScript INI settings:

```
< PrtType:PST >
; UI keyword is used if PPD is specified and keyword is found.
; Otherwise, quoted string is used.
Tray1="0 statusdict /setpapertray get exec" or Tray1=*InputSlot
Upper:
Tray2="1 statusdict /setpapertray get exec" or Tray2=*InputSlot
Lower:
Tray3="2 statusdict /setpapertray get exec" or Tray3=*InputSlot
Manual:
Tray4="3 statusdict /setpapertray get exec" or Tray4=*InputSlot
Envelope:
; Make trays 5 through 9 use the PostScript commands for tray 1
Tray5=1
Tray6=1
Tray7=1
Tray8=1
Tray9=1
```

For GDI Printers

You can override the GDI tray commands by specifying an alternate paper tray to use. Here are the default GDI INI settings:

```
< PrtType:GDI >
  Tray1 = 1
  Tray2 = 2
  Tray3 = 3
  Tray4 = 4
  Tray5 = 1
  Tray6 = 1
  Tray7 = 1
  Tray8 = 1
  Tray9 = 1
```

For AFP Printers

You can override the AFP tray commands by specifying an alternate paper tray to use. Here are the default AFP INI settings:

```
< PrtType:AFP >
  Tray1 = 1
  Tray2 = 2
  Tray3 = 3
  Tray4 = 4
  Tray5 = 1
  Tray6 = 1
  Tray7 = 1
  Tray8 = 1
  Tray9 = 1
```

For Metacode Printers

You can override the Metacode tray commands by specifying an alternate tray name to use. Here are the default Metacode INI settings:

```
< PrtType:XER >
  Tray1 = MAIN
  Tray2 = AUX
  Tray3 = AUX
  Tray4 = AUX
  Tray5 = AUX
  Tray6 = AUX
  Tray7 = AUX
  Tray8 = AUX
  Tray9 = AUX
```

INCLUDING TRAY SELECTIONS IN A PRINT STREAM BATCH

To include the header with the tray selection in a print stream batch, the first section written or triggered to the batch must have a tray, such as Tray 1 or Tray 2, listed in its FORM.DAT file. Otherwise, the information is not written to that batch print stream. Here is an example of header information from a PostScript print stream that had these INI options:

```
< PrtType:PST >
  Tray1 =*InputSlot Upper:
  Tray2 =*InputSlot Lower:
```

Here is the example header:

```
GenericDict begin
%%BeginSetup
%%BeginFeature: *Duplex
false statusdict /setduplexmode get exec false statusdict /settumble
get exec
%%EndFeature
%%BeginFeature: *InputSlot Upper
0 statusdict /setpapertray get exec
%%EndFeature
```

SPOT COLOR SUPPORT

This table summarizes the support for spot color by the various Documaker print drivers.

Printer driver	Spot color support	RGB color support	Comments
AFP	No	Yes	
Email	No	Yes	The Email and EPT print drivers use another print driver to generate the actual content. If the underlying print driver is the PDF Print Driver, you will have support for named spot colors. If it is another print driver, you will have support for RGB-defined colors.
EPT	No	Yes	
GDI	No	Yes	
HTML	No	Yes	
MDR	No	No	
Metacode	No	Yes	Metacode supports highlight color where non-black colors are printed using the highlight color. For more information, see <i>Specifying Spot Color</i> on page 121.
PCL PXL	No	Yes	PCL does not support for Pantone color names or spot color names.
PDF	Yes	Yes	The Documaker PDF print driver supports named spot colors.
Postscript	No	Yes	
RTF	No	Yes	
VIPP	No	Yes	
XML	No	No	
XPS	No	Yes	

INDEX

A

- A4
 - PaperSize option 38, 124
- A4 page size
 - PaperSize option 169
- ABNORMAL statements 135
- Acrobat Reader
 - embedded fonts 215
- AddBlankPages function 201
- AddComment function 47, 158, 159, 202
- AddDocusaveComment function 302
- AdditionalDJDE option 129
- AddMultiPageBitmap rule 176
- ADDPAGES utility 120
- AddPropFonts option
 - MPM print driver 144
- AdjLeftMargin option 150
- AdjustTextWidth option 214
- Advanced Encryption Standard (AES) 227
- AESEncryption option 227
- AFP control group 39
- AFP printers
 - fonts 49
 - form-level comments 51
 - handling multiple trays 306
 - highlight color printing 41
 - INI options 37
 - overlays 49
 - page segments 49
 - paper size 38
 - resources 49
 - TLE records 305
 - troubleshooting 52
- AFPFMDEF utility 49
- AllowAccessibility option 220
- AllowAnnotate option 219
- AllowAssembly option 220
- AllowColorSheetLink option 279
- AllowCopy option 219
- AllowFormFields option 219
- AllowHighQualityPrinting option 220, 230
- AllowInput option 253, 278
 - RTF Print Driver 248
- AllowModify option 219

- AllowPrinting option 219

- archive
 - creating print streams for Docusave 301
 - TLE records 305

- Asian languages

- PCL 6 152

- Auto-size option 52

B

- bar codes 65, 190

- BarCode option

- RTF Print Driver 248

- BARR

- format 140

- interface attachment 140

- BARR SPOOL

- OutMode option 115

- BARRWRAP utility 140

- record length 140

- base fonts

- Acrobat Reader 211

- batch active flag 174

- batch printing

- deferred 30

- distributed 30

- options 33

- BatchPrint control group 177, 251

- Begin Page (BPG)

- adding data 47

- bitmap compression

- PCL print driver 158

- PostScript printer driver 238

- Bitmap option

- RTF Print Driver 248

- Bitmap Print Driver

- INI options 59

- BitmapHTTP option

- MPM print driver 143

- BitmapPath option

- MPM print driver 143

- BitmapResolution option

- MPM print driver 143

- bitmaps

- color 185

- compression for Metacode printers 117

- highlight color printers 157
- Metacode LGO files 133
- scaling 40
- Xerox images 133
- black rectangles 52
- blank pages 119
- BlankPageImage option 200
- BmpType option 59, 64
- BMSUB option
 - RTF Print Driver 248
- BmSub option 280
- BMSUBChar option
 - RTF Print Driver 248
- BmSubChar option 280
- Bookmark option
 - custom bookmarks 193
 - PDF printers 167, 168
- bookmarks
 - creating custom 193
 - DisplayMode option 169
- Box option
 - RTF Print Driver 248
- boxes 65
- boxes, WriteFrame option 252
- BPGScript option 47
- business envelopes 155
- C**
- CacheFAPFiles option 127
- CacheFiles option 128, 176
 - VIPP Print Driver 267
- CacheLogos option
 - VIPP Print Driver 267
- CacheMethod option 128
- CallbackFunc option 71, 178
 - EPT Print Driver 84
- case toggles 114
- cc:Mail 173
- CD/IG 138
- CenterWindow option 190
- Character Set field 42
- Chart option
 - RTF Print Driver 248
- ChartResolution option
 - AFP printers 37
 - Metacode printers 122
- charts 65
- BARRWRAP utility 140
- compression for Metacode printers 117
- printing on Metacode printers 116
- rendering on Metacode printers 121
- using the Metacode loader 129
- CheckImageLoaded rule
 - rotated variable fields 53
- CheckNextRecip INI option 175
- Class option
 - AFP printers 38
 - EPT Print Driver 79
 - GDI driver 105
 - Metacode printers 129
 - PCL printers 150
 - PDF printers 168
 - PostScript printers 236
 - RTF Print Driver 247
 - VIPP Print Driver 268
- clipboard
 - PDF security 219
- CMY palette 156
- Code Page Font field 42
- CODE statement 115
- CODEPAGE.INI file 61, 65
- CollapsePage option 278
 - MPM print driver 143
- color
 - support 185
- color bitmaps 60, 64
- color palettes 74
- ColorCharts option 125
- colors
 - for charts 125
 - PCL support for 148
 - printing 41
 - simple color mode 156
 - specifying ink for Metacode printers 121
 - troubleshooting for Metacode printers 135
- ColorSheet option 279
- CommonFonts control group 123
- Comp Pack 185
- Comp TIFF 185
- Compact Font Format (CFF) 187
- CompileInStream option 117
- compressed LOG format 59, 64
- Compression option 186, 238

- CompressMode option 117
- console messages 125
- CreatePlainText option
 - MPM print driver 143, 144
- CreateScriptFile option 279
- CSTSetMailRecip function 82
- CUSSetMailRecip function 82
- CUSSetMailRecipGVM function 85
- custom page sizes
 - PaperSize option 169
- D**
- DAL scripts
 - creating print streams for Docusave 302
- DAP.INI file 231
 - PDF compression option 186
- data
 - length validation 138
- Debug option
 - MPM print driver 144
- DefaultSymSet option 60, 65
- deferred batch printing 28
- DefLib option 61
 - PostScript printers 236, 243
- Desc option 68, 69
- Device field 298
- Device option 60, 64, 70, 118, 278
 - AFP printers 37
 - EPT Print Driver 79
 - GDI driver 104
 - MPM print driver 142
 - PCL printers 149
 - PostScript printers 235
 - RTF Print Driver 248
 - VIPP Print Driver 267
- digital signatures 196
- Direction option 190
- DirLinks option 278
 - MPM print driver 143
- DisplayCodedFont option 39, 42
- DisplayDocTitle option 190
- DisplayMode option 169, 191
- distributed printing 29, 30
- DJDE command 130
- DJDE statements
 - user-defined 129
- DJDECarrControl option 130
- DJDEForceOffsetEnd option 120
- DJDEIden option 114
- DJDELevel option 127
- DJDEOffset option 114
- DJDESkip option 114
- Documaker 58, 166
- Documaker Add-In for Microsoft Word 256
- Documaker Desktop 58, 73, 166
- Docusave
 - creating print streams 301
- DocusaveScript option 39, 302
- dots per inch
 - Resolution option 37
- DoubleOutputRes option
 - AFP printers 40
- DownloadFAP option 175, 238
 - and the CompileInStream option 117
- DownloadFonts option 159, 187
 - embedding fonts 210, 216, 217
 - EPT Print Driver 80, 86
 - GDI driver 104
 - overview 278
 - PCL printer resources 161
 - PCL printers 150
 - PDF printers 168
 - PostScript printers 236, 243
 - VIPP Print Driver 268
- DPRAddBlankPages rule 201
- DPRFap2Html rule 283
- DPRTblLookUp rule 283
- DSCHeaderComment option
 - PostScript printers 235
 - VIPP Print Driver 267
- DumpScript option 279
- duplex
 - and simplex on Metacode printers 119
 - compressed PCL files 157
 - printing multipage FAP files 137
 - switching modes 126
- Duplex option 190
- E**
- EjectPage rule
 - multipage FAP files 137
- electronic signatures 196
- email
 - aliases 97

- PDF files 172
 - sending a print-ready file 78
- Email Application Servers 97
- Embed Font field 212, 217
- embedded fonts 159
- embedded hex values 114
- EmbeddedINI option 231
- embedding fonts
 - compressing 169
 - how to 216
- EmptyFooters option 255
 - RTF Print Driver 248
- EmptyHeaders option 255
 - RTF Print Driver 248
- EmulateDuplexPrinter option 200
- Encrypt option 218
- end of report conditions 119
- End Page (EPG)
 - adding data 47
- EntryBackColor option 280
- EntryFontColor option 280
- envelope feeders 154
- EPT Print Driver
 - and the MPM Print Driver 87
- EPT print driver
 - including attachments 90, 92
 - MPM print driver 141
- EPTLIB 79
- EPTSetRecipFunc function 82, 85
- error messages
 - negative left offsets 52
- E-SIGN Act 196
- executive
 - PaperSize option 38, 124
- executive page size
 - PaperSize option 169
- Ext option 68, 69
 - Metacode printers 128
- extra info 193
- F**
- FAP2CFA utility 135
- FAP2FRM utility 133, 139
- FAP2MET utility 118, 128, 134, 135
- FAP2OVL utility 49
- FAPCOMP.INI file
 - Metacode loader 128
- FAPGetExtraInfo function 194
- FAPPutExtraInfo function 194
- FAX drivers 107
- fax, drivers 102
- FEED command 138
- Field option
 - RTF Print Driver 248
- file formats 59, 64
- file names 60
- FileName option 172
 - EPT Print Driver 80, 84
- FitToWidth option
 - GDI driver 104
 - PCL printers 150
 - PostScript printers 236
- FitWindow option 189
- floating section limitations 52
- FMRes control group 61
- Folder option
 - VIPP Print Driver 268
- font cross-reference files
 - AFP printer resolution 54
 - and the PDF Print Driver 215
 - GDI drivers 103
 - optimizing 184
- Font File field 217
- Font File Name field 212
- font IDs
 - PDF Print Driver 215
 - removing 184
- Font Index field 215, 217
- Font Manager
 - embedding fonts 216
- FontCompression option 169
- FontLib option 61
 - MPM print driver 144
 - PCL printers 161
 - PostScript printers 243
- fonts
 - caching 176
 - common font lists 122
 - Compact Font Format 187
 - compressing embedded fonts 169
 - embedding 159
 - embedding bitmap fonts 212
 - embedding PostScript fonts 216

- PDF file size 217
- PostScript printers 243
 - system 49
- Fonts option 61, 64
 - MPM print driver 143
- FontSearchOrder option 212, 213
- footers
 - in RTF files 255
- ForceColorBitmaps option 169
- ForcePrintInColor option 60, 67
 - MPM print driver 143
- FORM.DAT file
 - marking forms printer resident 140
- FormDef, AFP resources 49
- form-level comments 51
- FormLib option
 - PostScript printers 235
 - pre-compiled MET files 118, 128
- FormNameCR option 51
- forms
 - background 118
- frames
 - WriteFrame option 252
- free form text 201
- FRM files
 - CompileInStream option 118
- FRMFile option 140
- FSISYS.INI file 59
 - and the PDF Print Driver 167
- FSRSetFileAttachment API 96
- FudgeWidth option
 - AFP printers 37
- full-screen mode
 - DisplayMode option 169
- FullSupport option 107, 108
- Func option 68, 69
- FXRVALID utility
 - embedding fonts 210, 212
 - optimizing PDF files 185
- G**
- GDI driver
 - handling multiple trays 306
 - INI options 104
 - Netware Client 32 for Windows 95 298
 - troubleshooting 298
- GDIDevice option 105, 108
 - and the Device option 106
- GenData
 - creating HTML files 281
- GenerateAddInXML option 249, 256
- GenPrint
 - CheckNextRecip option 175
 - MultiFilePrint option 175
 - SendOverlays option 175
- GenPrint program
 - creating print streams for Docusave 301
- GHO hardware 116
- GOCA charts support 37
- graphics 65
 - bypassing printing 116
 - compression for Metacode printers 117
 - orientation 135
 - rendering 124
 - using the Metacode loader 129
- Graphics Device Interface (GDI) print driver 102
- GraphicSupport option
 - AFP printers 37
- GrayShades option 60, 67
- GVG hardware card 116, 133
- H**
- H2 strings 114
- H6 strings 114
- headers
 - in RTF files 255
- HiddenFieldScript option 279
- HideMenubar option 189
- HideToolbar option 189
- HideWindowUI option 189
- highlight color printing
 - AFP 41
- HighlightBlackCmd option 157
- HighlightColor option 149
 - VIPP Print Driver 268
- HighlightColorCmd option 157
- horizontal motion index 134
- HR option 278
 - MPM print driver 143
- HTML
 - print driver 277
- HTML format
 - MPM print driver 141

I

- IBMXREF.TBL file 42
- IDEN statement 114
- IDS 231
- IECOLOR.CSS file 279
- IgnoreExternalBitmapError option 300
- ImageOpt option 116
 - Metacode printers 116
- ImageOptNotSet option 116
- ImagePath option 279
- ImagePathCreate option 279
- imaging systems 158, 201
 - adding PJI comments 158
- IMG_ZIndex option 280
- InfoPak 120
- INI files 59
 - FSISYS.INI and the PDF Print Driver 167
- InitFunc option 159, 176
 - EPT Print Driver 79, 84
 - RTF Print Driver 248
- ink color 121
- inkjet printers 102
- in-line graphics 116
- inline graphics
 - and the CompressMode option 117
 - BARRWRAP utility 140
 - LOG files 39
- installable functions 118
- Internet Document Server (IDS)
 - compressed PCF files 157
 - paper size 38, 123

J

- JavaScript option 279
- JDEName option 114
- JDLCode option 114
- JDLData option
 - defined 115
 - Metacode printers 139
- JDLHost option 115
- JDLName option 114
- JDLRPage option 119
- JDLRStack option 119
- JDLs
 - setting up Metacode printers 113
- JES2 format 140, 301
- jogging pages 120

- JPEG files 185

- JPEGCompression option 187

- JSLs

- setting up Metacode printers 113

- jump to new sheet condition 119

K

- KeepFile option

- EPT Print Driver 80

- KeyLength option 220

L

- landscape

- AFP limitations 52

- graphic orientation 135

- Landscape option

- GDI driver 105

- LandscapeSupport option

- AFP printers 37

- LanguageLevel option 236

- legal

- PaperSize option 38, 124

- legal page size

- PaperSize option 169

- letter

- PaperSize option 38, 124

- letter page size

- PaperSize option 169

- limitations

- floating sections 52

- multipage FAP files 53

- PDF Print Driver 179

- line density errors 138

- LINE statement 115

- Linearize option 169, 192

- lines 65

- Loader:Met control group 128

- LoadFAPBitmap option 135, 175

- LoadPrintOnly option 167

- local printing 26

- LOG2PSEG utility 49

- LogCaching option 128

- Logo Manager 124, 133

- LOGO.DAT file

- printing MET files 129

- logos

- optimizing PDF files 184

- LogoUnloader control group 69

- Lotus Notes 173
- LRECL values 139
- M**
- Mail control group 97
- MailType option 97
- Map Coded Font (MCF) fields 42
- margins
 - added by PCL printers 150
 - setting minimum 254
- MasterResource control group 61
 - PCL resources 161
 - pre-compiled MET files 118
- MaxFonts option 123
- message information 79
- Message option
 - EPT Print Driver 80, 84
- MessageFile option
 - EPT Print Driver 88
- MET files
 - and multipage FAP files 137
- Metacode printers
 - creating print streams for Docusave 301
 - data length validation 138
 - end of report conditions 119
 - handling multiple trays 306
 - JSL INI options 113
 - resources 133
 - setting up 111
 - troubleshooting 135
- METDUMP utility 132
- METOPT utility
 - common font lists 123
- MinBottomMargin option
 - RTF Print Driver 249
- MinLeftMargin option
 - RTF Print Driver 249
- MinRightMargin option
 - RTF Print Driver 249
- MinTopMargin option
 - RTF Print Driver 249
- Mixed Object Document Content Architecture data streams 36
- Mobius
 - InfoPak 120
 - ViewDirect APIs 132
- Module option 59, 64, 68, 69, 278
- AFP printers 37
- EPT Print Driver 80, 84, 86
- GDI driver 104
- MPM print driver 142
- PCL printers 149
- PDF printers 168
- PostScript printers 235
- RTF Print Driver 248
- VIPP Print Driver 267
- monochrome 185
- monocolor 185
- MonospaceFonts option
 - MPM print driver 143
- MPM Print Driver
 - and the EPT Print Driver 87
- MPM print driver
 - example output 145
 - INI options 142
 - MultiFilePrint callback function 144
 - overview 141
- MRG2FAP utility
 - paper size 38, 123
- MRG4 format 301
- MsgPrtType option
 - EPT Print Driver 88
- MTCLoadFormset rule 132
- MultiFileLog option 71, 72, 175
 - EPT Print Driver 84
- MultiFilePrint callback function 174, 175
 - MPM print driver 144
- MultiFilePrint option 72
- MultiLinesPerCommand option
 - AFP printers 40
- multipage FAP files
 - and pre-compiled MET files 137
 - creating multiple FRM files 139
 - limitations 53
- MultiPage option 279
- Multipart MIME
 - EPT print driver 90, 92
- Multipart MIME format 87
- multiple step mode 174
- N**
- NAFILE.DAT file
 - rotated variable fields 53
- NamedColors option 39, 41

- negative left offset 52
- networks
 - printing 27
- NoBatchSupport option 177, 251
- NonFullScreenPageMode option 190
- non-stapled forms
 - and stapled forms 154
- NUBACK statements 120
- NUFRONT statements 120
- NumCopies option 190
- O**
- objects
 - negative left offset 52
- Octal strings 114
- offset, negative left 52
- OnDemand command records 38
- OnDemandScript option 38
- OpenType fonts
 - subsetting 214
- Opentype fonts 209
- operator-initiated printing 26
- Optimize option 123
- optimizing a PDF file 192
- orthogonal color palette 67
- OTextString option 125
- OTH record 212
- Outlook 173
- OutMode option
 - AFP printers 302
 - Metacode printers 115
 - Mobius 132
 - print streams for Docusave 301
- OutputBin option 151, 154
- OutputFunc option 157
 - MPM print driver 142
- OutputHalfRes option 40
- OutputMod option 157
 - MPM print driver 142
- OverlayExt option
 - GDI driver 104
 - PCL printers 149
 - PostScript printers 236
 - RTF Print Driver 249
 - VIPP Print Driver 267
- OverlayPath option
 - GDI driver 104
 - PCL printers 149, 161
 - PostScript printers 235, 243
- overlays
 - AFP resources 49
 - and the PDF Print Driver 175
 - landscape pages 52
 - multipage FAP files 53
- OVLCOMP utility
 - and PCL resources 161
 - and PostScript resources 243
- OwnerKey
 - option 220
 - PDFKEYGEN built-in function 220, 227
- P**
- page segments 49
- PageBorder option
 - MPM print driver 143
- PageBreaks option 278
- PageNumbering option 61
- PageNumbers option 168, 279
 - AFP printers 37
 - EPT Print Driver 80
 - GDI driver 104
 - PCL printers 149
 - PostScript printers 236
 - RTF Print Driver 249
 - VIPP Print Driver 267
- pages
 - jogging 120
 - numbering 104
 - starting new pages 119
- paper size
 - overriding commands 154
- paper sizes
 - changing on Metacode printers 134
- paper trays
 - Metacode printers 125
 - on HP 5si printers 153
 - PCL support for 148
 - switching 137
- PaperSize option 38, 123, 169
- PaperStockID option 130
- pass-through printing 298, 299
- passwords
 - encrypting 226
 - setting up 220

- PCL
 - simple color mode 156
- PCL printers
 - adding PJJ comments 158
 - bitmap fonts 161
 - compressed PCL 157
 - handling multiple trays 306
 - INI options 149
 - mixing simplex and duplex 157
 - overlays 161
 - PCL version 5, 5c, and 5e 148
 - PCL version 6 151
 - resources 161
 - simple color mode 156
 - using a staple attachment 154
- PCO interface
 - OutMode option 116
- PDF files
 - and bar codes 190
 - creating separate files 177
 - embedded fonts 208, 215
 - initial display 169
 - linearized 192
 - optimizing 184
- PDF Print Driver
 - fonts 179
 - limitations 179
- PDF/A
 - multiple width tables 215
- PDFAOPTIONS option 198
- PDFKEY tool 220, 228, 230
- PDFKEY utility 226, 228
- PDFKEYGEN built-in function 220, 227
- PDS members
 - caching 127
- performance
 - caching PDS members 127
 - PDF Print Driver 175
 - SplitPercent option 54
- permissions 221
 - setting up 220
- PickTrayByPDFSize option 190
- pixels per inch 60, 64, 143
- PJJComment option 158
- PJJCommentOn option 151
- PJJCommentScript option 150, 158
- placeholders 196
- PMetLib option
 - and the CompileInStream option 118
 - Metacode printers 128
- PMETLIB PDS 118
- point sizes 215
- Port option 63, 71, 108
- Portable Document Format 163, 164
- portrait graphic
 - orientation 135
- PostScript 61, 65
- PostScript fonts
 - embedded fonts 215
 - embedding 210, 212
- PostScript printers
 - handling multiple trays 306
 - INI options 235
 - PPD files 236, 243
 - resources 243
 - setting up 233
 - Type 1 fonts 243
- PreLoadRequired option 177, 251
- PrePrintedPaper option
 - AFP printers 38
 - EPT Print Driver 81
 - GDI driver 105
 - PCL printers 150
 - PostScript printers 236
 - RTF Print Driver 249
- Print 246
- Print in Color option 67
- Print option 230
- Print Services Facility 36
- Print window
 - and the Device field (GDI printing) 298, 299
 - and the PrePrintedPaper option 237
 - and the PrePrintedPaper option (AFP) 38
 - and the PrePrintedPaper option (GDI) 105
 - and the PrePrintedPaper option (PCL) 150
 - and the PrePrintedPaper option (PostScript) 236
 - and the SelectRecipients option 60, 104, 150
 - and the SendColor option 60, 104, 143, 149, 236

- suppressing 105
- printer console messages 125
- Printer control group 63
- Printer Job Language (PCL) comments 158
- Printer option 63
- Printer Resident field 139
- PrinterInk option
 - and the ColorCharts option 125
 - spot colors 121
 - troubleshooting 135
- PrinterModel option 242
 - Metacode printers 125
 - PostScript printers 236, 243
- printers
 - configuring trays 306
 - default printer 105
- Printers control group 59
- PrintFunc option 59, 64, 168, 278
 - AFP printers 37
 - EPT Print Driver 80, 84, 86
 - GDI driver 104
 - MPM print driver 142
 - PCL printers 149
 - PostScript printers 235
 - RTF Print Driver 248
 - VIPP Print Driver 267
- printing
 - batch options 33
 - deferred 28, 30
 - distributed 29
 - local printing 26
 - network printing 27
 - operator-initiated 26
 - system-initiated 30
 - under Windows NT 152
- PrintPageRange option 190
- PrintScaling option 190
- PrintToFile option 108
- PrintViewOnly option
 - AFP printers 38, 150
 - GDI driver 104
 - Metacode printers 127
 - PostScript printers 236
- Project option
 - VIPP Print Driver 268
- ProjectPath option
 - VIPP Print Driver 269
- ProportionalFonts option
 - MPM print driver 144
- PRTLIB
 - and the PDF Print Driver 174
- PrtType
 - BMP control group 59, 64
- PrtType control group
 - PDF compression option 186
 - PDF Print Driver 216
- PrtType option 59, 63
 - EPT Print Driver 79, 84, 86
- PrtType:AFP control group 37
- PrtType:XER control group
 - installable functions 118
 - required options 113
- PRTZCompressOutPutFunc function 157
- R**
- RecipFunc option
 - EPT Print Driver 80, 85
- Recipient option
 - and email aliases 97
 - EPT Print Driver 81, 84
- RecipMod option
 - EPT Print Driver 80, 85
- records
 - length 139
 - maximum number (Metacode) 136
- Rehabilitation Act of 1973 202
- RelativeScan option 118
- repeat counts 114
- ReplaceBitmap option 39, 41
- ReplaceFAPHeadFoot option
 - RTF Print Driver 249
- Resolution option 60, 64
 - AFP printers 37
 - GDI driver 104
 - Metacode printers 125
 - PCL printers 149
 - PostScript printers 235
 - rounding errors 54
 - VIPP Print Driver 267
- RightFax 158, 201
- Rotated Fonts field 184
- rotated variable fields 53
- RotateLandscapePages option 61

- rounding errors
 - SplitPercent option 54
- RPAGE command 130
- RSTACK command 130
- RTF
 - margins 254
 - print driver 246
 - separate files 251
 - WriteFrames option 252
- RTF files
 - enhanced 256
- RTFCompatibilityOptions option 249, 256
- RuleFilePool option 127
- Run Length Encoding (RLE) compression 238
- S**
- scaling output 102
- screen fonts
 - GDI drivers 103
- ScriptPath option 279
- ScriptPathCreate option 280
- Section 508 compliance 202
- SecurityGroup option 219
- SelectRecipients option 60
 - GDI driver 104
 - PCL printers 150
 - PostScript printers 237
 - VIPP Print Driver 268
- SendColor option 41, 60, 64, 67, 279
 - AFP printers 39
 - and the ColorCharts option 125
 - and the PrinterInk option 121
 - EPT Print Driver 81, 86
 - GDI driver 104
 - MPM print driver 143
 - PCL printers 149
 - PDF printers 168
 - PostScript printers 236
 - RTF Print Driver 249
 - troubleshooting 135
 - VIPP Print Driver 268
- SendOverlays option
 - AFP printers 37
 - GDI driver 104
 - PCL printers 149, 161
 - PDF printers 168
 - PostScript printers 235, 243
 - VIPP Print Driver 267
- SetOrigin rule
 - floating sections 52
- SetOverprint option 237
- setting up
 - PDF compression options 186
- shades 65
- SIDE statements 120
- signature placeholders 196
- simple color mode 149, 156
- simplex
 - and duplex on Metacode printers 119
 - compressed PCL files 157
 - switching modes 126
- SkipChartColorChange option 39
- SplitPercent option
 - 240 dpi print problems 54
 - defined 37
- SplitText option 215, 280
 - 240 dpi print problems 54
 - defined 37
- SpoolBatches option 178
- spot color 310
- staple attachments
 - and PCL printers 154
- StapleBin option 150, 154
- StapleJDEName option 126
- StapleOff option 236, 240
- StapleOn option 236, 240
- stapling forms
 - Metacode 126
 - PostScript 240
- start new page 119
- StreamBufferSize option
 - EPT Print Driver 81
 - RTF Print Driver 249
- SUB INK commands 135
- subject information 79
- Subject option
 - EPT Print Driver 81, 84
- SubsetAllEmbeddedFonts option 169, 214
- SuppressDialog option 108
 - and the SuppressDlg option 106
- SuppressDlg option
 - and the SuppressDialog option 106
 - GDI print driver 105

- SuppressLogoUnload option 39
- SuppressZeroData option
 - AFP printers 40
 - and the MultiLinesPerCommand option 40
- symbol fonts 217
- symbol sets 60, 65
- system
 - initiated printing 30
- system fonts 49
- T**
- Tag Logical Element (TLE) records 305
- task flow 25
- TemplateFields option 278
 - GDI driver 104
 - MPM print driver 142
 - PCL printers 150
 - PostScript printers 236
 - RTF Print Driver 249
 - VIPP Print Driver 268
- TermFunc option 159, 176
 - EPT Print Driver 80, 84
 - RTF Print Driver 248
- TerSub paragraphs 283
- text areas 65
- text labels 65
- Text option
 - RTF Print Driver 249
- TEXTCommentOn option 159, 202
- TEXTScript option 158, 201
- thumbnails
 - DisplayMode option 169
- TL/DL buffers 134
- TLEEveryPage option 38, 305
- TLEScript option 38, 305
- TLESeparator option 38, 305
- transferring files
 - from Xerox format disks 140
- Tray option
 - VIPP Print Driver 268
- trays
 - configuring printer trays 306
 - for the HP 5SI printer 153
 - Metacode printers 125
 - overriding commands 154
 - selecting 308
 - troubleshooting 137
- TrimWhiteSpace option
 - AFP printers 40
- troubleshooting 230
- TrueType 61, 65
- TrueType fonts
 - Asian languages 152
 - embedding 210, 212
- Type 1 fonts 217
- TypeFace field 211
- U**
- Unicode 151
- UseCompactFonts option 188
- UserKey
 - option 220
 - PDFKEYGEN built-in function 220, 227
- using
 - the PDF Print Driver 163
- V**
- value-added processes 127
- variable fields 65
 - in text areas 54
 - rotated 53
- VB datasets 139
- VBPrOptions control group 105
- Vector option
 - RTF Print Driver 250
- vectors 65
- ViewDirect APIs 132
- Virtual Storage Access Method 127
- VSAM control group 127
- W**
- white outlines 237
- white space
 - suppressing 40
- Windows
 - PostScript printers 237
 - printer ports 152
- WingDings 217
- WordDateFormats control group 250, 253
- WordTimeFormats control group 250, 253
- WriteFrames option 252
 - RTF Print Driver 250
- WriteToFile option 94, 95
- WSCOLOR.CSS file 279
- X**
- XERDNLD utility 140

XERLoadDocuMerge loader function 132

Xerox

3700 printers 125

4000 printers 112

4050 printers 138

4135 printers 138

4235 printers 116, 134, 138

4635 printers 138

4850 printers 138

9000 printers 112

9700 printers 134

9790 printers 134, 138

fonts 133

format floppies 140

forms 133, 139

forms and memory 134

highlight color printers 121

images 133

JSL INI options 113

Laser Printing Systems 112

line drawing font 138

logos 133

setting up Metacode printers 112

XMP library 93

XRFFile option 61, 65

Z

z/OS

generating PostScript output 238