Policy Production System

Features and Enhancements

version 11.3

Skywire Software announces a new release of the Policy Production System (PPS). This document introduces PPS version 11.3 and describes its features and enhancements.

To receive the full benefits of the new product features included in this and earlier releases, Skywire Software's Education Services Group offers a comprehensive range of training classes. For a list of courses, including fees and availability, please visit our web site at www.skywiresoftware.com.

Installation

For installation instructions, see the Documaker Workstation Supervisor Guide..

Skywire Software, L.L.C.
3000 Internet Boulevard
Suite 200
Frisco, Texas 75034
www.skywiresoftware.com

Phone:	**(U. S.)**	**972.377.1110**
	(EMEA)	**+44 (0) 1372 366 200**
FAX:	**(U. S.)**	**972.377.1109**
	(EMEA)	**+44 (0) 1372 366 201**
Support:	**(U. S.)**	**866.4SKYWIRE**
	(EMEA)	**+44 (0) 1372 366 222**
		support@skywiresoftware.com

Contents

CHAPTER 1

PPS Features and Enhancements

Skywire Software proudly announces PPS version 11.3.

This document provides detailed information on the specific features and enhancements to PPS.

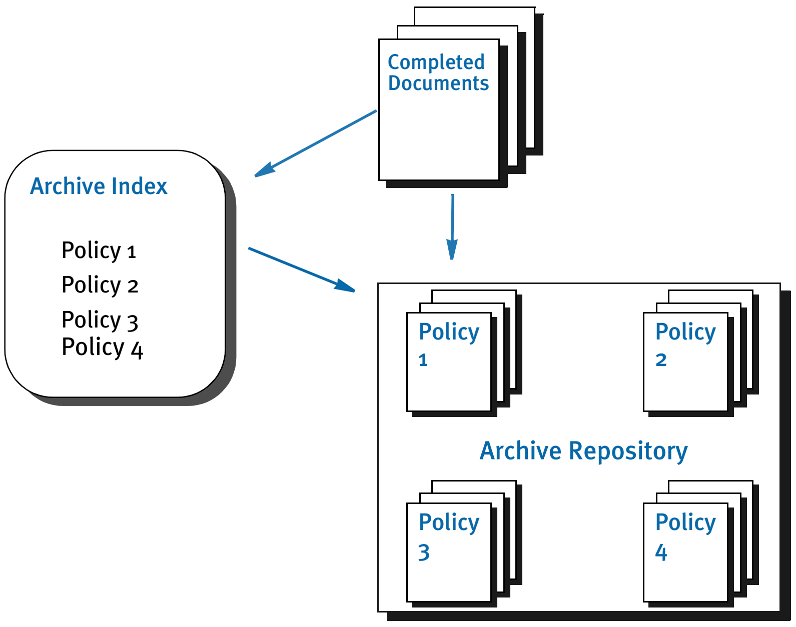
- For information about the new PPS Reporting Tool, see PPS Reporting Tool on page 2.
- For a list of the new features and enhancements, see PPS features on page 3.
- For a list of the new features and enhancements, see DAL enhancements on page 5.

System Enhancements

The following pages describe the features which have been included in PPS version 11.3. As you review the descriptions of the new features, keep in mind that for PPS version 11.3 you can purchase the add-on PPS Reporting Tool.

PPS Reporting Tool

PPS includes an archive of completed policy documents. You can retrieve documents from archive for viewing or reprinting, or as a source of data for new transactions.



The PPS archive is not a database, but it does have an index for searching a repository of compressed documents.

The completed documents in the archive are more than just pictures of what was printed. These documents are intelligent and are potential sources of valuable business information. The documents are comprised of references to forms in the library and each form is a container of the variable data that was put onto the form.

The archive is, in essence, a warehouse of valuable data — if only there was a way to query this archive and extract this valuable business data asset and use it in reports. Now there is a way — the PPS Reporting Tool.

The PPS Reporting Tool adds data mining and reporting capabilities to the PPS archive. With this add-on tool, the PPS archive becomes a valuable source of business intelligence -- an asset to be tapped for a myriad of reporting needs.

Note: For more information about the PPS Reporting Tool, contact your Skyware Software Sales Representative.

New Features

PPS features

Here is a list of the new features in PPS version 11.3:

Feature	For more information, see...
0190	Additional Paper Sizes on page 7
1294	Using Data Matrix 2-D Barcodes on page 17
1468	Exporting When Batch Printing WIP on page 18
1486	Enhanced Support for Imaging Systems on page 20
1522	Preventing Field Duplication on page 20
1527	Automatically Importing XML Files on page 21
1534	Creating Form Fields on page 23
1537	Adding Information to an Email Subject Line on page 24
1553	Setting Margins for the RTF Print Driver on page 24
1558	Recognizing All Caps on page 25
1562	Selecting Multiple Recipients on page 25
1577	Including a Form Set Number on a Banner Page on page 26
1579	Specifying the File Size when Splitting Archive Files on page 27
1581	Overriding a Date Range when Splitting an Archive on page 28
1594	Using the WIPField Built-in Function on page 28
1603	Encrypting INI Values on page 28
1604	Adding Hypertext Links on page 29
1606	Improved Support for Color and Graphics on page 30.
1617	Faster Compression of Archives on page 31
1648	Pasting Unformatted Text from the Clipboard on page 32
1674	Remembering the Size of Your Window on page 32.
1690	Using LZW Compression in TIFF Files on page 33.
1691	Keeping the Contents of a Text Area Together on page 33.

Feature	For more information, see...
1697	Changing the Case on page 34.
1717	Using the Mouse Wheel to Scroll on page 35.
1731	Copying Text to the Clipboard on page 35.
1732	Recognizing Hyphenation Keywords in RTF Files on page 35.
1766	Displaying the Required Forms on page 36.
1768	Selecting Multiple TERSUB Paragraphs on page 36.
1770	Selecting Forms on page 36.
1796	Embedding non-Logo Bitmap Files on page 37.
1803	Simple WIP Print on page 37.
1804	Rotating a Dynamic Bitmap on page 38.
1808	Miscellaneous Enhancements on page 39.
1811	Changing the Default Field Color on page 40.
1813	Showing Blank Fields as NULL Fields on page 41.
1824	Mapping Old Archive Keys to Current Document Keys on page 42.
1834	Activating the Spell Check Option on page 42.
1868	Automatically Re-Paginating Images on page 42.
1872	Storing the WIP Index and WIP Data in Database Tables on page 44.
1877	Generating Readability Statistics on page 59.
1895	Generating the USPS Intelligent Mail Barcode (4-State Customer Bar Code) on page 60.
1975	Inserting State Stamps and Signatures on page 61
2002	Using Superscript and Subscript in Text Labels, Text Areas, and Variable Fields on page 63.
2067	Prompting for Search Criteria on page 64.
2112	Mapping Alternate WIP Index Columns from Imported WIP Data on page 64.
2113	Suppressing Duplicate Form Descriptions on page 65.

Feature	For more information, see...
2136	Using the New WIP Fields on page 65
2144	Controlling Pagination when Editing a Form Set on page 66.
2155	Defaults for the Module and PrintFunc Options on page 66.
2164	Using the New Complete and Exit Option on page 66.
2178	Adding the Transaction Code During an Import on page 68.
2180	Printing Multiple Copies from the GDI Print Driver on page 68.
2324	Specifying the Number of Copies on page 69
2346	Resizing Grid Windows on page 69

DAL enhancements

Here is a list of the enhancements made to the Document Automation Language (DAL):

Feature	For more information, see...
1479	Using the TotalPages and TotalSheets Functions on page 70
1481	Using the PageImage Function on page 79.
1502	Using DAL Functions for Bit Manipulation on page 72.
1507	Enhanced Availability of DAL Functions on page 80.
1515	Using DAL to Manipulate File Names on page 81.
1524	Using the PageInfo Function on page 83.
1533	Using the NL Function on page 85.
1545	Using the Exists and GetValue DAL Functions on page 85
1548	Miscellaneous New DAL Functions on page 87
1549	Using the MOD function on page 95.
1554	Using the New ResetFld Function Parameters on page 96.
1563	Using the SetRequiredFld Function on page 97.
1589	Using the ListInList Function on page 98.
1641	Using the New DAL Functions on page 99.
1807	Creating Variable Length Records from DAL Using Flat Files on page 108.

Feature	For more information, see...
1995	Converting Integers to Characters on page 108.
2143	Paginating Forms Using DAL Functions on page 109.
2158	Using the STRCompare Function on page 112.
2197	Using the Time Zone DAL Functions on page 113.
2256	Using the SetLink Function on page 121.
2283	Using the IsPrintObject Function on page 122.
2311	Using the SetFormDesc Function on page 122.

0190
PPS

Additional Paper Sizes

The system now supports additional paper sizes including ledger and the international sizes of A3 and B5. The following tables show the paper sizes now supported. The width and height are in FAP units (2400 per inch), millimeters, and inches.

Note: Please note that the NA file now stores the actual section height and width for custom sized images instead of a code. This information is now stored in the SIZE entry in the NAFILE.DAT file. Here is an example:

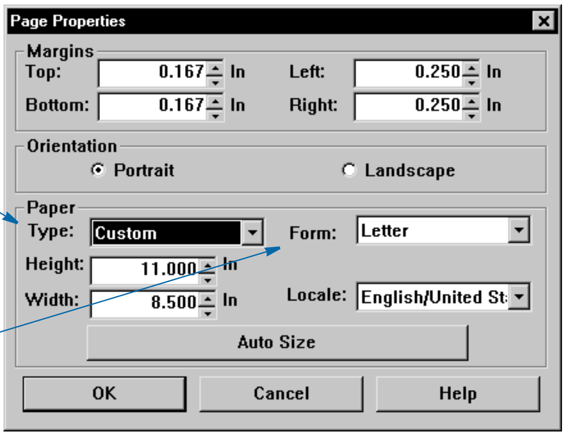
```
\NA=q1snam,LN=1,DUP=LB,SIZE=3360x18600,TRAY=U,X=600,Y=600...
```

The height and width are in FAP units (2400 per inch).

The Page Properties window in Image Editor (choose Format, Page Properties) now includes a new field called Form. This field is available when you set the paper type to Custom.

If you choose Custom here

the Form field is available



Page Properties

Margins
Top: 0.167 In
Left: 0.250 In
Bottom: 0.167 In
Right: 0.250 In

Orientation
(● Portrait) ○ Landscape

Paper
Type: Custom
Form: Letter
Height: 11.000 In
Width: 8.500 In
Locale: English/United St

The Form field helps identify what type of paper to use when you print the custom image. For example, if the dimensions of the custom image are 12x14, the system needs to know what size paper to use when you do a test print of the image.

The system defaults to the size of paper that will contain the custom image, but you must tell it what paper is installed on your printer. For images small enough to fit on letter size paper, the system defaults to letter.

Note: This affects image printing from Image Editor and has no effect on Form Set Manager or 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.

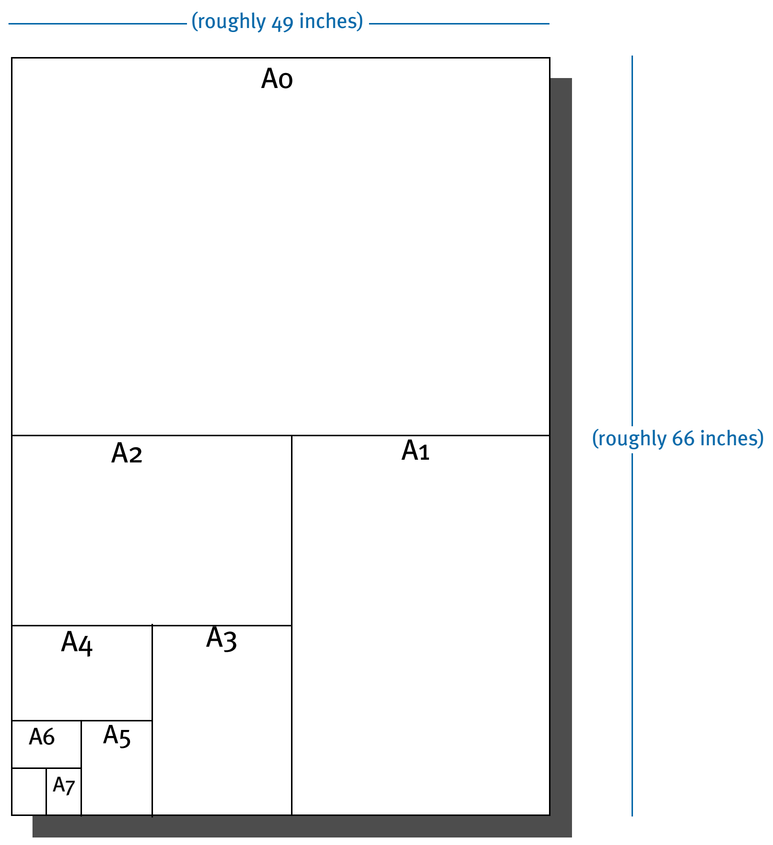
		Width x Height		
Name	Code	FAP units	Millimeters	Inches (approximate)
US letter	0	20400 x 26400	216 • 279	8• x 11
US legal	1	20400 x 33600	216 • 356	8• x 14
US executive	3	17400 x 25200	190 • 254	7• 10•
US ledger	4	40800 x 26400	432 x 279	17 x 11
US tabloid	5	26400 x 40800	279 • 432	11 x 17
US statement	6	13200 x 20400	140 x 216	5• x 8•
US folio	7	20400 x 31200	216 x 330	8• x 13
US fanfold	8	35700 x 26400	378 x 279	14^{7}⁄ $_{8}$ 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•
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• x 23 3/8
ISO A3	23	28063 x 39685	297 x 420	11• x 16•
ISO A4	2	19842 x 28063	210 x 297	8• x 11•
ISO A5	25	13984 x 19842	148 x 210	5 7/8 x 8•
ISO A6	26	9921 x 13984	105 x 148	4 1/8 x 5 7/8
ISO A7	27	6992 x 9921	74 x 105	2 7/8 x 4 1/8
ISO A8	28	4913 x 6992	52 x 74	2 x 2 7/8
ISO A9	29	3496 x 4913	37 x 52	1• x 2

		Width x Height		
Name	Code	FAP units	Millimeters	Inches (approximate)
ISO A10	30	2457 x 3496	26 x 37	1 x 1•
ISO 2A	32	112345 x 158927	1189 x 1682	46• x 66•
ISO 4A	34	158927 x 224690	1682 x 2378	66• x $93^{5}/_{8}$

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.

			Width x Height	
Name	Code	FAP units	Millimeters	Inches (approximate)
ISO B0	40	94487 x 133605	1000 x 1414	$39^{1}/_{8}$ x $55^{1}/_{8}$
ISO B1	41	66802 x 94487	707 x 1000	$27^{7}/_{8}$ x $39^{1}/_{8}$
ISO B2	42	47244 x 66802	500 x 707	$19^{5}/_{8}$ x $27^{7}/_{8}$
ISO B3	43	33354 x 47244	353 x 500	$13^{7}/_{8}$ x $19^{5}/_{8}$
ISO B4	44	23622 x 33354	250 x 353	$9^{7}/_{8}$ x $13^{7}/_{8}$
ISO B5	45	16630 x 23622	176 x 250	7 x $9^{7}/_{8}$
ISO B6	46	11811 x 16630	125 x 176	5 x 7
ISO B7	47	8315 x 11811	88 x 125	3• x 5
ISO B8	48	5858 x 8315	62 x 88	2• x 3•
ISO B9	49	4157 x 5858	44 x 62	1• x 2•
ISO B10	50	2929 x 4157	31 x 44	1• x 1•
ISO 2B	52	133605 x 188974	1414 x 2000	55• x 78•
ISO 4B	54	188974 x 267209	2000 x 2828	78• x 111•

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.

		Width x Height		
Name	Code	FAP units	Millimeters	Inches (approximate)
ISO C0	60	86645 x 122550	917 x 1297	36 1⁄8 x 51
ISO C1	61	61228 x 86645	648 x 917	25• x 36
ISO C2	62	43275 x 61228	458 x 648	18 x 25•
ISO C3	63	30614 x 43275	324 x 458	12• x 18
ISO C4	64	21638 x 30614	229 x 324	9 x 12•
ISO C5	65	15307 x 21638	162 x 229	6 3⁄8 x 9
ISO C6	66	10772 x 15307	114 x 162	4• x 6 3⁄8
ISO C7	67	7653 x 10772	81 x 114	3• x 4•
ISO C8	68	5386 x 7653	57 x 81	2• x 3•
ISO C9	69	3779 x 5386	40 x 57	1 5⁄8 x 2•
ISO C10	70	2646 x 3779	28 x 40	1 1⁄8 x 1 5⁄8
ISO DL	71	10394 x 20787	110 • 220	4 1⁄3 x 8 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 FAP units (2400 per inch), millimeters, and inches. The inch dimensions are approximate.

		Width x Height		
Name	Code	FAP units	Millimeters	Inches (approximate)
JIS B0	80	97322 x 137573	1030 x 1456	40• x 57•
JIS B1	81	68787 x 97322	728 x 1030	28• x 40•
JIS B2	82	48661 x 68787	515 x 728	20• x 28•
JIS B3	83	34393 x 48661	364 x 515	14• x 20•
JIS B4	84	24283 x 34393	257 x 364	10 1⁄8 x 14•
JIS B5	85	17197 x 24283	182 x 257	7• x 10 1⁄8

		Width x Height		
Name	Code	FAP units	Millimeters	Inches (approximate)
JIS B6	86	12094 x 17197	128 x 182	5 x 7•
JIS B7	87	8598 x 12094	91 x 128	3• x 5
JIS B8	88	6047 x 8598	64 x 91	2• x 3•
JIS B	89	4252 x 6047	45 x 64	1• x 2•
JIS B10	90	3024 x 4252	32 x 45	1• x 1•

Paper Size Support

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

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 16 for more information.

Paper size	PDF, RTF, HTML, and Metacode	PXL[1]	PCL[2]	GDI[2]	PST[3]	AFP[4]
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
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

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 16 for more information.

Paper size	PDF, RTF, HTML, and Metacode	PXL[1]	PCL[2]	GDI[2]	PST[3]	AFP[4]
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
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

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 16 for more information.

Paper size	PDF, RTF, HTML, and Metacode	PXL[1]	PCL[2]	GDI[2]	PST[3]	AFP[4]
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 16 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 Docutoolbox Reference.

1294 PPS

Using Data Matrix 2-D Barcodes

The system now supports Data Matrix 2-D barcodes. A Data Matrix barcode consists of black and white squares arranged in either a square or rectangular pattern. You can encode up to two kilobytes of text or raw data.

You can use the Data Matrix barcode with printer finishing equipment, such as equipment from manufacturers like Gunther or Pitney Bowes. Here is an example of a Data Matrix 2-D barcode:

Note: While the maximum number of alphanumeric characters for some symbol sizes, such as 88 x 88, in the Data Matrix specification can exceed 1024, the maximum number of alphanumeric characters for a variable field in a Documaker section (FAP) is 1024. So these larger symbol sizes are effectively restricted to 1024 characters.

Fonts

The system draws the Data Matrix barcode using fonts instead of graphic commands. The new fonts are listed below and referenced in the new font cross-reference (FXR) files (rel113.fxr and rel113sm.fxr) included in this release. The font IDs for the Data Matrix fonts are numbered 13504, 13505, and 13506.

The TrueType, PostScript, PCL, AFP (240 and 300 DPI), and Metacode fonts you need to produce the bar code are included in version 11.3 and listed below:

Size	Font name
TrueType	
All sizes	dm______.ttf
Postscript	

Size	Font name
All sizes	dm______.pfb
PCL	
4 point	fpdmn4.pcl
5 point	fpdmn5.pcl
6 point	fpdmn6.pcl
AFP 240 DPI	
4 point	xodadmn4.fnt, cofadmn4.240 (Coded Font, Character Set)
5 point	xodadmn5.fnt, cofadmn5.240
6 point	xodadmn6.fnt, cofadmn6.240
AFP 300 DPI	
4 point	xodadmn4.fnt, cofadmn4.300 (Coded Font, Character Set)
5 point	xodadmn5.fnt, cofadmn5.300
6 point	xodadmn6.fnt, cofadmn6.300
Metacode	
4 point	fxdmn4.fnt Rotations: f9dmn4.fnt, f1dmn4.fnt, f2dmn4.fnt (90, 180, 270)
5 point	fxdmn5.fnt Rotations: f9dmn5.fnt, f1dmn5.fnt, f2dmn5.fnt
6 point	fxdmn6.fnt Rotations: f9dmn6.fnt, f1dmn6.fnt, f2dmn6.fnt

1468
PPS

Exporting When Batch Printing WIP

Now you can have PPS export a form set when it batch prints documents in WIP. Through the use of INI options, PPS makes sure the export of the form set is delayed to make sure it is not modified before being printed.

PPS checks the BatchExport option to determine which export routine you want to use. It then loads the form set into memory. This is necessary for the export function, but not for printing.

The export function is called before the printing functions are called. This way, if an error occurs during the export, the printing process stops and returns the error. Since the form set is still in WIP, you can correct the error.

To use this feature, include the BatchExport option in the BatchPrint control group, as shown in this example:

```
< BatchPrint >
    BatchExport = DLL->FunctionName;SourceINI;DestINI;
```

NOTE: You only need to specify the *DLL->FunctionName* to run an export. The *SourceINI* and *DestINI* parameters are optional.

The BatchExport option identifies the DLL and name of the export function you want to use. The export function does not have to be defined in the ExportFormats control group. Define the export function name the same way you define other entry hooks — a DLL name followed by the name of the function to dynamically call to do the export. Here is an example:

```
< BatchPrint >
    BatchExport = TRNW32->TRNExportV2;Export2;ExpFile_CD;
```

Handling different export options

To handle different export options used when called in this manner, you can include two optional parameters after the export function name. Separate the parameters with semicolons. Each parameter names an INI control group that contains various options.

The first or source control group (*Export2* in the example) specifies the INI control group which contains alternative INI options you want to apply to the second INI control group (*ExpFile_CD* in the example). The second or destination control group should be the control group normally associated with the export function you want to use.

In the example above, the system copies the INI settings found under the Export2 control group to the ExpFile_CD control group. The standard V2 export function then uses the ExpFile_CD control group to find its options.

For instance, you might define the following:

```
< Export2 >
    Overwrite = Yes
    SuppressDlg = Yes
    File = ~HEXTIME .EXP
< ExpFile_CD]
    Overwrite = No
    File = Output.EXP
    Path = .\data\
```

NOTE: You only have to include in the source control group options you want to add or replace options in the destination control group.

Once the export function finishes, the system restores the original INI options and values from the destination group. This makes sure your default INI options are left intact once the export operation has finished.

So, using the example above, after the substitution takes place you would have the following options defined for the standard export:

```
< ExpFile_CD >
    Overwrite = Yes
```

```
        SuppressDlg = Yes
        File = ~HEXTIME .EXP
        Path = .\data\
```

By restoring the original INI options and values for the destination control group, the system lets manually called export functions work with a standard set of options, while allowing the BatchExport method to be more automated.

Keep in mind that the export function does not know it is being called indirectly. It will operate in its normal fashion. So, unless you override INI options as described here, the system may ask for an export file name or other information.

To further automate exports, be sure to check the INI options offered for the export function and define those that answer or suppress questions for the export process so it can operate without user input.

1486
PPS

Enhanced Support for Imaging Systems

The PCL and PDF drivers can now 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 > or < PrtType:PDF >
    TEXTScript     = imaging.DAL
    TEXTCommentOn  = formset
```

1522
PPS

Preventing Field Duplication

This feature lets you prevent the duplication of fields with global and form scope during a standard export. Now the occurrences are limited based on the scope of the field. For instance, a field with global scope will only appear once in your export file. A field with form scope will appear once for every form that has the scope of this field as form.

Here is an example of the INI option that turns on this feature:

```
< ExpFile_CD >
    AllowDuplicateField = No
```

This option defaults to Yes.

For this feature to work, you must also set the OutputFieldScope option to Yes. To summarize, if...

- The OutputFieldScope option is set to Yes
- The AllowDuplicateField option is set to No
- The form set has fields with both global and form scope, such as a policy number field,

then during a standard export the number of times a field such as a policy number appears is based on the scope.

1527 PPS

AUTOMATICALLY IMPORTING XML FILES

Now you can automatically import an XML file into PPS. To use this feature, you should know how to use the Timer Library.

Below is an example of two import options in the TimerFunc group. There may be occasions when you want to import data from more than one directory. For example, you may have one directory set up for form sets with a WIP status code and another directory for form sets with a BatchPrint status code. To handle these requirements, you can define more than one group settings in your INI file, as shown here:

```
< TimerFuncs >
    01 =;0;0;300;WXMW32->WXMAutoImportXML;
    02 =;0;0;300;WXMW32->WXMAutoImportXML; \AUTOBATCHXML
```

In this example, the first option points to the default control group, which is AutoImportXML. The second option points to the alternate control group, AutoBatchXML.

Each of these control groups can contain different options which determine the behavior of the import function. Also, the flags (STATE, URGENCY, and SECONDS), indicated by *0;0;300*, do not have to be the same. Check your Timer Service Functions handbook for more information on how to set up your timer.

The other INI control group is shown here:

```
< AutoImportXML >
    Path                = \data\
    File                =
    Match               = *
    Ext                 = .XML
    DeleteOnSuccess     = Yes
    DeleteOnFail        = No
    TransactionCode     = NB
    Description         = XMLImported
    StatusCode          = WIP
    RecordType          = NEW
```

```
Edit                = No
KeyID               = ~FORMSETID
SuppressAllMessages= Yes
```

Option	Description
Path	This option tells the system what directory the import files are in. Make sure the path specified by this option is valid and the user has access to it.
File	Use this option to specify the file name that should be imported by the system. Leaving this option blank tells the system to scan for any file with the extension specified in the Ext option.
Ext	Use this option to specify the file extension the import file should have. If the File option, does not include an extension, the system looks to this option for the import file extension. The default is XML. Avoid using .BAD as an extension, because unsuccessful import files are renamed with that as the extension. Also, do not use a single period because it can conflict with multi-user capabilities in a network environment.
Match	Use this option with the Ext option to locate import files when the File option is blank. The system searches the directory specified in the Path option. The default is an asterisk (*), which is the DOS wildcard. All valid DOS file name searches can be performed with this option.
DeleteOnSuccess	Use this option to delete the import file after it has been imported. The default is Yes. If you want to keep the file, enter No. The system renames the file using a randomly-generated name.
DeleteOnFail	Use this option to delete the import file if it fails to import. This defaults is No. The system will rename the import file with an extension BAD. Enter Yes if you want to delete the import file if it fails to import.
TransactionCode	If the imported file does contain a WIP transaction code, you can use this option to specify the value for the transaction code. The default is NB (New Business).
Description	Use this option to specify the WIP description. If the imported file has a WIP description, the system ignores this option. If you omit this option and the imported file description field is blank, the default is imported.

Option	Description
StatusCode	Use this option to specify the WIP status code. If the imported file had a WIP description, the system ignores this option. If you omit this option and the imported file status code field is blank, the default is WIP. The system checks the status codes in the INI file with the one given for a valid code. By setting the status code to WIP, the you can verify whether the form set is a work-in-process, or whether the form set needs to be marked with some other status code, such as Batch Print.
RecordType	If the imported file does not contain a WIP record type, you can use this option to specify the value for the record type. The system checks the record type with those listed in the INI file. The default is New.
Edit	By default, this option is set to No. This means the system saves import files directly to WIP without any data entry. A Yes value for this option opens the form set created by the imported file, for data entry.
KeyID	Use this option to supply a value for the KeyID, if the imported file did not have one. The default for this option is XMLIMPORT.
SuppressAllMessages	Use this option to suppress all messages. The default is No.

1534
PPS

Creating Form Fields

The RTF Print Driver now converts 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 in Image Editor 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 Skywire Software format used on the image. To the right, you list the Word format you want to use.

1537 PPS ADDING INFORMATION TO AN EMAIL SUBJECT LINE

You can now use the Append_Subject INI option to add information to the subject line when sending emails via PPS and Documaker Server. This option will work with all features that use the email system, such as EPT, routing, and so on.

Here are some examples of how you can use this option:

```
< MailType:MSM >
    Append_Subject = [ ~Key1 | ~USERID ]
```

or

```
    Append_Subject = ~Key1 | ~USERID
```

or

```
    Append_Subject = ~Key1
```

or

```
    Append_Subject = ~DALRUN MyMessage.DAL
```

or

```
    Append_Subject = Skywire
```

In these examples, the tilde (~) in front of Key and USERID tells the system to retrieve the Key1 and user ID values from the form set currently loaded in memory. If no form set is in memory, the value for the Append_Subject option will be blank.

The example *~DALRUN MyMessage.DAL* tells the system to run a DAL script called MyMessage.DAL if necessary.

The pipe symbol (|) between KEY1 and USERID is not required. It is only there to separate the two values. For example, if the value for ~KEY1 is *Skywire*, and the value for ~USERID is *James Brown*, the subject line will look like this:

```
FILE_FAPMSG [ Skywire| James Brown ]
```

Without the pipe symbol (|), the subject line will look like this:

```
FILE_FAPMSG [ Skywire intJames Brown ]
```

1553 PPS SETTING MARGINS FOR THE RTF PRINT DRIVER

The RTF print driver produces margins by calculating what is required and putting the result in the RTF output. Now you can 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.

1558 PPS

RECOGNIZING ALL CAPS

The system now recognizes the All Caps attribute applied to text when you use the RTF Print Driver to open a Microsoft Word DOC file or an RTF file. For example, suppose you want the word *NAME* to appear in all caps.

In Word, you can use the Format, Change Case option to change the text to uppercase. This change is retained in the RTF file and even in an ASCII text file. And now, the system retains it too when it opens open the RTF file.

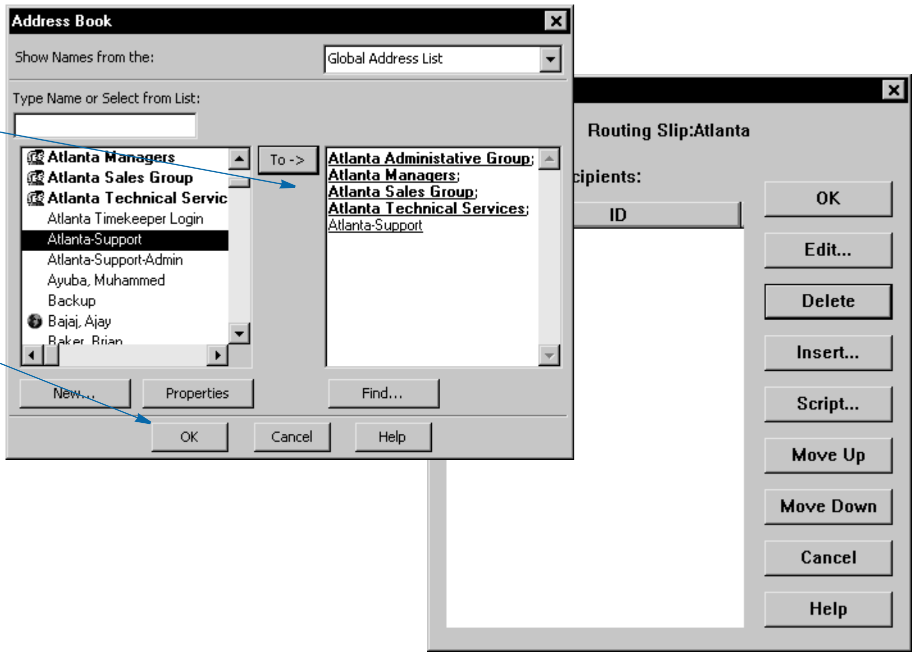
1562 PPS

SELECTING MULTIPLE RECIPIENTS

You can now select multiple recipients from your address book and add them to a routing slip all at once. Previously, you had to add them one at a time.

Select all the recipients you want to add to the routing slip

Then click Ok and the system will insert the entire list.



NOTE: This only works with Microsoft Outlook's address book.

1577
PPS

INCLUDING A FORM SET NUMBER ON A BANNER PAGE

You can now tell PPS to include the contents of a KeyID field on a banner page. For example, when you renew an insurance policy you retrieve the original policy from archive and then assign it a new policy number on the Form Selection window. Now you can add a field to the banner page that shows the previous policy number.

To do this, create a field on the banner page called, for instance, *PRVPOL*. In your INI file, include the new field in the Banner control group, as shown here:

```
< Banner >
    Field   = PRVPOL
    Field   = POLICY NBR
    Field   = INSURED NAME
    Name    = PFORM
    Size    =
```

In the BannerProc option, be sure to include the following:

```
< AFEProcedures >
    BannerProc  = TRNw32->TRNSetBannerFormInfo
```

Also, include the OriginalKeyID option in the FormSelection control group, as shown here

```
< FormSelection >
    OriginalKeyID = PrvPol
```

You can enter any name you like for the field. *PrvPol* is used as an example.

NOTE: The OriginalPolicyNumber and OrigPolicyNumber options perform the same function in the FormSelection control group as the OriginalKeyID option.

And finally, you must update the Banner control group to reference the field you named in the OriginalKeyID field. Here is an example:

```
< Banner >
    Field  = PrvPol
```

Keep in mind the order of the Field options in the Banner control group must match the order of fields on the banner page. For instance, if there are three fields on the banner page in the order of PolicyNum, RecipName, PrvPol, you must list the Field options in that same order, as shown here:

```
< Banner >
    Field = PolicyNum
    Field = RecipName
    Field = PrvPol
```

1579
PPS

Specifying the File Size when Splitting Archive Files

You can now limit the size of the CAR files created when you split an archive file using the ARCSPLIT utility or PPS.

To use this feature you must set the split file size no less than 100KB and no more than 1.4GB. The system does not accept values outside that range.

In the ARCSPLIT utility, you can use the either the /L parameter or the CARFileSize and EnableCARFileSize options to set the limit for the new CAR files the utility creates. The /L parameter overrides the INI options.

In PPS, you can enter the file size limit on the ArcSplit window.

Use these options to enable this feature for both the ARCSPLIT utility and PPS:

```
< ArcSplitConfig:Example >
    CARFileSize        = 1
    EnableCARFileSize  = No
```

Option	Description
CARFileSize	Enter a number between one (1) to 14,000 to define the size for the CARFile. If you enter one (1), the system interprets that as 100,000 bytes (100 KB). If you enter 14,000, the system interprets that as 1,400,000,000 bytes (1,4000,000 KB).
EnableCARFileSize	This option turns on and off the related radio button field on the ArcSplit window in PPS and also tells both PPS and the ARCSPLIT utility whether to use the CARFileSize option. The default is No.

1581
PPS

Overriding a Date Range when Splitting an Archive

Now you can override the required date range when splitting an archive via PPS or the ARCSPLIT utility.

In PPS, the ArcSplit window now lets you choose whether to activate this feature.

When using the ARCSPLIT utility, you can include the new /O parameter to turn it on.

You can also override the date range by adding this INI setting:

```
< ArcSplitConfig:Test1 >
    OverrideDateRange = Yes
```

1594
PPS

Using the WIPField Built-in Function

You can use this built-in INI function to tell the system to substitute a value in the INI file with a value from the WIP record. This works with either PPS (AFEMAIN) or the WIP Edit plug-in.

For example, if you want the UserDict value to equal the value for ORIGUSER in the current WIP record, you would set up the following option:

```
< Spell >
    UserDict = ~WIPFIELD ORIGUSER
```

1603
PPS

Encrypting INI Values

The ~ENCRYPTED INI built-in function lets you place encrypted values in an INI file.

To get the encrypted value, you can execute the CRYRU utility. Here is an example of how you could use this utility on Windows:

```
cryruw32.exe user1
```

The result would be something like this:

```
Encrypted string (2yz76tCkk0BRiPqLJLG00)
```

You then paste the value (*2yz76tCkkoBRiPqLJLGoo*) into an INI file and use the ~ENCRYPTED INI function, as shown in this example:

```
< SignOn >
    UserID = ~ENCRYPTED 2yz76tCkk0BRiPqLJLG00
```

When Documaker Server or Docupresentment runs and gets the value of the UserID option in the SignOn control group, it will get the real value *USER1*.

NOTE: The encryption method used is proprietary.

Keep in mind these limitations:

- Only Windows and UNIX platforms are supported.
- This feature has nothing to do with secure PDF or PDF encryption.
- Almost any INI option can be encrypted.

1604
PPS

ADDING HYPERTEXT LINKS

If you are using Documaker Studio to create your form sets, you can now produce PDF or HTML output with hypertext links that will launch a web browser and open a specified web site. You can create hypertext links in these types of objects:

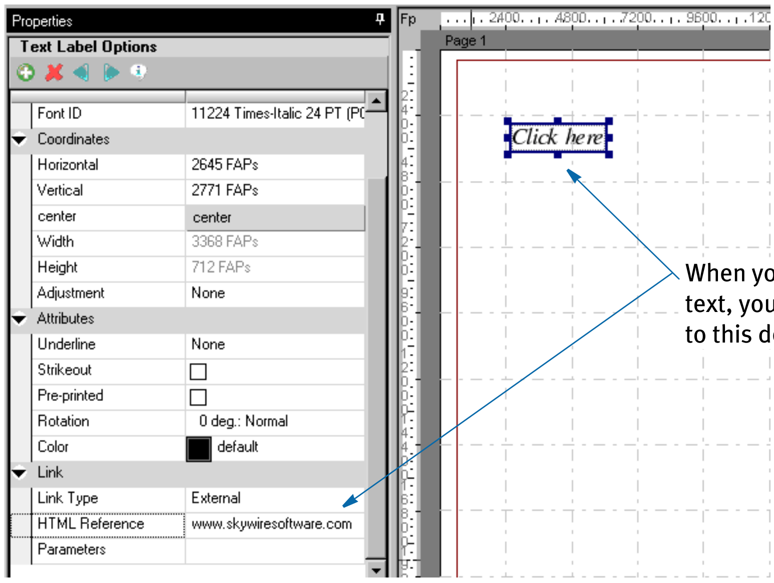
- Text labels
- Variable fields
- Logos

NOTE: Place the hypertext link at the actual field location and not at a reference to that field. For instance, if you define a variable field with a hypertext link and embed a reference to that field within a text area, the reference will not indicate the link is there.

Also keep in mind that links are assigned as an attribute of a field. The data placed in the field is not the target of the link, but rather the data you click on to go to that target.

This example shows a text label. You use the text label's Link properties to define the link.

Select the type of link and the appropriate fields for defining the link appear.



Font ID	11224 Times-Italic 24 PT (P(
Coordinates	
Horizontal	2645 FAPs
Vertical	2771 FAPs
center	center
Width	3368 FAPs
Height	712 FAPs
Adjustment	None
Attributes	
Underline	None
Strikeout	☐
Pre-printed	☐
Rotation	0 deg.: Normal
Color	■ default
Link	
Link Type	External
HTML Reference	www.skywiresoftware.com
Parameters	

When you click this text, you are taken to this destination.

Once you select *External* in the Link Type field, other fields appear so you can define the link.

Field	Description
Link Type	Select External to set up a link to a web page.
HTML Reference	Enter the address of the web page. Here is an example: `http://www.skywiresoftware.com`
Parameters	Use this field when producing HTML output to specify additional parameters to an HREF type link, such as a target frame or mouseover behavior. This example causes a web page to open in a new browser window: `target="new"`

NOTE: If you have the PDF Print Driver defined as your printer, you will see options appropriate for setting up a PDF link. For more information about licensing the PDF Print Driver, contact your Skywire Software sales representative.

If the example above were used on a text label containing the text *Click Here*, the system would generate the following command in an HTML print stream:

```
<a href="http://www.skywiresoftware.com"  target="new">Click Here</a>
```

When viewed in a browser, the HTML print stream would contain a link that appears as follows:

Click Here

You can find more information on HTML hypertext links at:

http://www.w3.org/TR/html401/

1606
PPS

Improved Support for Color and Graphics

Version 11.2 includes several enhancements that affect support for color and graphics. These include:

- The PCL print driver now supports bitmap compression
- The PostScript print driver now supports bitmap compression
- The PostScript print driver now 4-bit and 8-bit color bitmaps

NOTE: Compression is enabled by default in both the PCL and PostScript print drivers. To disable compression, add the following INI option to the PCL and PostScript printer control group:

```
< PrtType:XXX >
```

```
Compression = No
```

The PostScript print driver now supports 4-bit and 8-bit color bitmaps. It previously only supported monocolor and 24-bit bitmaps. 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 multicolor 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 monocolor (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 newer (version 11.2 or higher) 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.

1617
PPS

Faster Compression of Archives

ZLIB compression is now the default compression method for both Library Manager and archive files. The ZLIB compression method gives you smaller files and a faster compression. This compression method is described in detail on the following website:

www.ietf.org/rfc/rfc1950.txt

This document describes the ZLIB compressed data format specification version 3.3.

This affects archive, Library Manager, DPA, and DPW files.

1648
PPS

PASTING UNFORMATTED TEXT FROM THE CLIPBOARD

The new Unformatted Text option lets you paste either formatted or unformatted text from the clipboard into Text Editor. For instance, when you cut text from an application like Microsoft Word, Word places one copy of the "cut" onto the clipboard in RTF format and another copy of the "cut" in text format.

The normal paste option in Text Editor, now called Formatted Text (CTRL+V), copies in the RTF version which includes the formatting information Word included with the cut. The new Unformatted Text option (CTRL+ALT+V) copies in the text version, which will include much less formatting, depending on what Word included with the cut.

When you use the Unformatted Text option, the formatting information the initial application did include when it cut the text onto the clipboard, is retained but is applied based on the formatting in effect at the point of insertion.

For instance, there will be no font information. Instead, the Text Editor will apply the font in use at your cursor location. Be especially aware of this when you are pasting in symbols, such as those included in the Wingdings font. Some characters are unique to some fonts.

Tabs are another example. If the original application included tabs in the text it cut to the clipboard, those tabs are retained when the Text Editor pastes that text into your text area, however, the tab definitions used are those already defined for your text area -- not the ones defined in the original application.

Unformatted also means that none of the original paragraph information remains. You will not have hanging indents, centered, right-format, or similar paragraph attributes.

NOTE: You can easily see what will be retained when you use the Unformatted Text option by simply pasting the text into Windows Notepad before you paste it into Text Editor. By default, Notepad pastes in the text version and not the RTF version.

1674
PPS

REMEMBERING THE SIZE OF YOUR WINDOW

Now you can have the system remember the size of your window when you exit so the next time you log onto PPS the window will be sized to those dimensions.

To have PPS remember the size of your window, set the Maximized and SaveWindowSize options as shown here:

```
< Control >
    Maximized      = No
    SaveWindowSize = Yes
```

Option	Description
Maximized	Enter Yes to have the system always maximize your PPS window when you start the system. Enter No if you do not want your PPS window maximized.
SaveWindowSize	Enter Yes to have the system store the dimensions of the PPS window when you exit so PPS will open to the same window dimensions the next time you start the system. Enter No if you do not want your PPS window dimensions stored.

1690
PPS

Using LZW Compression in TIFF Files

PPS now supports TIFF files compressed using the Lempel-Ziv-Welch method (LZW compression). There is nothing you have to do to enable this feature.

1691
PPS

Keeping the Contents of a Text Area Together

You can use the new Must Fit on Page attribute to tell the system to keep the contents of a text area together on a page. For instance, if other objects on a page push a text area with the Must Fit attribute enabled into a position where it spans a page break, the system will move the text area to the next page to keep it together.

This attribute also works with the Can Grow attribute and, to a lessor extent, the Can Span attribute. For instance, if you set the Must Fit on Page and Can Grow attributes for a text area, that text area can expand in size and if it no longer fits on the current page without being split, the system moves it to the next page to keep it together.

Note: The text area would only split if it also specified the Can Span option. Without the Can Span option, the text area will continue to grow past the page boundary. That is normal behavior and is not a result of the Must Fit on Page attribute.

Suppose you have two text areas A and B on a page defined as *Can Grow*. Further suppose that text area B also defines the *Must Fit* attribute. If text area A expands in size to the point it pushes text area B to intersect with the page boundary, then text area B moves in its entirety to the next page. This happens despite the Must Fit option being set on text area B. When a text area grows, objects are pushed to the next page and when the text area shrinks, the objects are pulled back.

However, suppose A grew to a point where B is very close to the bottom of the page. Now should B increase in size to where it encounters the boundary, the entire text area moves to the next page because it defines the Must Fit attribute and no longer fits on the page where it was defined. Without the Must Fit attribute, the text area would have overgrown the boundary and continued on the same page. Having the Can Span attribute (without Must Fit) means that the text area would have split to the next page at the boundary, leaving a portion of itself on the original page.

Once moved to a new page, if text area B should decrease in size to where it would fit on the prior page, it will not move backwards. This is because the Must Fit option only considers the relationship the text area has to the page on which it resides. Said another way, the Must Fit option only moves a text area forward to the next page when it encounters the lower boundary of the page. To have the text area move backwards requires that a previous text area shrink to draw the object back to that page.

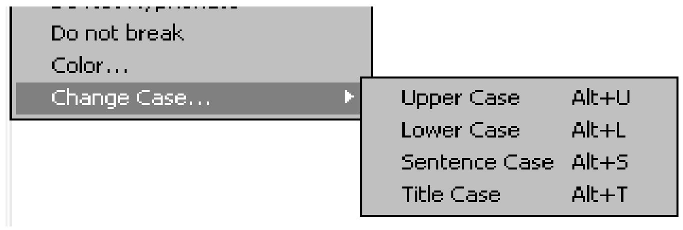
In a production setting, this should not be a problem, because the system populates the form set with data in a sequential (forward) fashion. This means that text areas are already fixed in size or grown when embedded data is assigned. Text does not typically shrink due to subsequent data mappings. When initially designing your templates, however, keep in mind the Must Fit attribute does not determine whether the text area will fit on prior page.

NOTE: The CanSplitImage rule ignores the Must Fit attribute. If an image using the CanSplitImage rule encounters on a page boundary a text area that contains the Must Fit and Can Span options, the text area is split without regard to the Must Fit option. This behavior preserves to the purpose of the CanSplitImage rule, which lets you reduce the amount of blank space at the bottom of a page.

1697
PPS

CHANGING THE CASE

Now you can highlight text and use the Change Case option on the Format menu in the Text Editor to change the case of the text. You have these choices:



Option	Description
Upper Case	All letters are capitalized. Here is an example: ALL LETTERS ARE CAPITALIZED.
lower Case	All letters are lower cased. Here is an example: all letters are lower cased.
Sentence Case	The first letter in the highlighted text is capitalized and the remaining letters are lower cased until the system finds a period, exclamation point, or question mark. Here is an example: The first letter is capitalized in each sentence. This affects all sentences.
Title Case	The first letter of each word is capitalized. Here is an example: The First Letter In Each Word Is Capitalized.

These options are applicable when you are working in a multi-line text variable field using the Text Editor in PPS or the Text Editor plug-in.

1717
PPS

Using the Mouse Wheel to Scroll

Now you can use the mouse wheel to scroll up or down on a page. You cannot use the wheel to move between pages. Scrolling this way does not change the current edit field, instead it is the same as clicking on the up or down arrows on the scroll bar.

1731
PPS

Copying Text to the Clipboard

You can now copy the text from a text area or text label to the clipboard when using PPS. This is in addition to the standard FAP object copy and paste.

This lets you, for instance, copy text from an image and paste it into another application such as a word processor or an email program.

1732
PPS

Recognizing Hyphenation Keywords in RTF Files

The system now supports automatic hyphenation in RTF files. For instance, if you are using Microsoft Word 2002 and you have turned on hyphenation (Tools, Language, Hyphenation) and saved the file as an RTF file, the system now recognizes the automatic hyphenation Word included in the RTF file.

1766
PPS

DISPLAYING THE REQUIRED FORMS

Use this new INI option to change the way you select forms on the Forms Selection window:

```
< FormSelection >
   SelRequiredFormOnUpdate = Yes
```

Option	Description
SelRequiredFormOnUpdate	Use this option if you would like the required forms on the Forms Selection window, pre-selected when you are editing a policy from WIP. During the normal form selection process, picking a new group (LOB) does not automatically select required forms. Use this option if you would like the Required Forms to be selected for new group (LOB) selections. If you first unselect a group (LOB) and then reselect it, this tells the system to mark as selected all the required forms in that group when using this feature.

NOTE: This applies to both selecting forms in PPS and via the WIP Edit plugin.

1768
PPS

SELECTING MULTIPLE TERSUB PARAGRAPHS

Now you can select more than one paragraph from the TERSUB table at a time. To select multiple paragraphs, use the SHIFT and CTRL keys with your mouse to select the paragraphs you need. Then click the Add button.

1770
PPS

SELECTING FORMS

In prior releases, selecting a new group (line of business) would also cause the required forms to be reselected in groups you had already selected. This meant that if you selected the first group and turned off some of the required forms, selecting a second group would restore the forms you turned off in the first group.

Now PPS no longer reselects the required forms in existing groups if you add another group. And if you deselect a group and then reselect it, the required forms are checked again.

1796
PPS

Embedding non-Logo Bitmap Files

The system now embeds non-LOG format bitmap files into a multi-line text field. Before, the system did not embed non-LOG format bitmaps, such as BMP or JPG files but instead referenced the location of that file. If you then passed the WIP or archive form set to another user who only had a copy of the basic master resource library (MRL), which did not include a copy of the bitmap, that user would not see the graphic.

Now if you insert a bitmap that is not a LOG type, the system embeds the data into the form itself. This means the external file is no longer necessary from that point on. So, if you pass the WIP or archived form set to someone who has the basic MRL, that user sees the bitmap even if he or she does not have the actual source file.

Note: The system does not embed LOG type bitmaps because those are Skywire Software standard format graphics files and would normally be stored in the MRL.

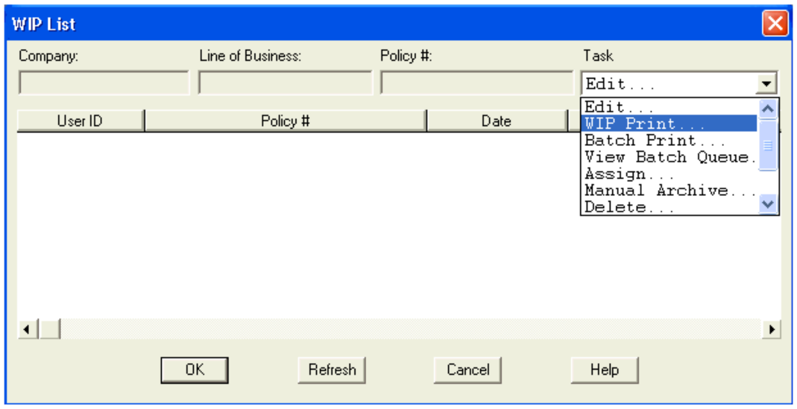
For instance, let's say you are entering insurance claims in PPS or iDocumaker. You begin by picking the appropriate form resources. On one of the forms you have a text area where you can insert a picture of the crashed car or house damage. You would do this by referencing the BMP or JPG file you took with your camera. Since this BMP or JPG file is not part of the MRL, other users would not have it and would not see the picture when the form is viewed via WIP or retrieved from archive. Now that the system embeds the data for the BMP or JPG file, other users will be able to see the picture.

1803
PPS

Simple WIP Print

You can now print a form set from the WIP List window. The new print feature does not change the form set's status nor does it archive or delete it — in this way it differs from the Batch Print option which moves things along in the workflow.

You can choose this option from the WIP List window. From the window that appears, you can select multiple form sets to print.



For new installations, this feature is automatically enabled. You can activate this feature in existing installations by removing the semicolon from the WIP Print line in the MEN.RES file, as shown here:

Remove this semicolon to turn on this feature

```
POPUP "W&ip" 257 "Wip menu"
 BEGIN
    MENUITEM "&Wip List..." 297 "AFEW32->AFECombineWipDlgs""Wip 
Functions"
;   MENUITEM "&Edit..." 270  "AFEW32->AFEUpdate" "Edit existing 
document"
;   MENUITEM "&Batch Print..." 271 "AFEW32->AFEPrint" "Batch Print 
existing document"
;   MENUITEM "&View Batch Queue..." 272  "AFEW32->AFEViewBatch" 
"View existing document in Batch Queue"
;   MENUITEM "&Assign..." 273  "AFEW32->AFEAssign" "Assign existing 
document"
;   MENUITEM "&Manual Archive..." 274  "AFEW32->AFEArchive" 
"Archive existing document"
;   MENUITEM "&Delete..." 275 "AFEW32->AFEPurge" "Delete existing 
document"
;   MENUITEM "S&tatus..." 276 "AFEW32->AFEStatus" "Change Status for 
existing document"
;   MENUITEM "&Wip Print..." 298 "AFEOS2->AFEWIPPrint" "Wip Print"
    SEPARATOR
    MENUITEM "&Save" 103  "AFEW32->AFESaveClose" "Save document to 
wip and close desktop"
    SEPARATOR
    MENUITEM "Se&nd..." 281 "AFEW32->AFEPackage" "Route documents 
via mail"
    MENUITEM "&Receive..."  282 "AFEW32->AFEReceive" "Receive 
documents via mail"
;   SEPARATOR
;   MENUITEM "Change In &Use..." 201 "AFEW32->AFEInUse" "Edit In Use" 
0
  END
```

1804
PPS

ROTATING A DYNAMIC BITMAP

When dynamically adding bitmaps — like multi-page TIFF files— of scanned images, occasionally you may have incorrectly inserted one or more of the pages and those pages appear upside down. This change lets you rotate the dynamic bitmap to the correct position from PPS or the WIP Edit plug-in.

To rotate dynamic bitmaps, there are two menu functions (with optional toolbar buttons) you can activate. One is a Rotate (90 degrees) and the other is Flip (180 degrees). Here is an example from the MEN.RES file:

```
MENUITEM "Rotate dynamic bitmap" 1090 "NULL" "NULL"
MENUITEM "Flip dynamic bitmap" 1091 "NULL" "NULL"

TOOL 61 1090 DISABLED BUTTON
TOOL 46 1091 DISABLED BUTTON
```

You cannot change the menu and tool IDs. 1090 is the ID for 90 degree rotation and 1091 is the ID for flipping the bitmap 180 degrees (upside down). You can include one or both options if you like. The toolbar options are optional.

Each time you click or choose Rotate, the bitmap is rotated 90 degrees. So clicking Rotate twice produces the same result as clicking Flip once. The system rotates the bitmap counter-clockwise.

Since these are defined via the MEN.RES file, you control where you want the options to appear on the menu and what text appears. The tool tip information is, however, built into the system and cannot be customized via the MEN.RES file.

This only applies to dynamically added bitmaps that are also embedded into the transaction. This means that normal logos (LOGs) in your FAP files are not rotated when you use these new options.

If you choose one of these option on a page where no dynamic bitmaps occur, you will see the following message:

```
No dynamic bitmap was found on this page.
```

Keep in mind these limitations:

- Because of the way that page orientations are determined, the ability to change the page to landscape on rotation only works if this is the first image on the page. In cases where you might have other images, including dummy images, inserted before the one that contains the bitmap, the page may not rotate with the bitmap.
- When the first page defines other FAP files before the image that contains the bitmap, rotating this one image will not cause the entire page to be changed to landscape.

  This is really only an issue if you plan to do a 90 degree rotation and expect the orientation of the page to change as well. The Flip option (180 degrees) does not require any page changes.
- You can overlay full size images on top of one another using SetOrigin. Any image can contain text or other objects that you might not want to print in a landscape fashion, even if the bitmap image does rotate.
- Not all the FAP files on a given page have to print for each recipient. The first printable image on the page, for a given recipient, determines the orientation of the page and its size. This means the view of the page may differ per recipient — not only in portrait or legal orientation, but also the page sizes can change for the same page with different recipients. For instance, the same page when printed for one recipient might be Letter+Portrait, but print as Legal+Landscape for a different recipient.

NOTE: It is not possible for the system to know whether the page will display or print successfully. The best plan would be to make sure when using this feature, you create your form containing the bitmap with a single FAP per page.

1808
PPS

MISCELLANEOUS ENHANCEMENTS

The following changes were made to PPS:

PCR 15354 - Added a warning message which appears if the function was not found for AFECallCheckDupFormHook.

PCR 17065 - The Form Selection window's search of form name or description columns now default the partial match checkbox. In addition, the application remembers the last setting of the checkbox until you log off.

PCR 18567 - Now you can customize the Routing Slips window to hide all buttons except Ok, Cancel, and Help, so users cannot modify or the delete slips created by the system administrator. Use these INI options to hide or reveal these buttons:

```
< Routing_Slip >
   HideEdit    = Yes
   HideDelete  = Yes
   HideInsert  = Yes
   HideScript  = Yes
   HideMove    = Yes
   HideRename  = Yes
```

Option	Description
HideEdit	Enter Yes to hide the Edit button. The default is No.
HideDelete	Enter Yes to hide the Delete button. The default is No.
HideInsert	Enter Yes to hide the Insert button. The default is No.
HideScript	Enter Yes to hide the Script button. The default is No.
HideMove	Enter Yes to hide the Move button. The default is No.
HideRename	Enter Yes to hide the Rename button. The default is No.

1811
PPS

Changing the Default Field Color

Use these new INI options to tell the system how to display fields before and after data is assigned. If you omit these options, the field color defaults to red.

```
< Control >
   DefaultFieldColor = (255,0,0,0)
   DefaultPlaceColor = (255,0,0,0)
```

Option	Description
DefaultFieldColor	This option defines the color to use to show data locations on the images. Specify a color using a RGB (red-green-blue) configuration followed by a control byte. Each color value is a number between 0 and 255 to assign the intensity of that color. The control byte will always be zero. The combination of RGB values is used to achieve the final color shown on the monitor.
DefaultPlaceColor	This option lets you show the field placeholders (for fields that have no data assigned) in another color. If you omit this option, the placeholder color is the same as the DefaultFieldColor. Specify a color using a RGB (red-green-blue) configuration followed by a control byte. Each color value is a number between 0 and 255 to assign the intensity of that color. The control byte will always be zero. The combination of RGB values is used to achieve the final color shown on the monitor.

NOTE: If you do not enter the RGB structure as defined and instead specify a long numeric value, the system assumes your entry represents a single (resolved) color value. For instance,

```
DefaultFieldColor = 16711680
```

is the resolved value for red.

Also note that these INI options only affect those fields using the default field color. If the color is overridden at the field or image level (for all fields on the image), these values will not apply.

1813
PPS

SHOWING BLANK FIELDS AS NULL FIELDS

Normally, only those fields that have never been assigned a value show the place holder. Fields with values — even an empty string — show the text assigned. You can use the new ShowEmptyFieldAsNull option to override the behavior of fields that have been assigned an empty string as a value.

```
< Control >
    ShowEmptyFieldAsNull = Yes
```

Option	Description
ShowEmptyFieldAsNull	Enter Yes to have the system show a field with an empty string as it would a field which has never had data entered into it. This means the place holder rectangle normally reserved for fields with null data will appear for fields containing an empty string. The default is No.

1824
PPS

Mapping Old Archive Keys to Current Document Keys

You can now map old keys to new keys in PPS using the Key1Table and Key2Table control groups.

For instance, if you have changed the Key1 or Key2 fields in your FORM.DAT file, but would like to retrieve data from archive or import data through the Form Selection window using the old Key1 and Key2 values, now you can.

To do this, you use the following control groups for Key1 and Key2 fields.

```
< Key1Table >
    OldCompany = NewCompany
< Key2Table >
    OldLOB = NewLOB
```

You do not have to add both groups if only one key was affected. Just add the appropriate control group and the key to which you would like the mapping done. In this example, the NewCompany and NewLOB options represent the new names of the keys. The values OldCompany and OldLOB represent the old names of the keys.

If you are using Documaker Studio, you can handle the alias mapping at a global level by adding them to the BDF file. By using the BDF file, any workstation or program using the library automatically inherits the necessary information.

If you are not using Documaker Studio, add the alias mappings to the INI file using Key1Table and Key2Table control group, as shown in the example above. If the users share a common INI file, such as the FSISYS.INI file, you can make this change in a global fashion. If your users do not share INI settings, you will have to make these changes in each user's FSIUSER.INI file.

1834
PPS

Activating the Spell Check Option

The Spell Check option is now automatically included in the MEN.RES file so this option appears in the application unless you specifically remove it. Here is an example of the line in the MEN.RES file that adds the Spell Check option:

```
MENUITEM "&Spell Check Fields..." 1087 "NULL" "NULL"
```

You can remove this option from the menu by commenting it out.

1868
PPS

Automatically Re-Paginating Images

PPS now better handles pages comprised of multiple floating images by automatically re-paginating the form set when an image changes size.

NOTE: This applies to both PPS and the WIP Edit plug-in. Documaker Server waits until the end of its processing to handle pagination.

Previously, PPS did not re-paginate a form when a page consisted of multiple images and one of those images changed size. Now, if a text area causes an image to grow or shrink — due to embedded fields changing or because it was a multi-line text field — the form that contains the image automatically re-paginates.

If you designed the form using Documaker Studio, the positioning information you established is reapplied to the form each time changes in a text area cause the dimensions of an image to change. You can, however, use the AutoPagination option to disable automatic re-pagination:

```
< Control >
    AutoPagination = Yes
```

Option	Description
AutoPagination	Enter No if you do not want the system to automatically re-paginate when image dimensions change. The default is Yes.

Keep in mind...

- The system will honor positioning information designed into the form via Documaker Studio. Positioning information is stored via SetOrigin rules.
- If a form consists of multiple images on a page, but those images comprise more space than defined for the page size, the system automatically paginates that page and moves the images that did not fit to a new page.
- If an image grows to push another image such that its positioning rule causes it to encroach on a defined footer or the bottom of the page, that image is moved to the next page and the entire form will be have the SetOrigin rules reapplied.
- When designing a form, avoid having a footer image that uses a relative position. This ultimately means there can only be one image on the page that is not a header or footer. Footers, typically should be placed using a rule that makes sure it has a relationship to the bottom of the page.
- When images shrink (due to text area shrinking) an image from the next page may be brought back to the current page. In other words, images can not only flow to the next page, but they can come back when space allows.
- Remember that the positioning (SetOrigin) defined in the form is applied. So although there may appear to be a space large enough to hold an image, you also have to account for any additional adjustments applied by the SetOrigin rules.

  For example, suppose you have an image that is 2 inches in height, but the SetOrigin rule for that image specifies a relative placement 1/2 inch from the previous image. In this case, 2 1/2 inches of space is required for that image to fit on the page. If there is less than 2 1/2 inches remaining before encountering a footer image or the bottom of the page layout, then that image will move to the next page.

- Images can only flow to and from pages that were created during pagination. If a page was specifically designed into the form via Documaker Studio, then no images will move onto that page from a prior page. Images can only move to or from pages that were created by overflowing their defined page.

1872
PPS

Storing the WIP Index and WIP Data in Database Tables

Documaker software now lets you store WIP data in a database to enhance security and the commit and rollback process. This also enhances performance when using multiple keys to access transactions in intra- and internet applications. To implement this enhancement, changes were made that affect Documaker Server, PPS, and the WIP Edit plug-in.

NOTE: The WIPDATA.DFD file is a new file.

Skywire Software recommends that you only use uppercase for table and column names when storing information in a database. For instance, avoid CustomerName, Customername, or customername and instead use CUSTOMERNAME.

Database management systems (DBMS) vary in how they handle case issues so it is best to standardize on uppercase. With version 11.2, all column names must be in uppercase.

The standard WIP system uses an xBase index and a flat file (NAFILE, POLFILE) format. For WIP transactions Documaker uses a default WIP.DFD when placing transactions into the default xBase system.

When you change database systems, to implement this new feature enhancement you will need a physical WIP.DFD and WIPDATA.DFD file.

The files will need to be modified to add a UNIQUE_ID field.

You can install our sample RPEX1 master resource library (MRL) which includes utilities to create WIP.DFD and WIPDATA.DFD files for MySQL, Microsoft SQL (MS SQL), DB2, and Oracle.

Similarly, Skywire Software includes the WIP.DFD and WIPDATA.DFD files for each database type in this document. You can copy and paste to a text file, call them WIP.DFD and WIPDATA.DFD and place them into your MRL's DefLib directory.

You also need to establish connection with your database and set up database files for WIP to talk to.

You can modify your INI files to automatically create the tables and fields for you.

Or, you can use the DFD2DDL utility (DFD2DDLW.EXE) to read the WIP.DFD and WIPDATA.DFD files and create DDL files for your database administrator to use in setting up the tables.

There are INI file changes you must make to implement this feature. These changes are discussed in detail below. You will also find instructions for running a DAL script that can update the INI files for you. A listing of the DAL script is included in this document. The actual DAL script (INI4WIP.DAL) is included in the sample RPEX1 MRL in the rpex1\deflib directory.

You can now store WIP indexes and WIP data in the following types of databases using the following ODBC drivers:

Databases	DB2 version 7.2.2. on AIX Oracle version 8.1.7 on Solaris Microsoft SQL version 8.0 MySQL
ODBC drivers	IBM ODBC driver version 7.0.00.65 Oracle ODBC driver version 8.01.07

Using this Feature

To use this feature, your master resource library (MRL) should be in the following structure:

- The MRL is in: \fap\mstrres\xxxxx (where *xxxxx* is an MRL name, such as RPEX1)
- The FSIUSER.INI and FSISYS.INI are in: \fap\mstrres\xxxxx
- The WIP.DFD and WIPDATA.DFD files are in: \fap\mstrres\xxxxx\deflib
- The DLL directory is: \fap\dll

You must also have these prerequisites:

- DB2, SQL, or Oracle supported database has been installed.
- A database client with a native or ODBC driver has been configured and tested.
- Documaker Server version 11.1 patch 22 or PPS version 11.3 or higher has been installed.

NOTE: Version 11.1, patch 22 includes version 11.2, feature 1872.

Setting Up Your Database

Here is an overview of the steps you will perform. These steps are explained in detail on the following pages.

- (DB2 only) Issue a DB2 Bind.
- Create a WIP.DFD file in your \fap\mstrres\xxxxx\deflib\ directory and make a backup.
- Create the WIPDATA.DFD file in your \fap\mstrres\xxxxx\deflib\ directory and make a backup.
- Modify your INI files to connect Documaker with the database.

- Test using PPS or Documaker Server.

Here are the detailed steps:

1 Issue a DB2 bind. This step is only necessary if you are using DB2.

Open a DB2 Command window and type the following to bind the database:

```
db2 connect to DATABASENAME user USERNAME using PASSWORD
```

Where *DATABASENAME* is the database name for the WIP index files and *USERNAME* and *PASSWORD* represent your user name and password.

```
db2 bind c:\fap\dll\db2lib.bnd collection 'DB2DB'
```

Decide upon a collection name, such as *DB2DB*. In the following steps you will update your INI file with this collection name:

```
< DBHandler:DB2 >
    CurrentPackageSet= DB2DB)
```

2 Create a WIP.DFD file in your \fap\mstrres\xxxxx\deflib\ directory and make a backup.

- If you already have a WIP.DFD file, create a backup copy. If you do not have a WIP.DFD file, see the example below. You can copy the example into a text file called WIP.DFD and use this as your default file.

or

- At a command prompt, using the sample RPEX1 MRL resources, create a WIP.DFD file by executing this command from the \fap\mstrres\rpex1 directory:

```
dalrw32.exe /x=.\deflib\WIP.DAL
```

Example WIP.DAL file

The WIP.DAL file is located in the \rpex\deflib directory. Here is an example of the WIP.DAL file:

```
***Beginning of WIP.DAL file***
* DAL script to add the UNIQUE_ID to the WIP.DFD file required for
* a database
rc = LoadINIFile("FSI", ".\deflib\wip.dfd")
*msg( "Return code = " & rc )
if (rc = 0)
   msg("wip.dfd was not found!")
   return(0)
end

PutINIString("FSI", "FIELDS", "FIELDNAME", "UNIQUE_ID")
PutINIString("FSI", "FIELD:UNIQUE_ID", "INT_TYPE", "CHAR_ARRAY")
PutINIString("FSI", "FIELD:UNIQUE_ID", "EXT_TYPE", "CHAR_ARRAY")
PutINIString("FSI", "FIELD:UNIQUE_ID", "INT_LENGTH", "32")
PutINIString("FSI", "FIELD:UNIQUE_ID", "EXT_LENGTH", "32")
PutINIString("FSI", "FIELD:UNIQUE_ID", "REQUIRED", "N")
PutINIString("FSI", "FIELD:UNIQUE_ID", "KEY", "N")
PutINIString("FSI", "FIELD:UNIQUE_ID", "KEY", "N")
PutINIString("FSI", "KEY:UNIQUE_ID", "EXPRESSION", "UNIQUE_ID")
PutINIString("FSI", "KEY:UNIQUE_ID", "FIELDLIST", "UNIQUE_ID")
SaveINIFile("FSI", ".\deflib\wip.dfd")
***END of WIP.DAL file***
```

Example WIP.DFD file

After you run the WIP.DAL script or after you manually create the WIP.DFD file, your WIP.DFD file should look identical to the following:

```
***Beginning of WIP.DFD file***
< FIELDS >
; UNIQUE_ID field is not part of the standard WIP.DFD, must add.
FIELDNAME = UNIQUE_ID
FIELDNAME = KEY1
FIELDNAME = KEY2
FIELDNAME = KEYID
FIELDNAME = RECTYPE
FIELDNAME = CREATETIME
FIELDNAME = ORIGUSER
FIELDNAME = CURRUSER
FIELDNAME = MODIFYTIME
FIELDNAME = FORMSETID
FIELDNAME = TRANCODE
FIELDNAME = STATUSCODE
FIELDNAME = FROMUSER
FIELDNAME = FROMTIME
FIELDNAME = TOUSER
FIELDNAME = TOTIME
FIELDNAME = DESC
FIELDNAME = INUSE
FIELDNAME = ARCKEY
FIELDNAME = APPDATA
FIELDNAME = RECNUM

< FIELD:UNIQUE_ID >
; This section is not part of the standard WIP.DFD, must add
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 32
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 32
KEY = N
REQUIRED = N
< FIELD:KEY1 >
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 30
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 30
KEY = N
REQUIRED = N
< FIELD:KEY2 >
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 30
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 30
KEY = N
REQUIRED = N
< FIELD:KEYID >
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 20
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 20
KEY = N
```

```
REQUIRED = N
< FIELD:RECTYPE >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 3
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 3
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:CREATETIME >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:ORIGUSER >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:CURRUSER >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:MODIFYTIME >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:FORMSETID >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 32
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 32
INT_PRECISION = 0
KEY = N
REQUIRED = N
```

```
< FIELD:TRANCODE >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 2
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 2
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:STATUSCODE >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 2
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 2
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:FROMUSER >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:FROMTIME >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:TOUSER >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:TOTIME >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 8
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 8
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:DESC >
```

```
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 30
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 30
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:INUSE >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 1
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 1
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:ARCKEY >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 18
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 18
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:RECNUM >
EXT_TYPE = NOT_PRESENT
EXT_LENGTH = 0
EXT_PRECISION = 0
INT_TYPE = LONG
INT_LENGTH = 4
INT_PRECISION = 0
KEY = N
REQUIRED = N
< FIELD:APPDATA >
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 150
EXT_PRECISION = 0
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 150
INT_PRECISION = 0
KEY = No
REQUIRED = No
< KEYS >
KEYNAME = UNIQUE_ID
KEYNAME = DOCTAG
KEYNAME = KEY2TAG
KEYNAME = KEYIDTAG
KEYNAME = USERTAG
< KEY:UNIQUE_ID >
EXPRESSION = UNIQUE_ID
FIELDLIST = UNIQUE_ID
< KEY:DOCTAG >
EXPRESSION = KEY1+KEY2+KEYID+RECTYPE
```

```
FIELDLIST = KEY1,KEY2,KEYID,RECTYPE
< KEY:KEY2TAG >
EXPRESSION = KEY2
FIELDLIST = KEY2
< KEY:KEYIDTAG >
EXPRESSION = KEYID
FIELDLIST = KEYID
< KEY:USERTAG >
EXPRESSION = CURRUSER

FIELDLIST = CURRUSER
***End of WIP.DFD file***
```

If you are using a custom WIP.DFD file, you must modify the WIP.DFD file for database support. Make sure no field listed is missing otherwise unpredictable results will occur.

For DB2, we recommend you change all Keys and Required options to No. While you can set these to Yes, you have to then be sure there is data for that field to avoid errors. Here is an example:

```
Key         = No
Required    = No
```

3 Create the WIPDATA.DFD file in your \fap\mstrres\xxxxx\deflib\ directory and make a backup.

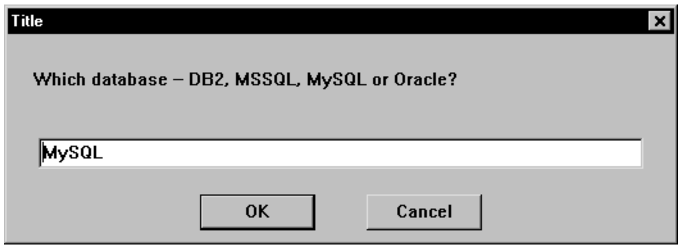
- If you already have a WIPDATA.DFD file, create a backup copy. If you do not have a WIPDATA.DFD file, see the following example. You can copy the example into a text file called WIPDATA.DFD and use this as your default file.

or

- At a command prompt, using the sample RPEX1 MRL resources, you can create a WIPDATA.DFD file by entering this command from the \fap\mstrres\rpex1 directory:

  ```
  dalrw32.exe /x=.\deflib\WIPDATA.DAL
  ```

  The following prompt appears:

  Which database – DB2, MSSQL, MySQL or Oracle?

  MySQL

  Enter the appropriate database type and click Ok.

Example WIPDATA.DAL file

The WIPDATA.DAL file is located in the rpex\deflib directory. If you do not have a WIPDATA.DAL file, here is an example:

```
***Beginning of WIPDATA.DAL file***
* DAL script to produce the wipdata.dfd for different database
rc = LoadINIFile("FSI", ".\deflib\wipdata1.dfd")
*msg( "Return code = " & rc )
```

```
if (rc = 0)
   msg("wipdata1.dfd was not found!")
   return(0)
end

myDB = INPUT("Which database -- DB2, MSSQL, MySQL or Oracle? ", , ,
"MySQL" )
if (myDB = "Oracle")
PutINIString("FSI", "FIELD:CARDATA", "INT_LENGTH", "1950")
PutINIString("FSI", "FIELD:CARDATA", "EXT_LENGTH", "1950")
end

if ((myDB = "MSSQL") OR (myDB = "MySQL"))
PutINIString("FSI", "FIELD:CARDATA", "INT_TYPE", "BLOB")
PutINIString("FSI", "FIELD:CARDATA", "EXT_TYPE", "BLOB")
PutINIString("FSI", "FIELD:CARDATA", "INT_LENGTH", "8")
PutINIString("FSI", "FIELD:CARDATA", "EXT_LENGTH", "8")
end

SaveINIFile("FSI", ".\deflib\wipdata.dfd")
***END of WIPDATA.DAL file***
```

Example WIPDATA.DFD file

After you run the WIPDATA.DAL script or after you manually create the WIPDATA.DFD file, your WIPDATA.DFD file should be identical to this example:

```
***Beginning of WIPDATA.DFD file***
< FIELDS >
FIELDNAME = FORMSETID
FIELDNAME = SEQ_NUM
FIELDNAME = CONT_FLAG
FIELDNAME = TOTAL_SIZE
FIELDNAME = CARDATA
< FIELD:FORMSETID >
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 32
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 32
KEY = N
REQUIRED = N
< FIELD:SEQ_NUM >
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 5
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 5
KEY = N
REQUIRED = N
< FIELD:CONT_FLAG >
INT_TYPE = CHAR_ARRAY
INT_LENGTH = 1
EXT_TYPE = CHAR_ARRAY
EXT_LENGTH = 1
KEY = N
REQUIRED = N
< FIELD:TOTAL_SIZE >
INT_Type = LONG
INT_Length = 4
```

```
EXT_Type = LONG
EXT_Length = 4
Key = No
Required = No

< FIELD:CARDATA >
INT_Type = VARCHAR
INT_Length = 3950
EXT_Type = VARCHAR
EXT_Length = 3950
Key = No
Required = No

< Keys >
KeyName = FORMSETID
KeyName = SEQ_NUM
< KEY:FORMSETID >
Expression = FORMSETID
FieldList = FORMSETID
< KEY:SEQ_NUM >
Expression = SEQ_NUM
FieldList = SEQ_NUM
***End of WIPDATA.DFD file***
```

Changing the CARDATA field

This is necessary if you manually need to change the WIPDATA.DFD file. This field is set for you if you use the DAL script and select the appropriate database type.

If you are using MS SQL and MySQL, change the CARDATA field as shown here:

```
< Field:CARDATA >
    INT_Type    = BLOB
    INT_Length  = 8
    EXT_Type    = BLOB
    EXT_Length  = 8
    Key         = No
    Required    = No
```

If you are using an Oracle native driver, change the CARDATA field as shown here:

```
< Field:CARDATA >
    INT_Type    = VARCHAR
    INT_Length  = 1950
    EXT_Type    = VARCHAR
    EXT_Length  = 1950
    Key         = No
    Required    = No
```

If you are using a DB2 native or ODBC driver, change the CARDATA field as shown here:

```
< Field:CARDATA >
    INT_Type    = VARCHAR
    INT_Length  = 3950
    EXT_Type    = VARCHAR
    EXT_Length  = 3950
    Key         = No
    Required    = No
```

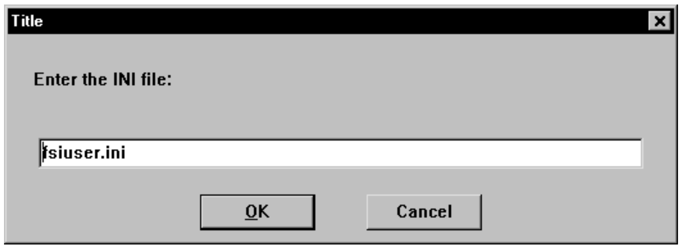
Copy the WIP.DFD and WIPDATA.DFD files into the MRL's DefLib directory, such as \fap\mstrres\rpex1\deflib\wip.dfd.

4 Modify your INI files to connect Documaker with the database.

You can automatically update your (FSIUSER or CONFIG) INI file by executing this command from the \fap\mstrres\rpex1 directory:

```
dalrw32.exe /x=.\deflib\INI4WIP.DAL
```

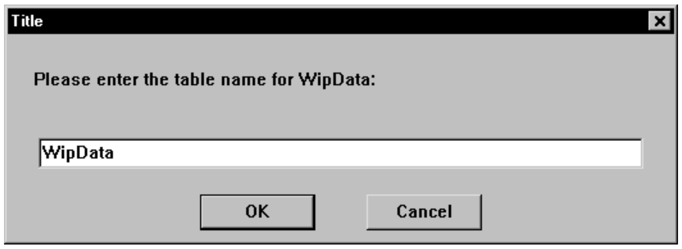
Make sure you run this DAL script in the directory where your INI file resides. The following prompt appears:



Enter the INI file:

fsiuser.ini

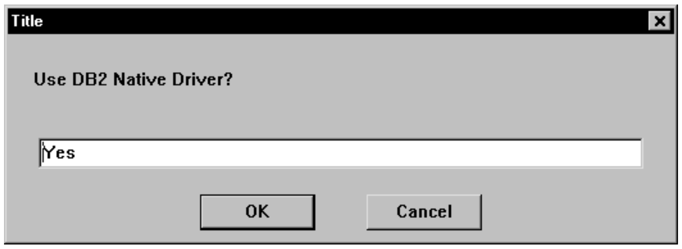
Enter the appropriate INI file name and click Ok. The following prompt appears:



Please enter the table name for WipData:

WipData

Enter the table name and click Ok. For best results, use the default table name (WIPData). The following prompt appears:



Use DB2 Native Driver?

Yes

Click Ok if you are using the DB2 Native driver to accept the default, otherwise enter **No** and click Ok to continue.

Depending on your database, you may be prompted for a database name, user ID, password, and other information.

Example INI4WIP.DAL file

If you do not have an INI4WIP.DAL file, here is an example of the script you should use:

```
***Beginning of INI.DAL file***
* DAL script to update the fsiuser.ini for database
inifile=INPUT("Enter the INI file: ",,,"fsiuser.ini")
rc = LoadINIFile("FSI", inifile)
```

```
*msg( "Return code = " & rc )
if (rc = 0)
   msg("fsiuser.ini was not found!")
   return(0)
end

* common to all
myWipData = INPUT("Please enter the table name for WipData: ", , ,
"WipData")
PutINIString("FSI", "WipData", "WipData", myWipData)
PutINIString("FSI", "WipData", "DatabaseWIP", "TRUE")
PutINIString("FSI", "WipData", "WipDataDFD", ".\deflib\wipdata.dfd")
PutINIString("FSI", "WipData", "WipDFDFile", ".\deflib\wip.dfd")

ans = INPUT("Use DB2 Native Driver? ",,,"Yes")
if (ans = "Yes")
    PutINIString("FSI", "DBHandler:DB2", ";BindFile", "")
  PutINIString("FSI", "DBHandler:DB2", ";CurrentPackageSet",
"collection_name");
    PutINIString("FSI", "DBHandler:DB2", "CommitEvery", "0")
    PutINIString("FSI", "DBHandler:DB2", "Connect", "Yes")
    PutINIString("FSI", "DBHandler:DB2", "CreateIndex", "No")
    PutINIString("FSI", "DBHandler:DB2", "CreateTable", "No")
    mydb = INPUT("Name of the database? ",,,)
*   check for not empty

    PutINIString("FSI", "DBHandler:DB2", "Database", mydb)
    myuserid = INPUT("userid? ",,,)
    PutINIString("FSI", "DBHandler:DB2", "UserID", myuserid )
    mypasswd = INPUT("password? ",,,)
    PutINIString("FSI", "DBHandler:DB2", "Passwd", mypasswd )
    PutINIString("FSI", "DBTable:WipData", "DBHandler", "DB2")
    PutINIString("FSI", "DBTable:WipData", "UniqueTag", "FORMSETID")
    SaveINIFile("FSI", "db2user.ini")
    return('db2')
end

ans = INPUT("Use Oracle Native Driver? ",,,"Yes")
if (ans = "Yes")
    PutINIString("FSI", "DBHandler:ORA", "CreateIndex", "No")
    PutINIString("FSI", "DBHandler:ORA", "CreateTable", "No")
    mydb = INPUT("Name of the database? ",,,)
*   check for not empty

    PutINIString("FSI", "DBHandler:ORA", "Database", mydb)
    myuserid = INPUT("userid? ",,,)
    PutINIString("FSI", "DBHandler:ORA", "UserID", myuserid )
    mypasswd = INPUT("password? ",,,)
    PutINIString("FSI", "DBHandler:ORA", "Passwd", mypasswd )
    PutINIString("FSI", "DBTable:WipData", "DBHandler", "ORA")
    PutINIString("FSI", "DBTable:WipData", "UniqueTag", "FORMSETID")
    PutINIString("FSI", "ORA_FileConvert", "Wipdata", "WIPDATA")
    SaveINIFile("FSI", "orauser.ini")
    return('oracle')
end
```

```
PutINIString("FSI", "DBHandler:ODBC", "AlwaysSQLPrepare", "Yes")
myServer = INPUT("Please enter the Data Source Name : ", , , )
*verify it is not blank
PutINIString("FSI", "DBHandler:ODBC", "Server", myServer)
PutINIString("FSI", "DBHandler:ODBC", "CreateTable", "No")
PutINIString("FSI", "DBHandler:ODBC", "CreateIndex", "No")
myUserID = INPUT("Please enter the UserID: ", , , )
PutINIString("FSI", "DBHandler:ODBC", "UserID", myUserID)
myPasswd = INPUT("Please enter the password: ", , , )
PutINIString("FSI", "DBHandler:ODBC", "Passwd", myPasswd)

PutINIString("FSI", "DBTable:Wip", "DBHandler", "ODBC")
PutINIString("FSI", "DBTable:Wip", "UniqueTag", "FORMSETID")

PutINIString("FSI", "DBTable:WipData", "DBHandler", "ODBC")
PutINIString("FSI", "DBTable:WipData", "UniqueTag", "FORMSETID")

PutINIString("FSI", "ODBC_FieldConvert", "DESC", "DESCRIPTION")

SaveINIFile("FSI", "odbcuser.ini")
return('odbc')
***END of INI.DAL file***
```

Manually updating INI files

To update your INI files manually, here are the settings you need to add. Add this control group regardless of the database type:

```
< WIPData >
    WIPData     = WIPData
    DatabaseWIP= Yes
    WIPDataDFD  = .\deflib\wipdata.dfd
    File        = WIP
    Path        = .\wip
    WIPDFDFile  = .\deflib\wip.dfd
```

Add these options if you are using the DB2 (native driver):

```
< DBTable:WIP >
    DBHandler   = DB2
    UniqueTag   = DocTag
< DBTable:WIPData >
    DBHandler   = DB2
    UniqueTag   = FormSetID
< DBHandler:DB2 >
;   BindFile    = d:\fap\dll110\db2lib.bnd
    CommitEvery= 0
    Connect     = Yes
    CreateIndex= No
    CreateTable= No
    Database    = ARCDB
    Debug       = No
    PassWd      =
    UserID      =
    CurrentPackageSet= REL112
    Class               = DB2
< DB2_FileConvert >
    WIP                 = WIP
```

Add these options if you are using the SQL (ODBC) driver:

```
< DBTable:WIP >
    DBHandler       = ODBC
    UniqueTag       = DocTag
< DBTable:WIPData >
    DBHandler       = ODBC
    UniqueTag       = FormSetID
< DBHandler:ODBC >
    CommitEvery     = 0
    Connect         = Yes
    CreateIndex     = No
    CreateTable     = Yes
    Server          = WIPDB
    Debug           = No
    PassWd          =
    UserID          =
    AlwaysSQLPrepare = Yes
<ODBC_FileConvert>
    WIP             = WIP
< ODBC_FieldConvert >
    Desc            = DESCRIPTION
```

Add these options if you are using the Oracle (ODBC) driver:

```
< DBTable:WIP >
    DBHandler       = ODBC
    UniqueTag       = DocTag
< DBTable:WIPData >
    DBHandler       = ODBC
    UniqueTag       = FormSetID
< DBHandler:ODBC >
    CommitEvery     = 0
    Connect         = Yes
    CreateIndex     = No
    CreateTable     = No
    Server          = WIPDB
    Debug           = No
    PassWd          =
    UserID          =
    AlwaysSQLPrepare= No
<ODBC_FileConvert>
    WIP             = WIP
<ODBC_FieldConvert >
    Desc            = DESCRIPTION
```

Add these options if you are using the Oracle (native) driver:

```
< DBTable:WIP >
    DBHandler       = ORA
    UniqueTag       = DocTag
< DBTable:WIPData >
    DBHandler       = ORA
    UniqueTag       = FormSetID
< DBHandler:ORA >
    Connect         = Yes
    CreateIndex     = No
    CreateTable     = No
    Database        = WIPDB
    Debug           = No
    PassWd          =
    UserID          =
    Class           = ORA
< ORA_FileConvert>
    WIP             = WIP
< ORA_FieldConvert >
    Desc            = DESCRIPTION
```

This table describes how you use the DBHandler and UniqueID fields in the DBTable:WIPData control group:

Option	Description
DBHandler	Tells the system that WIPDATA is a table in an DB2-compliant database used to store the WIP data.
UniqueTag	Use this option to assign a unique key. For example, you could enter FormSetID or DocTag, as long as it is unique.

If the name you assign to the WIP table is longer than eight characters, add the following control group and option for the name conversion to take place:

```
< DB2_FileConvert >
    WIP = FAPQA.WIP
```

If a DBA needs to create the WIP index table with all the fields defined in the WIP.DFD file and the WIPDATA table with all the fields defined in the WIPDATA.DFD, you can set the CreateTable option to No. Otherwise, set it to Yes to tell the system to create the table.

Oracle and DB2 database require that you set CreateTable to No. A DDL is required to be created and run on an Oracle and DB2 database by a DBA. The following steps have to be executed for those databases.

If you set the CreateTable option to No, you must create a DDL from the DFDs. From a command prompt, go to your xxxx\deflib directory, such as rpex1\deflib. Then execute one of these commands:

To	Enter this command
Produce the DDL	dfd2ddlw.exe /i=wip.dfd /o=wipdb.sql
Append to the DDL	dfd2ddlw /i=wipdata.dfd /o=wipdb.sql

The DBA can then execute the DDL to create WIP and WIPData tables.

Specify the same output DDL file to append the SQL statements into one file, such as WIPDB.SQL. If the database is not MS SQL, be sure to include the /D parameter to specify the database to produce the correct DDL.

NOTE: The Desc field in the WIP.DFD file is reserved for MS SQL and Oracle. Therefore, the DBA must change it to a different name, such as Description, in the DDL. Use these options to map the Desc field in the WIP.DFD file to the Description field in the WIP table:

```
< ODBC_FieldConvert >
    Desc = DESCRIPTION
< ORA_FieldConvert >
```

Desc = DESCRIPTION

5 Test the result.

To test	Follow these steps
PPS	Run PPS (AFEMAIN.EXE) to create and save a test transaction to WIP. Then complete it to delete the test transaction from the WIP List.

Documaker Server	To store WIP data in a database, run this command for multi-step execution:
	`gentnw32.exe, gendaw32.exe and genwpw32.exe`
	For single-step mode, use this command:
	`gendaw32.exe /jdt=wipjob.jdt`
	Then, to merge WIP data onto your form sets, run this command:
	`gendaw32.exe /jdt=afgjob.wip`

1877
PPS

GENERATING READABILITY STATISTICS

You can now generate readability statistics when working in PPS or the WIP Edit plug-in. In PPS and the WIP Edit plug-in, Flesch scoring considers multi-line text fields that contain text.

Flesch/Flesch–Kincaid Readability tests are designed to indicate how difficult a reading passage is to understand. Readability is based on factors such as the number of words in sentences and the number of letters or syllables per word.

Higher scores indicate material that is easier to read; lower numbers indicate harder-to-read passages. Here is the formula for the Flesch Reading Ease Score test:

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

where total syllables/total words = average number of syllables per word (ASW) and total words/total sentences = average sentence length (ASL).

NOTE: This test is designed for English. The scores may not be valid when you run the test on non-English text.

The system generates and displays readability statistics when you check grammar. You can choose the option to check grammar from the Tools menu when you are working in the Text Editor.

1895
PPS

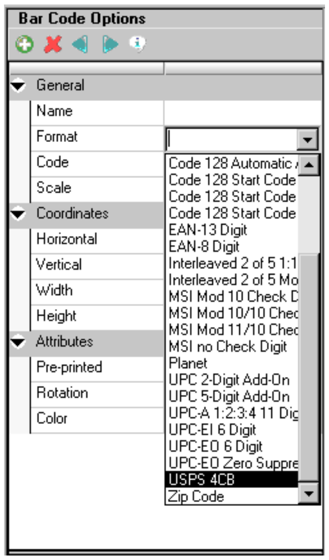
GENERATING THE USPS INTELLIGENT MAIL BARCODE (4-STATE CUSTOMER BAR CODE)

You can now generate the Intelligent Mail• bar code, formerly referred to as the *4-State Customer bar code* on your forms. The Intelligent Mail• bar code is a height-modulated bar code using four distinct vertical bar types. It encodes a 20, 25, 29, or 31-digit string into 65 vertical bars, each representing one of four possible states: full bar, ascender, tracker, and descender. Intelligent Mail bar code expands the ability to track individual mail pieces and provide its customers with greater visibility in the mail stream.

Type	Field	Digits
Tracking Code	Bar Code ID	2 (The 2nd digit must 0-4)
	Service Type ID	3
	Customer ID	6
	Sequence Number	9
Routing Code	Delivery Point ZIP Code	0, 5, 9, or 11
Total		31 maximum

Here is an example of the 4CB bar code:

In Studio or Image Editor, you would select the USPS 4CB option in the Format field. Here is an example from Studio:



1975
PPS

Inserting State Stamps and Signatures

You can now automatically insert state stamps and signature sections (images) via PPS.

For instance, there is an insurance industry requirement to apply a state stamp on certain forms. The stamp typically declares the document is valid in the state that has jurisdiction over the policy. Each state has its own stamp.

And, depending upon which agency issued the policy, different signatures may be required to make the policy valid within that insurance organization or state of issuance.

Configuring Your System

To handle the automatic insertion of a state stamp or a signature, create FAP files that contain the state stamp or signature or other information you want inserted and then add the following option in your INI file:

```
< AFEProcedures >
   AutoInsert = LSSW32->LSSAutoInserts
```

Note: You must make sure the dimensions of the FAP file that contain the information you want to insert fit appropriately within the target section.

On the target section — a DEC page, for instance — define a field that indicates where you want the system to place the inserted section. This field must have one of these *root* names:

```
LSS_LOGO
LSS_STAMP
```

```
LSS_SIGNATURE
```

NOTE: A *root* name means that the field name must start with these letters. The actual field name might be something like:

```
LSS_LOGO_LOGONAME
LSS_STAMP_TEXAS
LSS_SIGNATURE_JOHNDOE
```

Like page numbering fields, the system only requires that the field name starts with the required root name to determine it will be used. The remainder of the name is for your identification purposes.

How It Works

The system makes no assumptions on what the sections contain. When it locates a field with one of the pre-defined root names, it builds a corresponding FAP name from WIP index values defined for this document, as shown in this table:

This field	Builds a FAP name comprised of this information
LSS_STAMP	LSS_STAMP_{JURISDICTN}
LSS_LOGO	LSS_LOGO_{LOCID}_{%KEY1}_{%JURISDICTN}_{%SUBLOCID}
LSS_SIGNATURE	LSS_SIGNATURE_{KEY1}_{%LOCID}_{%JURISDICTN}_{%SUBLOCID}

In each of the definitions, the curly brace enclosures ({}) identify the WIP column that will substitute into the name. Column names preceded by a percent sign (%) indicate the column is optional.

If a WIP column is required and the data in that column is missing or spaces, the system removes the spaces. In this case, the character preceding the insertion point is examined and if found to be a space, period, hyphen, or underscore, it is also removed from the name. You cannot, therefore, assume that just because a column is required empty data will not be accepted.

For instance, suppose the definition is LSS_STAMP_{JURISDICTN}. The column JURISDICTN is not declared as optional, therefore data from the matching WIP column is inserted into the resulting name. If that column is blank, the resulting name generated will be LSS_STAMP because the preceding underscore character will be removed when the data is blank.

When there are optional columns, the system first tries the entire line, then removes one optional column at a time starting from the right-hand side. It will continue to try the name until all optional columns are removed. Here is an example:

```
LOCID      = ABC
SUBLOCID   = XYZ
JURISDICTN = GA
KEY1       = ANYSTATE
```

For LSS_STAMP, the only name the system would generate is:

```
LSS_STAMP_GA
```

If there was no FAP file with that name, nothing would be inserted.

For LSS_LOGO, the system would try the following names in sequence and select the first match:

```
LSS_LOGO_ABC_ANYSTATE_GA_XYZ
LSS_LOGO_ABC_ANYSTATE_GA
LSS_LOGO_ABC_ANYSTATE
```

If there was no FAP file with that name, nothing would be inserted.

For LSS_SIGNATURE, the system would try the following names in sequence and select the first match:

```
LSS_SIGNATURE_ANYSTATE_ABC_GA_XYZ
LSS_SIGNATURE_ANYSTATE_ABC_GA
LSS_SIGNATURE_ANYSTATE_ABC
```

If there was no FAP file with that name, nothing would be inserted.

NOTE: The system removes spaces, periods, underscores, or hyphens that precede an optional item.

You can override the way the system substitutes for the these fields using these INI options:

```
< LSS_INSERTS >
    LSS_STAMP = LSS_STAMP_{JURISDICTN}.FAP
    LSS_LOGO =
LSS_LOGO_{LOCID}_{%KEY1}_{%JURISDICTN}_{%SUBLOCID}.FAP
    LSS_SIGNATURE =
LSS_SIGNATURE_{KEY1}_{%LOCID}_{%JURISDICTN}_{%SUBLOCID}.FAP
```

You can include any valid WIP index column name within curly braces and you can include as many fields as necessary.

State Stamps

Skywire Software's Professional Services can create the state stamps you need. For more information, please contact your customer service representative.

2002
PPS

USING SUPERSCRIPT AND SUBSCRIPT IN TEXT LABELS, TEXT AREAS, AND VARIABLE FIELDS

You can now use superscript and subscript fonts in text labels, text areas, and variable fields. The system will automatically:

- Use a point size that is one-half of the current point size
- Position the superscript text above a mid-point line for the current font
- Position the subscript text below the mid-point line for the current font

NOTE: The superscript or subscript font you choose must be in your FXR file.

In Text Editor, select the text you want to superscript or subscript, then choose the appropriate option from the Edit menu. Text Editor tries to choose a smaller font from the FXR using the family name. If a smaller font is not available, the current font is used.

NOTE: The output from the RTF Print Driver may differ from the other drivers. RTF uses subscript and superscript commands to make the text easier to edit. Microsoft Word, however, automatically resizes the text in subscripts or superscripts. All other PPS print drivers maintain the font specified for the subscript or superscript text.

2067
RPS

PROMPTING FOR SEARCH CRITERIA

In PPS, the system now prompts you to enter at least one search criteria on the Retrieval window if you click Ok without entering any search criteria. This prevents you from inadvertently searching the entire archive.

2112
PPS

MAPPING ALTERNATE WIP INDEX COLUMNS FROM IMPORTED WIP DATA

You can add a new set of INI options to the INI control group associated with the import method you are using to identify globally-defined fields from which you want data transferred to the WIP index record.

Here is an example of how you would define one of these INI options:

```
WIPField = FAPField;DFDField
```

Option	Description
WIPField	FAPField is the name of a field stored in the global form set dictionary. The FAPField definition must name a field that is defined as Formset Global (globally) in the import file DFDField is the name of a WIP DFD field.

By default, the system looks for the Key1, Key2, and KeyID fields using their standard WIP names. After that, any additional WIPField definitions are located and mapped accordingly.

Here is an example:

```
< ImpFile_CD>
    WIPField = INSURED NAME;DESC
```

This definition looks for the globally-defined document field INSURED NAME and maps it into the WIP record DESC column. Since the DESC column is normally one of the WIP columns that would be automatically mapped in the V2 header, this definition overrides that and stores the value of the global field INSURED NAME in its place.

2113
PPS

Suppressing Duplicate Form Descriptions

Now you can suppress duplicate form descriptions. For instance, suppose you have several versions of what is basically the same form, but each instance of the form has a different name. For your purposes they are the same form and have the same description, even if the names are not identical. And when building the form table using the Form Description Lines feature, you would like to suppress the duplicate form descriptions. You can use the new ExcludeDuplicateDescriptions option to do this:

```
< FormDescTable >
   ExcludeDuplicateDescriptions = No
```

Option	Description
ExcludeDuplicateDescriptions	Enter Yes if you want the system to look for and exclude duplicate form descriptions. The default is No.

Note: By default, the Form Description Line feature will eliminate duplicate forms.

2136
PPS

Using the New WIP Fields

PPS and iPPS have been enhanced to capture and maintain new WIP fields. These new fields are typically used in the Carrier/MGA market or for Standard Lines Carriers with external agents. The fields include:

- LOCID
- SUBLOCID
- JURISDICTN
- TRNNAME

Typically, these fields represent the Agency Number, Agency Sub Number, Tax State, and Named Insured, respectively. These new fields work with the iPPS feature LocationProfileService and Stamps and Signatures feature (1975).

2144
PPS

Controlling Pagination when Editing a Form Set

You can use the new AutoPagination option to control pagination when you edit a form set in PPS or iDocumaker Workstation.

```
< EntryOptions >
    AutoPagination = No
```

Option	Description
AutoPagination	Enter No to turn off automatic re-pagination when editing a form set in PPS or iDocumaker Workstation. The default is Yes.

Note: Keep in mind you can also use the AutoPagination option in the Control control group to prevent the system from re-paginating the form set when an image grows because of expanding text in a text area. Here is an example:

```
< Control >
    AutoPagination = No
```

This tells the system not to re-paginate the form.

2155
PPS

Defaults for the Module and PrintFunc Options

Default values for the Module and PrintFunc options in the PrtType:xxx control group are now 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
```

2164
PPS

Using the New Complete and Exit Option

You can now add an option to your MEN.RES file, which defines the system menus, that lets you complete the current form set and then exit the system. This new option combines the Complete and Exit menu options.

NOTE: If there is an error on the form set, the system will not complete the form set and exit the system until you correct the error.

This option is typically used with iDocumaker Workstation or iPPS implementations but it can also be used if you have created specialized applications that launch entry via the RACLIB interface. PPS users can also use this option if users typically exit the application after completing a form set.

To add the Complete and Exit option to your system, you must first add the following line to the MEN.RES file:

```
MENUITEM "Complete and &Exit" 106 "AFEOS2->AFEComplShutDown"
"Complete the form set and exit the system"
```

You can add this line, or a line similar to it, under any menu group you like. You can also change the text of the option.

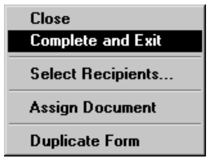
NOTE: In this example, *106* is used as the ID for the menu function. You can assign any ID between 100 and 200, as long as it is not used by another menu function.

The ampersand (&) indicates that the next character is the accelerator for this menu option. You can omit the ampersand if you like.

For instance, you could place this option on the Formset menu, as shown here:

```
POPUP "Form&set" 170 "Formset"
    BEGIN
        MENUITEM "&Close" 1069 "Close document" "NULL"
        MENUITEM "Complete and &Exit" 106 "AFEOS2->AFEComplShutDown"
"Complete the form set and exit the system"
        SEPARATOR
        MENUITEM "Select &Recipients..." 187 "AFEW32-
>AFESelectRecipients" "View forms by recipient"
        SEPARATOR
        MENUITEM "&Assign Document" 150 "AFEW32->AFEAssignDocument"
"Assign Document"
        SEPARATOR
        MENUITEM "&Duplicate Form" 104 "AFEW32->AFEDuplicateForm"
"Duplicate current form"
    END
```

To see this menu:

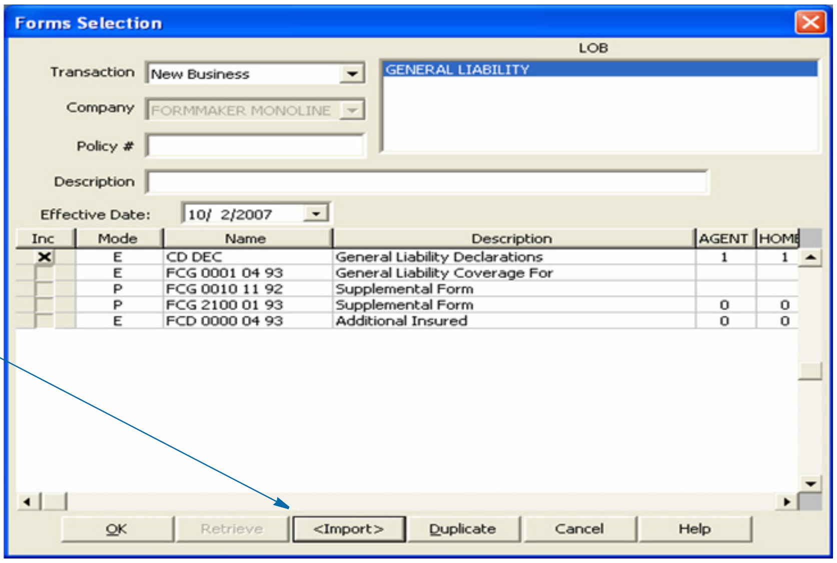


2178
PPS

Adding the Transaction Code During an Import

In PPS, when you import a transaction using the Import button on the Forms Selection window, the system now updates the Transaction field based on the transaction type specified in the import file.

Click here to import a new transaction.



Transaction	New Business
Company	FORMMAKER MONOLINE
Policy #	
Description	
Effective Date:	10/ 2/2007
LOB	GENERAL LIABILITY

Inc	Mode	Name	Description	AGENT	HOME
X	E	CD DEC	General Liability Declarations	1	1
	E	FCG 0001 04 93	General Liability Coverage For		
	P	FCG 0010 11 92	Supplemental Form		
	P	FCG 2100 01 93	Supplemental Form	0	0
	E	FCD 0000 04 93	Additional Insured	0	0

To turn on this capability, add the new SetTransCodeAfterImport option to your INI file, as shown here:

```
< FormSelection >
    SetTransCodeAfterImport = Yes
```

Note: All imports except XML are affected by this setting.

2180
PPS

Printing Multiple Copies from the GDI Print Driver

Now you can print multiple copies of a form set or page when using a GDI Print Driver in PPS. Previously, the GDI driver would only let you print one copy, regardless of the number you entered in the Number of Copies field on the Print window.

Keep in mind this only works when you are using the Windows printer as a normal (direct) output. It does not apply if you are routing one of the other Documaker printer language outputs through the GDI raw device.

2324
PPS

Specifying the Number of Copies

When selecting forms, any form with the Multicopy option turned on activates the Duplicate button. You can use this button to copy the form. Previously, if you wanted 10 copies, you had to click the Duplicate button 10 times.

In version 11.3, you can use the FormDuplicateCount option to tell the system to display a window that lets you enter the number of copies you want it to create.

To display this window, add the FormDuplicateCount option, as shown here:

```
< FormSelection >
   FormDuplicateCount = Yes
```

2346
PPS

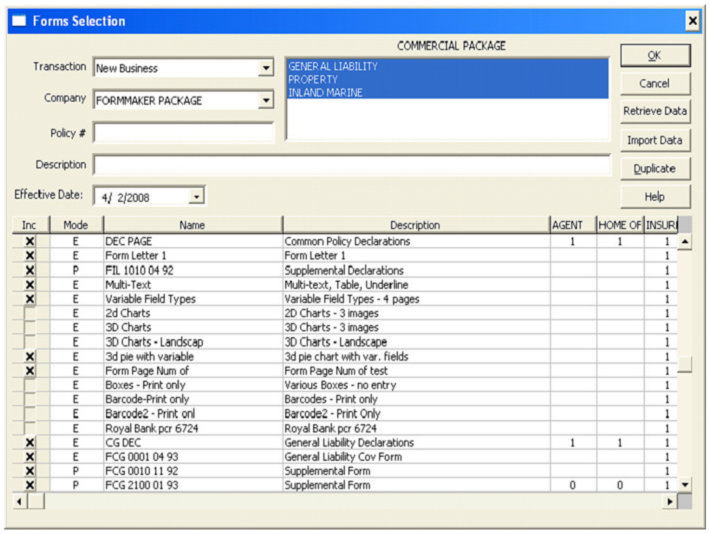
Resizing Grid Windows

In PPS, you can now resize grid windows. This lets you better customize the display of information, such as what you see on the Form Selection window, all WIP List windows, and the Archive Retrieve window.

To size the window, simply place your cursor on the edge of the window. The pointer will change to the sizing cursor. Then click and drag the window to the size you want. When you release the mouse, PPS remembers the size you have selected and displays the window using that size on subsequent visits.

NOTE: Each window has a minimum size.

Here is an example of a Form Selection window in which there is more data than can display at one time:



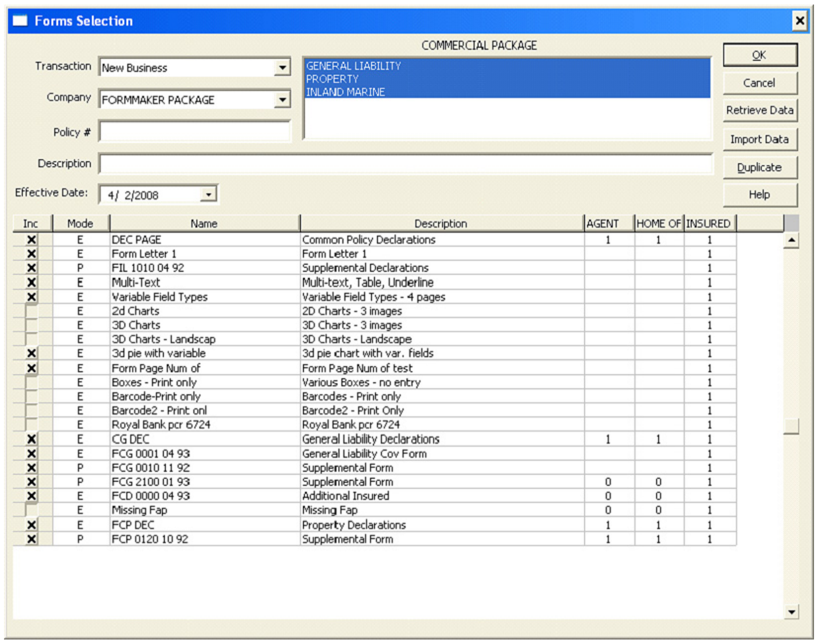
COMMERCIAL PACKAGE

Transaction: New Business
Company: FORMMAKER PACKAGE
Policy #:
Description:
Effective Date: 4/ 2/2008

GENERAL LIABILITY
PROPERTY
INLAND MARINE

Inc	Mode	Name	Description	AGENT	HOME OF	INSUR
X	E	DEC PAGE	Common Policy Declarations	1	1	1
X	E	Form Letter 1	Form Letter 1			1
X	P	FIL 1010 04 92	Supplemental Declarations			1
X	E	Multi-Text	Multi-text, Table, Underline			1
X	E	Variable Field Types	Variable Field Types - 4 pages			1
	E	2d Charts	2D Charts - 3 images			1
	E	3D Charts	3D Charts - 3 images			1
	E	3D Charts - Landscap	3D Charts - Landscape			1
X	E	3d pie with variable	3d pie chart with var. fields			1
X	E	Form Page Num of	Form Page Num of test			1
	E	Boxes - Print only	Various Boxes - no entry			1
	E	Barcode-Print only	Barcodes - Print only			1
	E	Barcode2 - Print onl	Barcode2 - Print Only			1
	E	Royal Bank pcr 6724	Royal Bank pcr 6724			1
X	E	CG DEC	General Liability Declarations	1	1	1
X	E	FCG 0001 04 93	General Liability Cov Form			1
X	P	FCG 0010 11 92	Supplemental Form			1
X	P	FCG 2100 01 93	Supplemental Form	0	0	1

You can now resize the window to see more of the rows and columns.



COMMERCIAL PACKAGE

GENERAL LIABILITY
PROPERTY
INLAND MARINE

Transaction: New Business
Company: FORMMAKER PACKAGE
Policy #:
Description:
Effective Date: 4/ 2/2008

Inc	Mode	Name	Description	AGENT	HOME OF	INSURED
X	E	DEC PAGE	Common Policy Declarations	1	1	1
X	E	Form Letter 1	Form Letter 1			1
X	P	FIL 1010 04 92	Supplemental Declarations			1
X	E	Multi-Text	Multi-text, Table, Underline			1
X	E	Variable Field Types	Variable Field Types - 4 pages			1
	E	2d Charts	2D Charts - 3 images			1
	E	3D Charts	3D Charts - 3 images			1
	E	3D Charts - Landscap	3D Charts - Landscape			1
X	E	3d pie with variable	3d pie chart with var. fields			1
X	E	Form Page Num of	Form Page Num of test			1
	E	Boxes - Print only	Various Boxes - no entry			1
	E	Barcode-Print only	Barcodes - Print only			1
	E	Barcode2 - Print onl	Barcode2 - Print Only			1
	E	Royal Bank pcr 6724	Royal Bank pcr 6724			1
X	E	CG DEC	General Liability Declarations	1	1	1
X	E	FCG 0001 04 93	General Liability Cov Form			1
X	P	FCG 0010 11 92	Supplemental Form			1
X	P	FCG 2100 01 93	Supplemental Form	0	0	1
X	E	FCD 0000 04 93	Additional Insured	0	0	1
	E	Missing Fap	Missing Fap	0	0	1
X	E	FCP DEC	Property Declarations	1	1	1
X	P	FCP 0120 10 92	Supplemental Form	1	1	1

1479
PPS

USING THE TOTALPAGES AND TOTALSHEETS FUNCTIONS

Use the TotalPages function to return the number of pages that will print for a given recipient or for all recipients. A page is considered any *side* of paper that has a printable image for a recipient. A duplex sheet with a front and back images counts as two pages.

Syntax

```
TotalPages(Recipient)
```

If you include the Recipient parameter, the count only reflects the pages that print for that recipient. If you omit the Recipient parameter, the count includes all recipients.

The count considers copy-counts and reflects the total number of printed sides that will be referenced. An image may be empty (containing no text or discernible print objects) and still be designated to print. So, the count does not necessarily mean the pages will contain any real text.

For example, assume you have a one-page document that has two recipients. Recipient1 gets one copy, while Recipient2 gets two copies.

With this command:

```
TotalPages("Recipient1")
```

The system returns one (1) as the page count

With this command;

```
TotalPages("Recipient2")
```

The system returns two (2) as the page count, since the one-page document will be printed twice. if you omit the Recipient parameter, the system returns three (3) as the page count.

NOTE: The count reflects when the function is called. The function cannot predict whether banner pages will be created or whether additional formatting or data entry will add or remove pages. Make sure you do not call this function until all page items have been created and formatted.

TotalSheets

Use this function to return the total number of sheets of paper that will print for a recipient. A sheet is considered a physical piece of paper that may have print on one or both sides. Therefore a duplex sheet with a front and back images will count as one sheet.

NOTE: Although the TotalSheets function does take duplex options into consideration, it has no knowledge of whether you will actually print to a printer that supports duplex commands. The count reflects what the document defines, not what the printer will support

Syntax

```
TotalSheets(Recipient)
```

If you include the Recipient parameter, the count only reflects the sheets that print for that recipient. If you omit the Recipient parameter, then the count reflects all recipients.

The count takes into consideration recipient copy counts and duplex options. An image may be empty (containing no text or discernible print objects) and still be designated to print. So, the count does not necessarily mean that the sheets will contain any real text.

For example, assume you have a two-page document that is duplexed (prints front and back). Recipient1 gets one copy, while Recipient2 gets two copies.

With this command:

```
TotalSheets(Recipient1)
```

The system returns one (1) as the sheet count

With this command;

```
TotalSheets(Recipient2)
```

The system returns two (2) as the sheet count, since the two-page document will be printed twice. if you omit the Recipient parameter, the system returns three (3) as the sheet count.

NOTE: The count reflects when the function is called. The function cannot predict whether banner pages will be created or whether additional formatting or data entry will add or remove pages. Make sure you do not call this function until all page items have been created and formatted.

1502 PPS

USING DAL FUNCTIONS FOR BIT MANIPULATION

This feature includes a set of new DAL functions to allow bit manipulation within integers. The new functions include:

- BitAnd on page 72
- BitOr on page 73
- BitXor on page 74
- BitNot on page 74
- BitShift on page 75
- BitRotate on page 76
- BitSet on page 77
- BitClear on page 77
- BitTest on page 78
- Hex2Dec on page 78
- Dec2Hex on page 79

BitAnd

Use this function to return the result of a bitwise AND operation performed on two numeric values.

Syntax

```
BitAnd(value1, value2)
```

The parameters specify the numeric values on which the bitwise AND operation is performed. If either parameter is not an integer, it will be converted to an integer before the bitwise AND operation is performed.

The bitwise AND operation compares each bit of value1 to the corresponding bit of value2. If both bits are 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to zero (0). Note that integer values have 32 bits to compare.

The following table shows the result of a bitwise AND operation:

Value1 bit	Value2 bit	Result bit
0	0	0
0	1	0
1	1	1
1	0	0

Example

Here is an example:

```
x = 3  (3 is 0011 in binary)
y = 6  (6 is 0110 in binary)

z = BitAnd(x,y)
z = 2  (2 is 0010 in binary)
```

BitOr

This function returns the result of a bitwise inclusive OR operation performed on two numeric values.

Syntax

```
BitOr(value1, value2)
```

Parameters specify the numeric values on which the bitwise OR operation is performed. If either parameter is not an integer, it will be converted to an integer before the bitwise OR operation is performed.

The bitwise inclusive OR operation compares each bit of value1 to the corresponding bit of value2. If either bit is 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to zero (0). Note that integer values have 32 bits to compare.

The following table shows the result of a bitwise OR operation:

Value1 bit	Value2 bit	Result bit
0	0	0
0	1	1
1	1	1
1	0	1

Example

Here is an example:

```
x = 3  (3 is 0011 in binary)
y = 6  (6 is 0110 in binary)

z = BitOr(x,y)
z = 7  (7 is 0111 in binary)
```

BitXor

This function returns the result of a bitwise exclusive OR operation performed on two numeric values.

Syntax

```
BitXor(value1, value2)
```

The parameters specify the numeric values on which the bitwise XOR operation is performed. If either parameter is not an integer, it will be converted to an integer before the bitwise XOR operation is performed.

The bitwise exclusive OR operation compares each bit of value1 to the corresponding bit of value2. If one bit is zero (0) and the other bit is 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to zero (0). Note that integer values have 32 bits to compare.

The following table shows the result of a bitwise XOR (exclusive OR) operation:

Value1 bit	Value2 bit	Result bit
0	0	0
0	1	1
1	1	0
1	0	1

Example

Here is an example:

```
x = 3  (3 is 0011 in binary)
y = 6  (6 is 0110 in binary)

z = BitXor(x,y)
z = 5  (5 is 0101 in binary)
```

BitNot

This function returns the result of a bitwise logical NOT operation performed on a numeric value.

Syntax

```
BitNot(value1)
```

The parameter specifies the numeric value on which the bitwise logical NOT operation is performed. If the parameter is not an integer, it will be converted to an integer before the bitwise logical NOT operation is performed.

The bitwise logical NOT operation reverses the sense of the bits in the value. For each value bit that is 1, the corresponding result bit will be set to zero (0). For each value bit that is zero (0), the corresponding result bit will be set to 1.

It is especially important to note that integer values have 32 bits to compare when examining the results of a NOT operation. All bits of the integer will be altered by this operation.

The following table shows the result of a bitwise logical NOT operation:

Value1 bit	Result bit
0	1
1	0

Example

Here is an example:

```
x = 3  (3 is 0000 0000 0000 0000 0000 0000 0000 0011 in binary)

z = BitNot(x)
z = -4 (-4 is 1111 1111 1111 1111 1111 1111 1111 1100 in binary)
```

Notice that the NOT operation affects all bits of the integer.

BitShift

This function returns the result of a bit logical shift operation performed on a numeric value.

Syntax

```
BitShift(value1, shiftAmt)
```

The first parameter specifies the numeric value on which the bitwise shift operation is performed. The second parameter specifies the number of bit positions to shift. If either parameter is not an integer, it will be converted to an integer before the bitwise shift operation is performed.

This is a logical shift, as opposed to a shift-and-rotate operation. This means bits shifted off the end of a value are considered lost.

NOTE: See the BitRotate on page 76 function for shift-and-rotate.

A positive shiftAmt value causes the bit pattern in value1 to be shifted left the number of bits specified by shiftAmt. Bits vacated by the shift operation are zero-filled.

A negative shiftAmt value causes the bit pattern in value1 to be shifted right the number of bits specified by shiftAmt. Bits vacated by the shift operation are zero-filled.

Note that integer values have 32 bits. Attempting to shift more than 31 bit positions will result in a zero (0) being returned, as all bits are cleared.

The following table shows the result of a bitwise SHIFT operation:

Value1 bits	Shift	Result value bits
6 (0110)	1	12 (1100)
6 (0110)	2	24 (0001 1000)
6 (0110)	3	48 (0011 0000)
6 (0110)	4	96 (0110 0000)
6 (0110)	-1	3 (0011)

Value1 bits	Shift	Result value bits
6 (0110)	-2	1 (0001)
6 (0110)	-3	0 (0000)
6 (0110)	-4	0 (0000)

Example

Here is an example:

```
z = BitShift(6,8)
z = 1536  (1536 is 0110 0000 0000 in binary)
```

BitRotate

This function returns the result of a bit shift-and-rotate operation performed on a numeric value.

Syntax

```
BitRotate(value1, shiftAmt)
```

The first parameter specifies the numeric value on which the bitwise shift-and-rotate operation is performed. The second parameter specifies the number of bit positions to shift. If either parameter is not an integer, it will be converted to an integer before the bitwise shift-and-rotate operation is performed.

This is a shift-and-rotate operation. This means that bits shifted off the end of a value are rotated back onto the value at the *other* end. In other words, the bits rotate in what might be thought of as a circular pattern — thus no bits are ever lost.

NOTE: See the BitShift on page 75 function for logical shift operations that do not shift-and-rotate.

A positive shiftAmt value causes the bit pattern in value1 to shift-and-rotate left the number of bits specified by shiftAmt. Bits that rotate off the left (high) end of the value return on the right (low) end.

A negative shiftAmt value causes the bit pattern in value1 to shift-and-rotate right the number of bits specified by shiftAmt. Bits that rotate off the right (low) end of the value return on the left (high) end. Note that integer values have 32 bits.

The following table shows the result of a bitwise shift-and-rotate operation:

Value1 bits	Shift	Result value bits
6 (0110)	1	12 (1100)
6 (0110)	2	24 (0001 1000)
6 (0110)	3	48 (0011 0000)
6 (0110)	4	96 (0110 0000)
6 (0110)	-1	3 (0011)

Value1 bits	Shift	Result value bits
6 (0110)	-2	-2147483647 (1000 0000 0000 0000 0000 0000 0000 0001)
6 (0110)	-3	-1073741824 (1100 0000 0000 0000 0000 0000 0000 0000)
6 (0110)	-4	1610612736 (0110 0000 0000 0000 0000 0000 0000 0000)

Example

Here is an example:

```
z = BitRotate(6,-8)
z = 100663296 (0000 0110 0000 0000 0000 0000 0000 0000)
```

BitSet

This function returns the result after setting the specified bit on in a value.

Syntax

```
BitSet(value1, bitpos)
```

The parameters specify the numeric value and the bit position on which the operation is performed. The specified bit is set to a 1 in the value provided. If the bit was already on, the value is unchanged. Specifying a negative or zero bit position does not result in any change to the value.

Note that integer values have 32 bits. When looking at the value in binary form, bit 1 is on the left and bit 32 is on the right.

```
Bit 32 -->0000 0000 0000 0000 0000 0000 0000 0000<-- Bit 1
```

Example

Here is an example:

```
y = 6  (6 is 0110 in binary)
z = BitSet(x,1)
z = 7  (7 is 0111 in binary)

y = 6  (6 is 0110 in binary)
z = BitSet(x,4)
z = 15 (15 is 1110 in binary)
```

BitClear

This function returns the result after clearing the specified bit in a value.

Syntax

```
BitClear(value1, bitpos)
```

The parameters specify the numeric value and the bit position on which the operation is performed. The specified bit is set to a zero (0) in the value provided. If the bit was not on, the value is unchanged. Specifying a negative or zero bit position does not result in any change to the value.

Note that integer values have 32 bits. When looking at the value in binary form, bit 1 is on the left and bit 32 is on the right.

```
Bit 32 -->0000 0000 0000 0000 0000 0000 0000 0000<-- Bit 1
```

Example

Here is an example:

```
y = 6  (6 is 0110 in binary)
```

```
z = BitClear(x,1)
z = 6  (6 is 0110 in binary) (bit 1 was already zero)

y = 6  (6 is 0110 in binary)
z = BitClear(x,2)
z = 4  (4 is 0100 in binary)
```

BitTest

This function returns TRUE (1) if the specified bit in a value is a 1; otherwise FALSE (0) is returned.

Syntax

```
BitTest(value1, bitpos)
```

Parameters specify the numeric value and the bit position on which the operation is performed. The specified bit is tested for a 1 value. If the bit is a 1, then 1 is returned. If the bit is zero (0), then zero (0) is returned. Specifying a negative or zero bit position will result in zero (0) being returned.

Note that integer values have 32 bits. When looking at the value in binary form, bit 1 is on the left and bit 32 is on the right.

```
Bit 32 -->0000 0000 0000 0000 0000 0000 0000 0000<-- Bit 1
```

Example

Here is an example:

```
y = 6  (6 is 0110 in binary)
z = BitTest(x,1)
z = 0  (bit 1 was not on)

y = 6  (6 is 0110 in binary)
z = BitTest(x,2)
z = 1  (bit 2 was on)
```

Hex2Dec

This function returns the integer equivalent of a hexadecimal string.

Syntax

```
Hex2Dec(value1)
```

The parameter specifies a string of characters that are to be converted into an integer value.

If the string value does not represent a valid hexadecimal number, the results are questionable and can result in only part of the value being converted.

The largest hexadecimal value supported is FFFFFFFF. Keep in mind, however, that hexadecimal values are considered *unsigned* while integer values can be both positive and negative.

The largest integer value 2,147,483,647 is 7FFFFFFF when represented using hexadecimal. HEX values greater than 80000000 represent negative integer values. Hex value FFFFFFFF represents the integer value -1.

Example

Here is an example:

```
y = "1A2B"
z = Hex2Dec(y)
```

```
Result is z = 6699

y = "FF00"
z = Hex2Dec(y)
Result is z = 65280
```

Dec2Hex

This function returns the hexadecimal equivalent of an integer value.

Syntax

```
Dec2Hex(value1, digits)
```

The value1 parameter specifies a integer value to be converted into a hexadecimal string value. If the parameter is not specified as an integer, it will be converted to an integer before performing the operation.

The largest hexadecimal value supported is FFFFFFFF. Keep in mind, however, that hexadecimal values are considered *unsigned* while integer values can be both positive and negative.

The largest integer value 2,147,483,647 is 7FFFFFFF when represented using hexadecimal. HEX values greater than 80000000 represent negative integer values. Hex value FFFFFFFF represents the integer value -1.

The second parameter (digits) defaults to zero (0) and means the resulting hexadecimal value will not have leading zeros. You can set this parameter from 1 to 8 to control the minimum number of hexadecimal digits returned in the string. If you set the minimum too small to represent the value, it will be ignored.

Example

Here is an example:

```
y = 1000
z = Dec2Hex(y)
Result is z = 3E8

y = 254220
z = Dec2Hex(y,8)
Result is z = 0003E10C

y = -2
z = Dec2Hex(y)
Result is z = FFFFFFFE
```

1481
PPS

Using the PageImage Function

Use this function to return the name of an image on a given page number within the form set or form. If you include the name of a recipient as a parameter, the system will filter the images by that name. Once you have an image name, you can use other DAL functions to query the image, to insert a new image, or to delete the image.

Syntax

```
PageImage(Page, Recipient, Form, Group)
```

Parameter	Description
Page	Include this parameter to indicate the specific page where you want to locate an image. If you omit this parameter, an image from page one is located. Depending upon the remaining parameters, this page will be the page within the entire form set, or within a given form.
Recipient	Include this parameter to filter the images located by that recipient. If you omit this parameter, the name of the first image on the requested page is returned.
Form	Include this parameter if you want the system to first locate the specified form and then use the Page parameter to find the specified page within that form. If you omit this parameter, the Page parameter is based on a page located by starting at the first page of the form set or group (if the Group parameter is specified).
Group	Include this parameter to tell the system to first locate a specific group. If you also include the Form parameter, the system will find that form in that group. If you omit the group but include the form, the system looks for that form in the current group — which is identified by the current field or image executing the script. If you include the group but omit the form, the system uses the Page parameter to return that page in the specified group.

The name returned by this function also includes the *occurrence* value if the image occurs more than once. For instance, if an image named, MYIMAGE, is located on the given page, but this is the second occurrence of the image within the named form, the name returned will be *MYIMAGE\2*.

1507
PPS

ENHANCED AVAILABILITY OF DAL FUNCTIONS

These DAL functions, formerly specific to Documaker Server, are now available for system-wide use:

- Print_It
- MajorVersion
- MinorVersion

These DAL functions, formerly specific to Documaker Server, are now also available for PPS or any application that prints and supports DAL:

- PrinterClass
- PrinterGroup
- RecipBatch

In addition, the RecipName function, which was supported by Entry, is now supported by any application that prints and uses DAL.

For more information on these functions, see the DAL Reference.

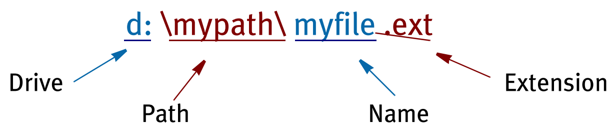
1515
PPS

USING DAL TO MANIPULATE FILE NAMES

Since you can use DAL functions to read tables and to set device names for output print stream files, this feature further extends DAL functionality by letting you manipulate file names.

NOTE: See Feature 1510 for more information on setting device names for output print stream files.

For instance, this feature lets you get the components of a file name (drive, path, name, and extension) and combine those into a full file name. For example, for computers running Windows file names look like this:



This feature includes these new DAL functions:

- FileDrive
- FilePath
- FileName
- FileExt
- FullFileName

All platforms are supported and both Documaker Server and PPS are supported. Here are descriptions of these new functions:

FileDrive

Use this function to get the drive component of a file name.

Syntax

```
FileDrive("FullFileName")
```

This function accepts a string containing a fully qualified file name, returns a string that contains the drive component of that file name.

Here is an example:

```
MYDRIVE = FileDrive("d:\mypath\myfile.ext")
```

In this example, MYDRIVE would contain:

"d:"

FilePath

Use this function to get the path component of a file name.

Syntax

```
FilePath("FullFileName")
```

This function accepts a string containing a fully qualified file name, returns a string that contains the path component of that file name.

Here is an example:

```
MYPATH = FilePath("d:\mypath\myfile.ext")
```

In this example, MYPATH would contain:

"\mypath\"

FileName

Use this function to get the name component of a file name.

Syntax

```
FileName("FullFileName")
```

This function accepts a string containing a fully qualified file name, returns a string that contains the name component of that file name.

Here is an example:

```
MYNAME = FileName("d:\mypath\myfile.ext")
```

In this example, MYNAME would contain:

"myfile"

FileExt

Use this function to get the extension component of a file name.

Syntax

```
FileExt("FullFileName")
```

This function accepts a string containing a fully qualified file name, returns a string that contains the extension component of that file name.

Here is an example:

```
MYEXT = FileExt("d:\mypath\myfile.ext")
```

In this example MYEXT would contain:

".ext"

FullFileName

Use this function to make the full file name.

Syntax

```
FullFileName("Drive","Path","Name","Ext")
```

This function accepts a string containing the drive, path, name, and extension components of a fully qualified file name, assembles them, and returns a string that contains the full file name.

Here is an example:

```
MYFILENAME = FullFileName("d:","\mypath\","myfile",".ext")
```

In this example, MYFILENAME would contain:

"d:\mypath\myfile.ext"

NOTE: If, in this example, *\mypath* had no trailing slash, the FullFileName function would have added it for you.

1524
PPS

USING THE PAGEINFO FUNCTION

Use this function to get information about the page of a form you specify. This information includes height, width, and orientation.

Syntax

```
PageInfo(Prefix, Page, Recipient, Image, Form, Group)
```

Parameter	Description:
Prefix	This parameter identifies a prefix for creating the variable names to contain the page information.
Page	(optional) This parameter determines the relative page number that should be examined, once the starting page is located by examining the remaining parameters. The default is the first page located.
Recipient	(optional) This parameter names a specific recipient that must be used on an image of the page located. If you omit this parameter, the function matches the first page identified by the remaining search criteria.
Image	(optional) This parameter names an image that should be found to identify the page.
Form	(optional) This parameter names a form that contains the page to be found.
Group	(optional) This parameter names a group that must contain the page to be found.

The Image, Form, and Group parameters are optional and when used will work together to locate the starting page for the search. Here are some examples:

- If you name an image, but no form or group, the assumption is the image is on the current form.
- If you name a form, without a group, the assumption is the form must be within the current group of forms.
- If you name an image and a group, but no form, the assumption is the image can occur on any form within that group.
- If you omit the image, form, and group parameters the search starts from the beginning of the document set.

Once the requested page is located, the system assigns the page information to DAL variables using the Prefix parameter. If these variables do not exist in DAL, the system creates them for you. The system creates four internal variables: *prefix*.height, *prefix*.width, *prefix*.landscape, and *prefix*.paper. If these variables exist, the system modifies them with the new information.

For example, a call like this will create four variables.

```
PageInfo("MYPAGE");
```

Variable	Description
MYPAGE.Height	Contains the height of the page in FAP units (2400 DPI).
MYPAGE.Width	Contains the width of the page in FAP units (2400 DPI).
MYPAGE.Landscape	Contains one (1) if the page is landscape, otherwise zero (0).
MYPAGE.Paper	Contains a value that corresponds to a paper size table entry.

NOTE: See Feature 0190 to see a table that shows all of the supported paper sizes.

Note that for landscape pages, the height and width values reflect the rotation of the width and height. For instance, non-landscape letter documents return a height of 26400 and a width of 20400. Landscape letter documents return a height of 20400 and a width of 26400.

The page size (height and width) is determined by finding the first image on a page with the required recipient. If no recipient is specified, the first image on the page is used. The form pages within a document do not have to be the same size. Also note that if the first image on a page is a custom size, the width and height will reflect the *best* values.

Generally when an image is a custom size, the actual page size is found in the form definition. If, however, the form size (height or width) is smaller than the corresponding image size, then the larger of the values is returned.

Also remember since page size is determined by the first image designated for a given recipient, it is possible for the *same* page to have a different size for different recipients.

The PageInfo function returns a value if used in an expression that requires it. The possible return values are zero (0) if the requested page could not be found, or non-zero if the page is found.

Possible reasons for a page not to be located include:

- The page number is outside the range of pages for the given search criteria. For instance, you ask for page three of a form that only has two pages.
- The recipient cannot be located within the document search criteria.
- The image, form, or group (or combination thereof) cannot be located within the specified document.

1533
PPS

Using the NL Function

Use this function to retrieve a string that contains a new line character sequence. This is useful when you are creating output text messages that contain line breaks.

NOTE: On Windows, this function returns a carriage return/line feed pair. On UNIX, it returns a line feed. The function works in both Documaker Server and PPS.

Syntax **NL()**

There are no parameters for this function.

Example This example shows how you can use this function with the Print_It function:

```
Print_It("This is line one." & NL() & "This is line two.")
```

In this example, two lines are output to the command line during Documaker Server processing. Without this function, you would have to include two Print_It statements.

```
This is line one.
This is line two.
```

This example shows how you can create multi-line text area messages:

```
data = ?("cus_name") & NL() & ?("state") & ",  " & ?("zip")
SetFld(data, "cus_ss")
```

In this example, two lines are stored in a multi-line text area on separate lines. Without this function, you would have to define the multi-line text area, a fixed-size font, and the script would have calculated the number of spaces to pad to the first line to make sure the line wrapped properly.

```
John A. Smith
CA,  81234-4444
```

You can also use the NL function when you are creating comment strings you want inserted into a print stream using the AddComment procedure.

1545
PPS

Using the Exists and GetValue DAL Functions

These new functions have been added:

- Exists
- GetValue

Exists

Use this function to determine if a DAL symbolic variable exists. This can be useful because referencing a variable that does not exist will cause a runtime syntax error. You can use this function to verify that DAL variables which are created external to your script have been created before you try to reference them.

Syntax

```
Exists(Symbol)
```

Parameter	Description	Defaults to
Symbol	Accepts a string that specifies the name of a DAL symbolic variable. This can be from an expression or from another string variable.	None, but you must make an entry.

This function returns True (1) if the variable exists. It returns False (0) if there is no such symbol defined.

Example

Here is an example. Assume the string variables 'tbl_1', 'tbl_2', 'tbl_3', and 'tbl_4' respectively contain: '*Ford*', '*Chev*', '*Olds*', and '*VW*'.

```
If Exists("tbl_" & #line) Then
        Return ( GetValue("tbl_" & #line) )
    Else
        Return (" ")
End
```

In this example, if *#line* is set to 3, the string '*Olds*' is returned. If *#line* is set to 5, a 'blank' is returned.

See also

GetValue

GetValue

Use this function to return a string that contains the contents of the DAL symbolic variable specified by the parameter. You can use this function when the name of the DAL variable is also stored in a variable, such as when a variable has to be addressed in another external script.

Syntax

```
GetValue(Symbol)
```

Parameter	Description	Defaults to
Symbol	Accepts a string that specifies the name of a DAL symbolic variable. This can be from an expression or from another string variable.	None, but you must make an entry.

NOTE: You will get a syntax error if you omit the Symbol parameter or if the DAL symbolic variable does not exist. It is wise to use this function with the Exists function.

Example

Here are some examples. Assume the:

- String variable 'my_variable' contains: *"Hello World"*
- Numeric variable '#_veh' contains: *20*
- String variables 'tbl_1', 'tbl_2', 'tbl_3', and 'tbl_4' respectively contain: '*Ford*', '*Chev*', '*Olds*', and '*VW*'.

In this example, the variable named *contents* is set to the string "*Hello World*":

```
variable_name = "my_variable"
contents = GetValue(variable_name)
```

This example stores the value, *20*, in the field entitled '*total # of vehicles*' in the current image:

```
SetFld ( GetValue("#_veh"), "total # of vehicles")
```

In this example, if *#line* is set to 3, the string '*Olds*' is returned. If *#line* is set to 5, a 'blank' is returned.

```
If Exists("tbl_" & #line) Then
            Return ( GetVaule("tbl_" & #line) )
    Else
            Return (" ")
End
```

1548 PPS

Miscellaneous New DAL Functions

This feature adds these new DAL functions:

- BankRound on page 87
- CompressFlds on page 88
- ConnectFlds on page 90
- SpanField on page 93
- RootName on page 94
- XMLFirstText on page 95
- XMLNextText on page 95

BankRound

Use this function to round numbers based on Banker's rounding. With Banker's rounding, values below 0.5 go down and values above 0.5 go up. Values of exactly 0.5 go to the nearest even number. In contrast, the Round function always rounds 0.5 upwards.

Note: When you add values which have been rounded using the standard method of always rounding .5 in the same direction, the result includes a bias that grows as you include more rounded numbers. Banker's rounding is designed to minimize this.

Syntax

```
BankRound(Value)
```

Parameter	Description
Value	Enter the value you want the system to round.

Example

Here are some examples that compare BankRound with Round:

With BankRound		Whereas, with Round	
This	Returns	This	Returns
BankRound(123.425)	123.42	Round(123.425)	123.43
BanKRound(123.435)	123.44	Round(123.435)	123.44

CompressFlds

Use this function/procedure to move field data to compress blank space. This function moves field data from one field to a prior named field to compress the space between the fields. Typically you use this function to compress vertical space, as in address lines, but the fields do not have to be vertical relative to each other. You can compress any field.

NOTE: The data moves between the fields; the actual location of each physical field remains the same.

CompressFlds can be used as a procedure or as a function.

Syntax

```
CompressFlds(FieldList, Image, Form, Group)
```

Parameter	Description	Defaults to...
FieldList	A list of the fields you want to compress, separated by commas. Here is an example: `"FIELD1, FIELD2, FIELD3"`	no default
Image	(optional) Name of an image that contains the fields you named.	the current image
Form	(optional) Name of a form that contains the image and/or field you named.	the current form
Group	(optional) Name of the form group that contains the form, image, or fields.	the current group

NOTE: When using this function in batch processing, make sure the fields exist on the image. Some implementations that use versions of the system prior to version 11.0 do not load FAP files in all cases, and fields will not be created when data mapping did not place any data into the field.

Keep in mind...

- Each subsequent field with data is mapped into the first available empty field which you included in the list.

- Fields are defined in FAP images with a tabbing order. This tabbing order typically matches the order in which field level rules are processed during batch processing. Unlike the SetAddr rules, the CompressFlds function can compress fields in any order, and the field spaces do not have to be *compressed up* following the tabbing order.
- The last movement of that field determines the final location of a given field's data.
- Always specify a set of unique field names. Do not attempt to name a field more than once within a field list as this can produce unpredictable results.
- This function does not work with barcode or multi-line text fields.

Example

For this example, assume the following fields and data:

This field	Contains
FIELD_A	*ABCDEFG*
FIELD_B	is empty
FIELD_C	is empty
FIELD_D	*TUVWXYZ*

Assume your field list looks like this:

```
"FIELD_A, FIELD_B, FIELD_C, FIELD_D"
```

FIELD_A does not move because there is no field named before it.

FIELD_B and FIELD_C are empty; therefore, the data from FIELD_D moves into FIELD_B, which is the first available empty field.

The result looks like this:

This field	Contains
FIELD_A	*ABCDEFG*
FIELD_B	*TUVWXYZ*
FIELD_C	is empty
FIELD_D	is empty

If you had specified the field list parameter had been specified like this:

```
"FIELD_D, FIELD_C, FIELD_B, FIELD_A"
```

The result would be as follows:

This field	Contains
FIELD_A	is empty
FIELD_B	is empty
FIELD_C	*ABCDEFG*
FIELD_D	*TUVWXYZ*

ConnectFlds

Use this function/procedure to move fields (change field coordinates) in such a way as to make the field's text appear to be concatenated. This function does not literally concatenate the fields but instead repositions and aligns field text along a common horizontal coordinate so the field's data appears concatenated. It does not move fields vertically.

This function automatically loads the image — either the FAP file or the compiled version of the FAP file —if the image has not already been loaded. FAP files must be loaded to provide some of the information required to perform the operation.

Syntax

```
ConnectFlds(FieldList, Image, Form, Group)
```

Parameter	Description	Defaults to...
FieldList	A list of the fields you want to connect, preceded by a movement flag and separated with commas. Here is an example: `"FIELD1, FIELD2, FIELD3"` If a field name is not preceded by a movement flag or if it is preceded by the *F* movement flag, which indicates it is a fixed field, the field is not moved. The first field you name in the parameter must be a fixed field. The rest of the field names in your list indicate fields you want moved adjacent to the fixed field. Each field you name is moved according to the use described by the movement flag that precedes its name.	no default
Image	(optional) Name of an image that contains the fields you named.	the current image
Form	(optional) Name of a form that contains the image and/or field you named.	the current form
Group	(optional) Name of the form group that contains the form, image, or fields.	the current group

In the FieldList parameter you must specify a fixed field and at least one field to move (visually concatenate) to the left or right side of the fixed field. You can specify multiple fields to move.

NOTE: This function does not move fields vertically. Fields are only moved horizontally. FAP image composers should adjust the vertical alignment between fields at composition time.

By default, each concatenation will be placed the distance of one space character from the fixed field, unless the parameter indicates otherwise. You can include these movement flags in the FieldLIst parameter:

Flag	Description
L	Tells the system to move the specified field so it appears to be appended to the left of the fixed field.
R	Tells the system to append the specified field to the right of the fixed field.
NO	Tells the system you want no spacing between the two fields.

Here is an example:

```
"F=FIELD1,RNO=FIELD2"
```

Here, the contents of FIELD2 are placed immediately adjacent to the end of the contents of FIELD1 without an intervening space.

Keep in mind...

- As each field is appended to the fixed field, the fixed rectangle grows. By growing the fixed rectangle, additional fields that append move based upon where the prior appended field ended.
- If a field specified for appending does not contain any data or is not valid, then no space, or space holder, is included in the concatenation.
- If a field contains centered or right justified data padded with spaces then the results may appear to be incorrect. This function calculates the width of a field based upon the entire contents and will not remove spaces, or any other white space characters, in the fields.
- Naming a field to move more than once in the first parameter can cause unpredictable results.
- The last movement of a field will determine the final location of a field's data.
- During any movement operation, the field being moved cannot also be named as the fixed field.
- This function does not work with barcode or multi-line text fields.
- This function does not handle rotated fields.

Example

For the following examples, make these assumptions:

This field	Contains
FIELD1	ABC
FIELD2	DEF
FIELD3	XYZ

If you enter:

```
ConnectFlds("F=FIELD1,R=FIELD2")
```

You get this result:

```
ABC DEF
```

If you enter:

```
ConnectFlds("F=FIELD1,L=FIELD2,R=FIELD3")
```

You get this result

```
DEF ABC XYZ
```

This example appended FIELD2 to the left side of FIELD1 and appended FIELD3 to the right side of FIELD1. The fixed field, FIELD1, did not move. FIELD2 and FIELD3 moved to align with FIELD1. During this operation, FIELD1 never moved.

If you enter:

```
ConnectFlds("FIELD1,LNO=FIELD2,RNO=FIELD3")
```

You get this result:

```
DEFABCXYZ
```

This example is similar to the prior example but uses the NO parameter.

If you enter:

```
ConnectFlds("F=FIELD1,R=FIELD2,R=FIELD3")
```

You get this result:

```
ABC DEF XYZ
```

In this example, two fields are appended to the right of the fixed field. The first appended field expanded the rectangle, which allows the next one to append after the last.

If you enter:

```
ConnectFlds("F=FIELD1,R=FIELD2,F=FIELD2,R=FIELD3")
```

You get this result:

```
ABC DEF XYZ
```

Notice that the result of this example is the same as the previous example. In this case, the fixed field was changed to FIELD2 after FIELD2 had moved adjacent to FIELD1. Then FIELD3 was moved adjacent to FIELD2 in its new location.

If you enter:

```
ConnectFlds("F=FIELD1,R=FIELD2,R=FIELD2")
```

You get this result:

```
ABC       DEF
```

In this case, FIELD2 is defined to move twice. Since the operations are sequential, the field first moved adjacent to FIELD1. This movement expanded the fixed rectangle used by subsequent movements. When the field was named again, it moved relative to the newly expanded rectangle, resulting in the field appearing farther to the right, a distance equal to the size of the text in the field plus the width of two spaces.

SpanField

Use this function/procedure to move a field horizontally and then resize it to span the distance between two other fields you specify. This function sets the span field's contents to be enough of a fill character to span the distance.

This function only moves the field horizontally. It will not move the other two fields. The image designer must ensure vertical alignment between the fields.

NOTE: If you use this function with resources created prior to version 11.0, which had separate FAP and DDT files, this procedure automatically loads the image (FAP or compiled FAP) if it is not already loaded.

Syntax

```
SpanField(SpanField, LeftField, RightField, Image, Form, Group)
```

Parameter	Description	Defaults to...
SpanField,	Specifies the filler character to use to span the distance between the end of the left field text and the beginning of the right field text. If either field is empty, the left coordinate of the field is used. The system only uses the first character of the text contained in the field you specify as the filler character. In addition to the filler character, the field you specify also determines the font ID to be used for calculating the number of characters required to fill the width of the field. If there is fractional space remaining in the width, the filler character is duplicated. The extra white space will be placed to the left of the span field, so that the spanned field will is placed against the right-most field.	a period (.)
LeftField	Enter the name of the field on the left of the area you want to span.	no default
RightField	Enter the name of the field to the right of the area you want to span.	no default

Parameter	Description	Defaults to...
Image	(optional) Name of an image that contains the fields you named.	the current image
Form	(optional) Name of a form that contains the image and/or field you named.	the current form
Group	(optional) Name of the form group that contains the form, image, or fields.	the current group

The SpanField parameter is always the first parameter, but you can specify the LeftField and RightField parameters in any order. The function automatically determines which of the two fields is to the right or left of the span field.

NOTE: If you are using the SpanField function in batch processing, the JustFld rule may be useful to right justify the right-most field to make sure the maximum distance is spanned. If you use the Move_It rule, or other rules that support right justification by padding the data with spaces, the results will be incorrect. The SpanField function calculates the width of a field based upon the entire contents and does not remove space, or any other white space or characters in the fields.

Example

Here is an example:

Assume LeftField contains ABCDEFG, RightField contains $123.45, and SpanField contains a dash (-).

```
SpanField("SPANFIELD", "LEFTFIELD", "RIGHTFIELD")
```

Yields: ABCDEFG-----------$123.45

The horizontal location of the span field is adjusted to make sure it is positioned against the right edge of the left field, and then expanded with enough of the fill character to fill the gap between the left and right fields. The image designer is responsible for vertical alignment.

RootName

Use the RootName function to extract and return the root name, or the original part of the name, of a string you specify. This function strips off the #nnn portion of a field name to get the root field name.

Understanding the System

Documaker requires that all fields on an image be uniquely named. Image Editor forces a unique name if a field is duplicated. Appending *#002* or *#003*, for example, to the end of the field name creates unique names. In some cases you may want to use the name of a field to supply the name of a data dictionary symbol to use to fill that field. If each unique instance of a field is to use the same name, this can present a problem.

Syntax

```
RootName (FieldName)
```

Parameter	Description
FieldName	Enter the name of the field for which you want the system to return the root portion of that name.

Example

Here are some examples:

```
RootName("Street address #002")
```

Returns *Street address*.

```
MYFIELDNAME = "Comment #003"
RootName(MYFIELDNAME)
```

Returns *Comment*.

```
RootName(FieldName())
```

Returns the root name of the current field.

XMLFirstText

Use this function to set the current text to be the first text element in the XML search list and then retrieve that text.

Syntax

```
XMLFirstText(List)
```

Parameter	Description
List	Enter the name of the list.

Example

Here is an example:

```
Mystring = XMLFirstText(%mylist)
```

XMLNextText

Use this function to retrieve the next text element in the XML search list.

Syntax

```
XMLNextText(List)
```

Parameter	Description
List	Enter the name of the list.

Example

Here is an example:

```
Mystring = XMLNextText(%textlistH);
```

1549
PPS

USING THE MOD FUNCTION

Use this function to return the remainder from modular arithmetic.

Syntax

```
MOD(Numerator, Denominator)
```

Parameter	Description
Numerator	Enter the value you want used as the numerator.
Denominator	Enter the value you want used as the denominator.

This function returns the integer remainder from an integer division.

NOTE: If you enter zero (0) as either the numerator or denominator, the system returns zero. Decimal or string input parameters are converted to integer values prior to the calculation.

Example

Assume you have the following entry in the SETRCPTBL.DAT file for the form trigger being processed. Also assume there are 30 records in the extract file that match the search mask.

```
;RP10;CIS;qa_f1550;;;Customer(1);;1,M;25;0;1;;DALTrigger;FEATURE155
0;
```

Here is an example:

```
BeginSub Feature1550
    #rec = CountRec("1,Feature1550,31,Data")
    #remaining = MOD(#rec, TriggerRecsPerOvFlw( ))
    While(#remaining > 0)
*       write additional records
        Write_fm( )
        #mod -= 1
    Wend
    Return(#rec)
EndSub
```

In this example, the MOD function returns the integer remainder of 5. If no extract records matched the search mask, the system would have returned zero (0).

1554
PPS

USING THE NEW RESETFLD FUNCTION PARAMETERS

You can use the new Image, Form, and Group parameters when specifying a field to reset. The ResetFld function lets you delete the data from a variable field, including multi-line variable fields. This function works even if no data was entered into the field.

Syntax

```
ResetFld (Field, Image, Form, Group)
```

Parameter	Description	Defaults to...
Field	Enter the name of the field you want to reset. Enclose the field name in quotation marks. Here is an example: `"FIELD01"`	No default, entry required
Image	Enter the name of the image that contains the field name.	The current image
Form	Enter the name of a form that contains the image or field name or both.	The current form
Group	Enter the name of the form group that contains the form, image, and/or field name.	The current group

Example

Here are some examples:

Function	Result	Explanation
ResetFld (“ACCUM_TOT”)	1 or 0	Clears the data from the ACCUM_TOT field.
ResetFld("TOTAL_PREM", "BOAT PREM")	1 or 0	Clears the data in the field, TOTAL_PREM, in the image, BOAT PREM.

1563 PPS

USING THE SETREQUIREDFLD FUNCTION

Use this function to change the required option of a field to Required or Not Required.

Syntax

```
SetRequiredFld (Required, Field, Image, Form, Group)
```

Parameter	Description
Required	Enter Yes if you want to make the field required. Enter No if you want to make the field not required.
Field	(Optional) Enter the name of the field. If you omit this parameter, they system uses the current field.
Image	(Optional) Enter the name of the image. If you omit this parameter, they system uses the current image.
Form	(Optional) Enter the name of the form. If you omit this parameter, they system uses the current form.
Group	(Optional) Enter the name of the group. If you omit this parameter, they system uses the current group.

Example

Here are some examples:

```
SetRequiredFld ("Yes", "Myfield", :MyImage", "Myform", "MyGroup");
SetRequiredFld ("Yes", "Myfield", :MyImage", "Myform", );
```

```
    SetRequiredFld ("Yes", "Myfield", :MyImage", );
    SetRequiredFld ("Yes", "Myfield",);
    SetRequiredFld ("Yes", );
```

If you include the Image parameter, but omit the field parameter, the system uses the first field on that image. If you omit the Image and Field parameters, but include the Form, the system looks for the first field on the first image of the form you specified, and so on.

1589
PPS

Using the ListInList Function

This function searches for the comma-delimited list specified by the second parameter for each character string in the comma-delimited list specified by the first parameter. If a match is found, the function returns the ordinal position (integer) of the first string in the second parameter that matches any of the strings in the first parameter. If no match is found, the function returns a zero (0).

Syntax

```
    ListInList (String_list, List_string)
```

Parameter	Description	Defaults to
String_list	Enter the name of the list of character strings or enter the list of character strings you want to search for. Use commas to separate each character string entry you want to find. Keep in mind the system considers spaces when searching, so strings must match exactly.	No default
List_string	Enter the name of the string list or the character string list to be searched. Use commas to separate each string entry you want to search for.	No default

The function returns a number that indicates which string entry was found. For instance, if the third string entry was found, the function returns a three (3).

Example

Here is an example:

This function statement	Returns	Assuming
ListInList(@("e_codes"), "ABC,AB,DE,A,GFHI,ABCD")	1	Field *e_codes* contains: *ABC,A.*
ListInList(GetValue("e_codes"), "ABC,AB,DE,A,GFHI")	2	DAL variable, *e_codes*, contains: *AB,abcd.*

This function statement	Returns	Assuming
ListInList(?("e_codes"), "ABC,AB,DE,A,GFHI,ABCD")	3	XDB entry *e_codes* returns: *DE,a*.
ListInList(?("e_codes"), ?("t_codes"))	4	XDB entry *e_codes* returns *A*. The entry *t_codes* contains: *ABC,AB,DE,A,GFHI, ABCD*.
ListInList(?("e_codes"), "ABC,AB,DE,A,GFHI,ABCD")	0	XDB entry *e_codes* returns: *XYZ*.

If you omit the first parameter, you get the data from the current field. If you omit the second parameter, you receive this error message:

```
Wrong number of parameters
```

Here is another example. For this example assume the following parameters contain:

- GetValue(col_name1) results in the character string: AA, EE.
- DAL variable col_name1_codes contains the string: EEacb,XXEE,EE,AEEAC.
- GetValue(ca_codes) contains the string: Xxaab,YYEE, EE,AA,AeeAC.

This statement	Returns
#rc = ListInList(GetValue(col_name1), col_name1_codes)	3
#rc = ListInList(GetValue(col_name1), GetValue(col_name1_codes))	4

The return value for the above example returns a four (4) because two spaces exist between the comma and EE.

Keep in mind:

- The search is not case-sensitive. This means *A* will match *a*.
- Spaces are considered. This means the system will find no matches in the following examples:

```
ListInList("Steel,Wood", " Steel,Aluminum")
ListInList("Steel,Wood", "Steel ,Aluminum")
ListInList("Steel,Wood", "Aluminum,Steel ")
```

1641
PPS

Using the New DAL Functions

You can use the following new DAL functions:

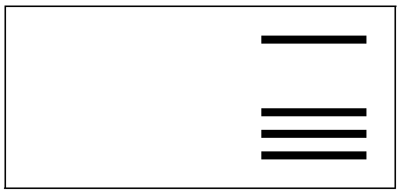
- DashCode on page 100
- ParseListCount on page 103

- ParseListItem on page 104
- PathCreate on page 106
- PathExist on page 106
- PrinterID on page 107
- PrinterOutputSize on page 107
- SetFont on page 107

DashCode

Use this function to build a value to assign a series of fields from the binary value of an integer. This is sometimes called a *dash code*. A dash code is a type of OMR mark that is read by certain mail, binding, or inserting equipment.

A dash code is a series of horizontal lines aligned in a column — each usually around 1/2 to one inch in length — that are typically on the left or right edge of the paper. The marks are usually expected to be in a uniform (fixed) position. Here is an example of a dash code:



Dash codes can be used, for instance, to represent the beginning or end of a set of pages that are associated in some way. The marks might indicate sequencing, first page, last page, staple requirements, additional pages to be inserted at a given point, the envelope size, or binding requirements.

NOTE: The exact meaning, order, and position of each mark depends on the finishing equipment you are using. Check the specifications that came with your equipment and assign the values appropriately.

Syntax

```
DashCode(Value, Bits, RootName, Image, Form, Group, OnString,
OffString, Direction, AltLens)
```

Parameter	Description
Value	Each bit of the value parameter is tested for a one (1) or zero (0). If the bit is one (1), it is considered on and the character you specify in the OnString parameter is appended to the string result being built. If the bit is zero (0), the OffString parameter is appended to the string result.
Bits	This parameter identifies how many of the bits from the value need to be evaluated. By default all 32 bits are evaluated. If you specify a negative or zero value, you'll get an empty string.
RootName	This parameter identifies the initial portion of a series of field names that are to be the repository for the OnString and OffString filled values. The bit number referenced will be appended to each name to form the final name expected to be found on the resulting image. For instance, if *MVALUE_* is passed as the RootName, the first fill value is assigned to *MVALUE_1*, the second to *MVALUE_2*, *MVALUE_3*, and so on, until the maximum number of bits specified are all mapped. If all 32 bits are mapped, the last field would be *MVALUE_32*. The associated fields will be filled to their defined length. In most dash code (barcode) type situations, you will want all the fields to be the same length.
Image	The name of an image that contains the field named. You can enter an asterisk (*) to tell the function to search all images. Keep in mind, however, that including an asterisk (*) degrades performance.
Form	The name of a form that contains the image and/or field named. You can enter an asterisk (*) to tell the function to search all forms. Keep in mind, however, that including an asterisk (*) degrades performance.
Group	The name of the form group that contains the form, image, or field. You can enter an asterisk (*) to tell the function to search all groups. Keep in mind, however, that including an asterisk (*) degrades performance.
OnString	By default, OnString is an underscore (_). You can specify alternative OnString and OffString values and each can be more than one character. The two parameters do not have to be the same length. If you define multiple characters, the fill value will repeat those characters as necessary to fill the entire field. If the field length is not evenly divisible by the length of the string you enter, a partial copy of the string can appear at the end. For instance, suppose the field length is five; OnString is *ABC*; and OffString is *XY*. If the bit value for this field is one (1), the fill value generated will be: *ABCAB*. If the bit value is zero (0), the fill value generated for this field will be *XYXYX*.

Parameter	Description
OffString	By default, Offstring is a space ( ). You can specify alternative OnString and OffString values and each can be more than one character. The two parameters do not have to be the same length. If you define multiple characters, the fill value will repeat those characters as necessary to fill the entire field. If the field length is not evenly divisible by the length of the string you enter, a partial copy of the string can appear at the end.

Direction	Note that integer values have 32 bits. When looking at the value in binary form, bit 1 is on the right and bit 32 is on the left. To override the default behavior, you can supply a non-zero Direction parameter.
	`0000 0000 0000 0000 0000 0000 0000 0000`
	`Bit 32 │ │ Bit 1`

AltLens	The final parameter is a comma-delimited pattern string to identify alternate lengths for each field associated with the bits. By default, each field is assigned a value equal to its defined length. If you want to use a different length, supply the appropriate lengths in string form separated by commas. The order of the length values starts with the field associated with the first bit, followed by the length for the second field, and so on. Remember the *first bit* is determined by the direction parameter. If you do not provide enough length values to match the number of bits you are using, the undefined positions will default to the default field length.

The return value indicates the number of fields assigned. A return value of zero (0) means that no fields were found.

Example

Here are some examples:

#val = 11 (which is 1011 in binary)

```
DASHCODE(#val, 4, "BFLD");
```

Assuming that *BFLD* is a root field name and matching fields are located on the current image, the following assignments are made. Further assume that each field is five characters in length.

BFLD1 is assigned "_____"
BFLD2 is assigned "_____"
BFLD3 is assigned "   " (five spaces)
BFLD4 is assigned "_____"

```
DASHCODE(#val, 4, "BFLD", , , , "A", "B");
```

This example uses the parameters to supply different OnString and OffString parameters.

BFLD1 is assigned "AAAA"
BFLD2 is assigned "AAAA"
BFLD3 is assigned "BBBB"
BFLD4 is assigned "AAAA"

```
DASHCODE(#val, 4, "BFLD", , , , "A","B",1);
```

Note the Direction parameter was used to reverse the order of the bits interpretation.

BFLD1 is assigned "AAAA"
BFLD2 is assigned "BBBB"
BFLD3 is assigned "AAAA"
BFLD4 is assigned "AAAA"

```
DASHCODE(#val, 4, "AB", "XYZ", 0, "1,2,3,5");
```

In this example, the last parameter applies differing lengths to the fields you are mapping. This example also uses alternate OnString and OffString parameters and uses text greater than one character. In this case, the string may be truncated or repeated as necessary to fill the field length.

BFLD1 is assigned "A"
BFLD2 is assigned "AB"
BFLD3 is assigned "XYZ"
BFLD4 is assigned "ABAB" or "ABABA"

Note that the last example indicates two possible results. During entry in PPS, the field length is considered paramount and cannot be overridden. During batch operations, it is possible for the data length to override the field length.

ParseListCount

Use this function to count the indexed components within the formatted text.

NOTE: Use the ParseListCount and ParseListItem functions when accepting tokenized (comma or semicolon-delimited) data, such as data from a spreadsheet program or other application. These are sometimes referred to as CSV (comma separated value) files.

Syntax

```
ParseListCount(String, Separator)
```

Parameter	Description
String	Enter the formatted string you want the system to search and parse.
Separator	Enter the list of character separators used within the formatted text parameter. If you omit this parameter, the system uses semicolons and commas.

This function returns the number of formatted items found within the String parameter. If the String parameter text starts with delimiter characters, those characters are skipped.

If you do not have at least a space character between delimiters, this will not be identified as a separate index item.

NOTE: You can use the ParseListItem function to return the text components parsed from the formatted text.

Example

For these examples, assume xString = “A,B;C”

```
value = ParseListCount(xString)
```

The value is *3*.

```
value = ParseListCount(xString,";")
```

The value is *2*. In this example the parameter overrides and assigns only a semicolon as a valid separator. Therefore, there are two items within this string.

For these examples, assume xString = ";A;,B,;C"

```
value = ParseListCount(xString)
```

The value is *3*. If the formatted string starts with separator characters, these characters are skipped. Note that adjacent separators are treated as a single separation.

For these examples, assume xString = "; ,A; ,B;"

```
value = ParseListCount(xString)
```

The value is *4*. Note the intervening character – a space - between some of the separator characters.

```
value = ParseListCount(xString,";")
```

The value is *2*. This overrides and assigns only a semicolon as the format separator, therefore there are only two components. Also note that although there are three separators, the first one that starts the string and the final one that ends the string are also ignored.

ParseListItem

Use this function to return indexed components from the formatted text.

NOTE: Use the ParseListCount and ParseListItem functions when accepting tokenized (comma or semicolon-delimited) data, such as data from a spreadsheet program or other application. These are sometimes referred to as CSV (comma separated value) files.

Syntax

```
ParseListItem(String, Item, Separator)
```

Parameter	Description
String	Enter a formatted string to search and parse.
Item	Enter the number of the item you want from within that formatted string. If you omit this parameter, the first item parsed from the formatted text is returned.
Separator	Enter a list of character separators used within the formatted text parameter. If you omit this parameter, the semicolons and commas are used.

The return value is a string of text. If the formatted text contains leading or trailing spaces on items formatted within it, they are not removed. You can use the Trim function on the returned text if you do not want the spaces.

If the first parameter text starts with delimiter characters, they will be skipped. Because the function will return spaces, you know when you have exceeded the number of items formatted within the string when you get an empty string returned.

NOTE: If you do not have at least a space character between delimiters, this will not be identified as a separate index item.

Example

Here are some examples. Assume xString = “A,B;C”

```
value = ParseListItem(xString)
```

The value is *A*.

```
value = ParseListItem(xString,3)
```

The value is *C* because the default separators include both commas and semicolons.

```
value = ParseListItem(xString,1,”;”)
```

The value is *A,B*. Note in this example the third parameter overrides and assigns only the semicolon as a valid separator. Therefore, the first item includes all text up to the first semicolon.

For these examples, assume xString = “;A;,B,;C”

```
value = ParseListItem(xString)
```

The value is *A*. Note that if the formatted string starts with separator characters they are skipped.

```
value = ParseListItem(xString,2)
```

The value is *B*. Note again how adjacent separators without intervening characters (or space) are skipped. Therefore the semicolon and comma (;,) between the A and B are treated as a single separation.

```
value = ParseListItem(xString,3)
```

The value is *C*. Note again how adjacent separators without intervening characters (or space) are skipped. Therefore the semicolon and comma (;,) between the *A* and *B* are treated as a single separation and the semicolon and comma (;,) between the *B* and *C* are also treated as a single separation.

```
value = ParseListItem(xString,3,”,”)
```

The value is *;C*. Note the third parameter overrides and assigns only the comma as a valid separator. Therefore the third index item includes all text following the second comma to the end of the string (because no other separators were encountered).

For these examples, assume xString = “; ,A; ,B;”

```
value = ParseListItem(xString)
```

The value is a space. Note that there is at least one intervening character —a space — between the first set of separator characters.

```
value = ParseListItem(xString,2)
```

The value is *A*.

```
value = ParseListItem(xString,3)
```

The value is a space.

```
value = ParseListItem(xString,4)
```

The value is *B*.

```
value = ParseListItem(xString,5)
```

The value is an empty string because this index item exceeds the list of items provided.

PathCreate

Use this function to create the parameter subdirectory path if it does not already exist. The function assumes all of the text you pass in is a path and does not remove any of it before it tries to verify or create the path.

The function creates multiple subdirectories as necessary in an attempt to satisfy the request.

NOTE: The PathCreate and PathExist functions let you create paths and verify that paths exist. These are useful, for instance, if you are trying to create printed output and organize that output into subdirectories on disk. You can do this using one of the print callback methods that support a DAL script.

Syntax

```
PathCreate(Path)
```

Parameter	Description
Path	Enter the full path you want the system to verify or create.

The function returns zero (0) if it cannot create the path requested. Anything else means the path now exists, but is not an indication that it had to be created.

NOTE: This function is not valid on the MVS, OS/390, ZOS operating systems.

PathExist

Use this function to take the parameter path you provide and check for its existence. This function does not create subdirectories.

NOTE: The PathCreate and PathExist functions let you create paths and verify that paths exist. These are useful, for instance, if you are trying to create printed output and organize that output into subdirectories on disk. You can do this using one of the print callback methods that support a DAL script.

Syntax

```
PathExist(Path)
```

Parameter	Description
Path	Enter the full path you want the system to verify.

The function returns zero (0) if the path is invalid. Anything else indicates the path you provided exists.

NOTE: This function merely checks for the existence of the path you specified. Provided the path does exist, this is not an indication that the process will be able to access or create files within that path.

PrinterID

Use this function to return the active printer ID assigned during a batch processing run. The printer ID is a string of text associated with the current batch output and normally determined via INI option during a batch run. The IDs are associated from the PrinterInfo control group with each batch printer definition.

You can use this ID, for instance, when naming print file. For example, you might want all the files from one printer ID in a separate location or have the names prefixed in a certain manner.

Syntax

```
PrinterID( )
```

There are no parameters for this function.

NOTE: The printer ID is only valid during a batch print operation and calling the function at other times returns the last value assigned or an empty string.

PrinterOutputSize

Use this function to get the approximate size of the current print output file during a batch print operation.

Syntax

```
PrinterOutputSize( );
```

There are no parameters for this function.

This function is only available during Documaker batch process operations, such as GenPrint, and only returns a non-zero value if a print stream is actively being built and written to a physical file on disk.

NOTE: When printing through the Windows GDI device, there is no physical file and therefore the value returned is unreliable and may be zero.

SetFont

Use this function to change the font on a field. For instance, you can use SetFont on non-multi-line text fields or barcode fields. You cannot use the SetFont function to reformat a text area.

Syntax SetFont(FontID, Field, Image, Form, Group)

Parameter	Description	Defaults to...
FontID	Enter the font ID of the font to which you want to change.A FontID of less than one (1) causes the function to fail.	no default
Field	(optional) A field name that identifies a multi-line text area. This is the field that receives the appended text.	the current field
Image	(optional) Name of an image that contains the field named.	the current image
Form	(optional) Name of a form that contains the image and/or field named.	the current form
Group	(optional) Name of the form group that contains the form, image, and/or field named.	the current form group

This function returns one (1) on success or zero (0) on failure.

The system applies the font change to the first field that matches the criteria.

1807
PPS

Creating Variable Length Records from DAL Using Flat Files

When you use DAL database functions, such as DBOpen, DBClose, and so on, to write flat files, the record length is usually fixed and data is padded with spaces to equal the maximum size of the record. This feature lets you specify that no trailing spaces are to be output. You would typically use this capability to output flat files used to create index information you will import into a 3rd-party application, such as FileNET.

To specify no trailing spaces, include the following syntax in your DAL script:

```
DBOPEN(FN_LogFile,"ASCII",".\deflib\filenet.dfd",
"READ&WRITE&TRUNCATE&CREATE_IF_NEW&CLIPSPACES");
```

CLIPSPACES tells the system to remove any trailing spaces.

Keep in mind that CLIPSPACES only affects flat files. For the rest of the databases, each column is set separately and no trailing space exists on the whole record.

1995
RPS

Converting Integers to Characters

This version includes these new DAL functions which you can use to convert integers into characters and vice versa.

Char

Use this function to convert an integer into a single character.

Syntax

```
Char (Integer)
```

Parameter	Description	Defaults to...
Integer	An integer value that ranges zero (0) to 255.	No default.

Example

Here is an example:

```
what_char = Char (64)
```

The variable, *what_char*, is set to the character: '@'.

CharV

Use this function to convert a single character into an integer value.

Syntax

```
CharV (String)
```

Parameter	Description	Defaults to...
String	A character string. If the string contains more than one character, only the first character is converted. The remaining characters are ignored.	No default.

Example

In this example, assume the variable, char_to_convert, contains the single character: "@".

```
#_the_integer = CharV(char_to_convert)
```

The integer variable, *#_the_integer*, is set the value: *64*.

In this example, assume the variable, the_string, contains the characters: "@()".

```
#_the_integer = CharV(the_string)
```

The integer variable, *#_the_integer*, is set the value: *64*. The remaining characters are ignored.

2143
PPS

Paginating Forms Using DAL Functions

This version includes a new DAL function/procedure (PaginateForm) and changes to the AddImage and DelImage DAL functions to make it easier to control the pagination of forms.

The PaginateForm function lets you apply image origins and re-paginate the form if necessary. During this re-pagination, the function will create or delete pages as needed.

The AddImage and DelImage DAL functions have been enhanced to include a new parameter you can use to force re-pagination after the affected image has been manipulated.

PaginateForm

Use this function/procedure to apply image origins and re-paginate the form if necessary. During this re-pagination, the function will create or delete pages as needed.

Syntax

```
PaginateForm (Form, Group)
```

Parameter	Description
Form	(Optional) If you omit this parameter, the current form controlling the active script is paginated. If you include the name of a form, that form is located and paginated. You can include the occurrence indicator (a backslash followed by a number, such as BIZ\3) to indicate a specific occurrence of the form to find and paginate. If you do not specify an occurrence with the name, the first occurrence of the form is paginated.
Group	(Optional) This parameter identifies the Key2 or GroupName2-level parent that contains the form. This is sometimes referred to as the *line of business* that contains the form. If you omit the Group parameter, the system tries to locate the named form within the current group that is controlling the execution of the script.

You can call PaginateForm as a function or procedure. As a function, it returns a one (1) if the requested form is located or a zero (0) if it could not be located.

Note that if the form is found and paginated, there may not be any visible change to the document. The form layout is determined when you design the form and by the application of image origin rules.

AddImage

The AddImage function now includes an optional parameter that indicates whether form pagination should occur after the image is added. Form pagination includes the application of image origin rules to determine whether new pages are required for the pre-defined page sizes.

```
AddImage(FAP, Image, Form, Group, Flag, Paginate)
```

Parameter	Description
Paginate	(Optional) This parameter follows the Flag parameter. If you enter anything other than a zero (0), it tells the system you do want form pagination to occur upon successful inclusion of the new image. If the image does contain an origin rule and you omit the Paginate option or set it to zero (0), the image origin rule executes upon insertion. Whether the inserted image has an origin rule or not, the positioning of this image when the Paginate option is omitted or zero (0) does not cause the entire form to be re-paginated. This means if the placement of the image causes it to overlap another image or to be out of the page boundary, no additional re-pagination occurs. If you are manipulating multiple images in series, you may want to conclude your script with a call to PaginateForm to make sure the entire form is re-paginated. Here is an example: `AddImage( "myFAP", "mainImage" , , , 1,1)` This example omits the Form and Group parameters, but does specify the Flag parameter as well as the Pagination parameter.

NOTE: If you enter zero (0) or omit this parameter, the function works as it did in prior versions.

DelImage

The DelImage function now includes an optional parameter that indicates whether form pagination should occur after the image is deleted.

```
DelImage( Image, Form, Group, Paginate)
```

Parameter	Description
Paginate	(Optional) This parameter follows the Group parameter. If you enter anything other than a zero (0), it tells the system that you want form pagination to occur upon the successful removal of the image. If you omit this parameter or enter zero(0), the image is deleted, but no other images are moved to occupy the space left vacant. Subsequent form re-pagination and the application of image origins may change the layout of the form. Here is an example: `DelImage( "myImage", , , 1)` This example omits the Form and Group parameters, but does include the Paginate parameter.

NOTE: If you enter zero (0) or omit this parameter, the function works as it did in prior versions.

2158
PPS

Using the STRCompare Function

Use this function to compare two strings with case a consideration. In normal DAL string expressions, strings are compared in a case-insensitive manner. For example, the system would normally evaluate the following strings to be equal:

```
ABC
abc
```

If, however, you use the STRCompare function, the system will consider case and judge these strings to not be equal.

Note: The best way to use this function is to test for equality. For instance, use this function to test two strings and compare for a zero (0) value being returned to indicate the strings are equal or a non-zero value to indicate they are unequal.

You can use this function to determine if one string is greater or less than the other, but the result can be confusing if the strings contain mixed case or have different lengths.

Syntax

```
#RTN = STRCOMPARE ( String1 , String2 , #count )
```

Parameter	Description	Defaults to...
String1	The text for the first string you want to compare	an empty string
String2	The text for the second string you want to compare	an empty string
#Count	Optional. The number of characters to compare. If you enter a value greater than zero, the system compares that number of characters. If you enter zero (0) or less, the system compares all characters. If you enter a value greater than the length of either string, the system pads the strings with blank characters to match the number of characters you specified.	-1 which indicates that all characters will be compared.

If String1 and String2 compare as equal, the system returns a zero (0).

The system returns a negative one (-1) if String1 is less than String2.

The system returns a one (1) if String1 is greater than String2.

Example

Assume String1 is *ABCDEF* and String2 is *ABCdef* in these examples:

This example	Returns
#RTN = STRCompare(string1 , string2)	-1
#RTN = STRCompare(string2 , string1)	1
#RTN = STRCompare(string1 , string2 , 3)	0

2197 PPS

USING THE TIME ZONE DAL FUNCTIONS

This feature adds these new time zone functions to the Document Automation Language (DAL):

Use this function	To
TimeZone	Return the system's time zone setting or validate a time zone setting.
TimeZone2TimeZone	Convert date and time values from one geographic region into date and time values that are local to another geographic region. The functions will also adjust for daylight savings time as needed.

NOTE: These functions are not available on mainframe platforms like z/OS. They are only available on Windows and UNIX platforms.

These functions use the International Components for Unicode (ICU) library. The ICU system time zones are derived from the tz database (also known as the Olson database) available at...

ftp://elsie.nci.nih.gov/pub

This is the data used across much of the industry, including by UNIX systems.

The ICU time zone functionality supports

- Standard time zones, such as Eastern Standard Time (EST), Central Standard Time (CST), and so on.
- Time zone IDs defined in the standard Olson data used by UNIX systems. These time zone IDs use the following format:

  ```
  continent/city or ocean/city
  ```

  For example, *America/Los_Angeles* is an ID for Pacific Standard Time.
- Custom time zones based on Greenwich Mean Time (GMT), in this format:

  ```
  "GMT[+|-]hh[[:]mm]")
  ```

TimeZone

Use this function to return the system's time zone setting or to make sure a time zone is valid.

Syntax

```
TimeZone (TimeZone)
```

Parameter	Description	Defaults to...
TimeZone	Optional. If you include a time zone string, the functions makes sure that string is valid. If it is invalid, an empty string is returned.	If you omit this parameter, the function returns the system's time zone setting.

Examples

Here are some examples:

This example returns the system time zone, such as *America/New_York*:

```
T1 = TimeZone()
```

This example checks to see if a time zone string, such as *Europe/London*, is valid:

```
T1 = 'Europe/London'
T2 = TimeZone(T1)
if (T2 = '') then
Print_It(T1 & 'is not a valid time zone string')
else
Print_It(T1 & 'is a valid time zone string')
end
```

TimeZone2TimeZone

Use this function to convert date and time values from one geographic region into date and time values that are local to another geographic region. The function will also adjust for daylight savings time as needed.

Syntax

```
TimeZone2TimeZone (PrefixName, TimeZone, NewTimeZone)
```

Parameter	Description	Defaults to...
PrefixName	Enter the prefix name associated with variables that will be used to hold date and time settings. Here are some examples: PrefixName.day PrefixName.month PrefixName.year PrefixName.hour PrefixName.minutes PrefixName.seconds	No default. You must define a PrefixName.
TimeZone	Optional. Enter the time zone used for the PrefixName variables. If you enter an invalid time zone string, the function returns a value of zero (0) and sets variables associated with the PrefixName to zero (0).	If you omit this parameter, the function uses the system's default time zone.
NewTimeZone	Optional. Enter the time zone by which you want to adjust the values in the PrefixName variables. If you enter an invalid time zone string, the function returns a value of zero (0) and sets variables associated with the PrefixName to zero (0).	If you omit this parameter, the function uses the system’s default time zone.

If you define these variables, the system uses the PrefixName and time you specified and converts that time to the equivalent time in the location you specified via the NewTimeZone parameter.

If you do not define these variables, the system creates these variables based on the PrefixName you entered and assigns values into these variables based on the current date and time.

If there are no errors, the function returns a non-zero value.

Examples

Here are some examples:

This example creates date and time variables using *tz* as a prefix (tz.day, tz.month, tz.year, tz.hour, tz.minute, tz.second) and stores the current date and time values based on the system's time zone:

```
TimeZone2TimeZone('tz', ,)
Print_It('Date:' & Date(, tz.day, tz.month, tz.year))
Print_It('Time:' & Time(, tz.hour, tz.minute, tz.second))
```

This example converts date and time variables (tz.xxxx) that use the system’s time zone into GMT date and time:

```
TimeZone2TimeZone('tz', , 'GMT')
Print_It('GMT Date:' & Date(, tz.day, tz.month, tz.year))
Print_It('GMT Time:' & Time(, tz.hour, tz.minute, tz.second))
```

This example converts a current America/New_York date and time into an Australia/Melbourne date and time:

```
tz.day = ''
tz.month = ''
tz.year = ''
tz.hour = ''
tz.minute = ''
tz.second = ''
if (TimeZone2TimeZone('tz', 'America/New_York', 'Australia/
Melbourne')) then
Print_It('Australia/Melbourne Date:' & Date(, tz.day, tz.month,
tz.year))
Print_It('Australia/Melbourne Time:' & Time(, tz.hour, tz.minute,
tz.second))
else
Print_it('Error calling TimeZone2TimeZone')
end
```

List of Time Zones

Here is a list of the various ICU time zones:

ICU System Time Zones

ACT	AET	Africa/Abidjan	Africa/Accra
Africa/Addis_Ababa	Africa/Algiers	Africa/Asmera	Africa/Bamako
Africa/Bangui	Africa/Banjul	Africa/Bissau	Africa/Blantyre
Africa/Brazzaville	Africa/Bujumbura	Africa/Cairo	Africa/Casablanca
Africa/Ceuta	Africa/Conakry	Africa/Dakar	Africa/Dar_es_Salaam

ICU System Time Zones			
Africa/Djibouti	Africa/Douala	Africa/El_Aaiun	Africa/Freetown
Africa/Gaborone	Africa/Harare	Africa/Johannesburg	Africa/Kampala
Africa/Khartoum	Africa/Kigali	Africa/Kinshasa	Africa/Lagos
Africa/Libreville	Africa/Lome	Africa/Luanda	Africa/Lubumbashi
Africa/Lusaka	Africa/Malabo	Africa/Maputo	Africa/Maseru
Africa/Mbabane	Africa/Mogadishu	Africa/Monrovia	Africa/Nairobi
Africa/Ndjamena	Africa/Niamey	Africa/Nouakchott	Africa/Ouagadougou
Africa/Porto-Novo	Africa/Sao_Tome	Africa/Timbuktu	Africa/Tripoli
Africa/Tunis	Africa/Windhoek	AGT	America/Adak
America/Anchorage	America/Anguilla	America/Antigua	America/Araguaina
America/Argentina/ Buenos_Aires	America/Argentina/ Catamarca	America/Argentina/ ComodRivadavia	America/Argentina/ Cordoba
America/Argentina/Jujuy	America/Argentina/ La_Rioja	America/Argentina/ Mendoza	America/Argentina/ Rio_Gallegos
America/Argentina/ San_Juan	America/Argentina/ Tucuman	America/Argentina/Ushuaia	America/Aruba
America/Asuncion	America/Atikokan	America/Atka	America/Bahia
America/Barbados	America/Belem	America/Belize	America/Blanc-Sablon
America/Boa_Vista	America/Bogota	America/Boise	America/Buenos_Aires
America/Cambridge_Bay	America/Campo_Grande	America/Cancun	America/Caracas
America/Catamarca	America/Cayenne	America/Cayman	America/Chicago
America/Chihuahua	America/Coral_Harbour	America/Cordoba	America/Costa_Rica
America/Cuiaba	America/Curacao	America/Danmarkshavn	America/Dawson
America/Dawson_Creek	America/Denver	America/Detroit	America/Dominica
America/Edmonton	America/Eirunepe	America/El_Salvador	America/Ensenada
America/Fort_Wayne	America/Fortaleza	America/Glace_Bay	America/Godthab
America/Goose_Bay	America/Grand_Turk	America/Grenada	America/Guadeloupe
America/Guatemala	America/Guayaquil	America/Guyana	America/Halifax
America/Havana	America/Hermosillo	America/Indiana/ Indianapolis	America/Indiana/Knox

ICU System Time Zones			
America/Indiana/Marengo	America/Indiana/Petersburg	America/Indiana/Vevay	America/Indiana/Vincennes
America/Indianapolis	America/Inuvik	America/Iqaluit	America/Jamaica
America/Jujuy	America/Juneau	America/Kentucky/Louisville	America/Kentucky/Monticello
America/Knox_IN	America/La_Paz	America/Lima	America/Los_Angeles
America/Louisville	America/Maceio	America/Managua	America/Manaus
America/Martinique	America/Mazatlan	America/Mendoza	America/Menominee
America/Merida	America/Mexico_City	America/Miquelon	America/Moncton
America/Monterrey	America/Montevideo	America/Montreal	America/Montserrat
America/Nassau	America/New_York	America/Nipigon	America/Nome
America/Noronha	America/North_Dakota/Center	America/North_Dakota/New_Salem	America/Panama
America/Pangnirtung	America/Paramaribo	America/Phoenix	America/Port-au-Prince
America/Port_of_Spain	America/Porto_Acre	America/Porto_Velho	America/Puerto_Rico
America/Rainy_River	America/Rankin_Inlet	America/Recife	America/Regina
America/Rio_Branco	America/Rosario	America/Santiago	America/Santo_Domingo
America/Sao_Paulo	America/Scoresbysund	America/Shiprock	America/St_Johns
America/St_Kitts	America/St_Lucia	America/St_Thomas	America/St_Vincent
America/Swift_Current	America/Tegucigalpa	America/Thule	America/Thunder_Bay
America/Tijuana	America/Toronto	America/Tortola	America/Vancouver
America/Virgin	America/Whitehorse	America/Winnipeg	America/Yakutat
America/Yellowknife	Antarctica/Casey	Antarctica/Davis	Antarctica/DumontDUrville
Antarctica/Mawson	Antarctica/McMurdo	Antarctica/Palmer	Antarctica/Rothera
Antarctica/South_Pole	Antarctica/Syowa	Antarctica/Vostok	Arctic/Longyearbyen
ART	Asia/Aden	Asia/Almaty	Asia/Amman
Asia/Anadyr	Asia/Aqtau	Asia/Aqtobe	Asia/Ashgabat
Asia/Ashkhabad	Asia/Baghdad	Asia/Bahrain	Asia/Baku
Asia/Bangkok	Asia/Beirut	Asia/Bishkek	Asia/Brunei

ICU System Time Zones			
Asia/Calcutta	Asia/Choibalsan	Asia/Chongqing	Asia/Chungking
Asia/Colombo	Asia/Dacca	Asia/Damascus	Asia/Dhaka
Asia/Dili	Asia/Dubai	Asia/Dushanbe	Asia/Gaza
Asia/Harbin	Asia/Hong_Kong	Asia/Hovd	Asia/Irkutsk
Asia/Istanbul	Asia/Jakarta	Asia/Jayapura	Asia/Jerusalem
Asia/Kabul	Asia/Kamchatka	Asia/Karachi	Asia/Kashgar
Asia/Katmandu	Asia/Krasnoyarsk	Asia/Kuala_Lumpur	Asia/Kuching
Asia/Kuwait	Asia/Macao	Asia/Macau	Asia/Magadan
Asia/Makassar	Asia/Manila	Asia/Muscat	Asia/Nicosia
Asia/Novosibirsk	Asia/Omsk	Asia/Oral	Asia/Phnom_Penh
Asia/Pontianak	Asia/Pyongyang	Asia/Qatar	Asia/Qyzylorda
Asia/Rangoon	Asia/Riyadh	Asia/Riyadh87	Asia/Riyadh88
Asia/Riyadh89	Asia/Saigon	Asia/Sakhalin	Asia/Samarkand
Asia/Seoul	Asia/Shanghai	Asia/Singapore	Asia/Taipei
Asia/Tashkent	Asia/Tbilisi	Asia/Tehran	Asia/Tel_Aviv
Asia/Thimbu	Asia/Thimphu	Asia/Tokyo	Asia/Ujung_Pandang
Asia/Ulaanbaatar	Asia/Ulan_Bator	Asia/Urumqi	Asia/Vientiane
Asia/Vladivostok	Asia/Yakutsk	Asia/Yekaterinburg	Asia/Yerevan
AST	Atlantic/Azores	Atlantic/Bermuda	Atlantic/Canary
Atlantic/Cape_Verde	Atlantic/Faeroe	Atlantic/Jan_Mayen	Atlantic/Madeira
Atlantic/Reykjavik	Atlantic/South_Georgia	Atlantic/St_Helena	Atlantic/Stanley
Australia/ACT	Australia/Adelaide	Australia/Brisbane	Australia/Broken_Hill
Australia/Canberra	Australia/Currie	Australia/Darwin	Australia/Hobart
Australia/LHI	Australia/Lindeman	Australia/Lord_Howe	Australia/Melbourne
Australia/North	Australia/NSW	Australia/Perth	Australia/Queensland
Australia/South	Australia/Sydney	Australia/Tasmania	Australia/Victoria
Australia/West	Australia/Yancowinna	BET	Brazil/Acre
Brazil/DeNoronha	Brazil/East	Brazil/West	BST

ICU System Time Zones			
Canada/Atlantic	Canada/Central	Canada/East-Saskatchewan	Canada/Eastern
Canada/Mountain	Canada/Newfoundland	Canada/Pacific	Canada/Saskatchewan
Canada/Yukon	CAT	CET	Chile/Continental
Chile/EasterIsland	CNT	CST	CST6CDT
CTT	Cuba	EAT	ECT
EET	Egypt	Eire	EST
EST5EDT	Etc/GMT	Etc/GMT+0	Etc/GMT+1
Etc/GMT+10	Etc/GMT+11	Etc/GMT+12	Etc/GMT+2
Etc/GMT+3	Etc/GMT+4	Etc/GMT+5	Etc/GMT+6
Etc/GMT+7	Etc/GMT+8	Etc/GMT+9	Etc/GMT-0
Etc/GMT-1	Etc/GMT-10	Etc/GMT-11	Etc/GMT-12
Etc/GMT-13	Etc/GMT-14	Etc/GMT-2	Etc/GMT-3
Etc/GMT-4	Etc/GMT-5	Etc/GMT-6	Etc/GMT-7
Etc/GMT-8	Etc/GMT-9	Etc/GMT0	Etc/Greenwich
Etc/UCT	Etc/Universal	Etc/UTC	Etc/Zulu
Europe/Amsterdam	Europe/Andorra	Europe/Athens	Europe/Belfast
Europe/Belgrade	Europe/Berlin	Europe/Bratislava	Europe/Brussels
Europe/Bucharest	Europe/Budapest	Europe/Chisinau	Europe/Copenhagen
Europe/Dublin	Europe/Gibraltar	Europe/Guernsey	Europe/Helsinki
Europe/Isle_of_Man	Europe/Istanbul	Europe/Jersey	Europe/Kaliningrad
Europe/Kiev	Europe/Lisbon	Europe/Ljubljana	Europe/London
Europe/Luxembourg	Europe/Madrid	Europe/Malta	Europe/Mariehamn
Europe/Minsk	Europe/Monaco	Europe/Moscow	Europe/Nicosia
Europe/Oslo	Europe/Paris	Europe/Prague	Europe/Riga
Europe/Rome	Europe/Samara	Europe/San_Marino	Europe/Sarajevo
Europe/Simferopol	Europe/Skopje	Europe/Sofia	Europe/Stockholm
Europe/Tallinn	Europe/Tirane	Europe/Tiraspol	Europe/Uzhgorod
Europe/Vaduz	Europe/Vatican	Europe/Vienna	Europe/Vilnius

ICU System Time Zones			
Europe/Volgograd	Europe/Warsaw	Europe/Zagreb	Europe/Zaporozhye
Europe/Zurich	Factory	GB	GB-Eire
GMT	GMT+0	GMT-0	GMT0
Greenwich	Hongkong	HST	Iceland
IET	Indian/Antananarivo	Indian/Chagos	Indian/Christmas
Indian/Cocos	Indian/Comoro	Indian/Kerguelen	Indian/Mahe
Indian/Maldives	Indian/Mauritius	Indian/Mayotte	Indian/Reunion
Iran	Israel	IST	Jamaica
Japan	JST	Kwajalein	Libya
MET	Mexico/BajaNorte	Mexico/BajaSur	Mexico/General
Mideast/Riyadh87	Mideast/Riyadh88	Mideast/Riyadh89	MIT
MST	MST7MDT	Navajo	NET
NST	NZ	NZ-CHAT	Pacific/Apia
Pacific/Auckland	Pacific/Chatham	Pacific/Easter	Pacific/Efate
Pacific/Enderbury	Pacific/Fakaofo	Pacific/Fiji	Pacific/Funafuti
Pacific/Galapagos	Pacific/Gambier	Pacific/Guadalcanal	Pacific/Guam
Pacific/Honolulu	Pacific/Johnston	Pacific/Kiritimati	Pacific/Kosrae
Pacific/Kwajalein	Pacific/Majuro	Pacific/Marquesas	Pacific/Midway
Pacific/Nauru	Pacific/Niue	Pacific/Norfolk	Pacific/Noumea
Pacific/Pago_Pago	Pacific/Palau	Pacific/Pitcairn	Pacific/Ponape
Pacific/Port_Moresby	Pacific/Rarotonga	Pacific/Saipan	Pacific/Samoa
Pacific/Tahiti	Pacific/Tarawa	Pacific/Tongatapu	Pacific/Truk
Pacific/Wake	Pacific/Wallis	Pacific/Yap	PLT
PNT	Poland	Portugal	PRC
PRT	PST	PST8PDT	ROC
ROK	Singapore	SST	Turkey
UCT	Universal	US/Alaska	US/Aleutian
US/Arizona	US/Central	US/East-Indiana	US/Eastern

ICU System Time Zones			
US/Hawaii	US/Indiana-Starke	US/Michigan	US/Mountain
US/Pacific	US/Pacific-New	US/Samoa	UTC
VST	W-SU	WET	Zulu

2256
PPS

USING THE SETLINK FUNCTION

You can use the new SetLink function to update a hyperlink setting in a variable field, a logo, or a text label.

Syntax

```
SetLink(Target,Parms,ObjectName,Section,Form,Key2,ObjectType)
```

Parameter	Description
Target	Enter the name of the target object (the HREF value). If the target object has a hyperlink type of *internal* or *target*, enter the name of the target object. If the target object has a hyperlink type of *external*, this parameter should contain a hypertext reference, such as: `www.skywiresoftware.com` and the Parms parameter should contain additional parameters to an HREF type link. Make sure this parameter contains valid HTML syntax.
Parms	Optional. Enter any link parameters (HREF parameters), such as a target frame or mouseover behavior. Here is an example: `"target="new"` Make sure this parameter contains valid HTML syntax.
ObjectName	Enter the name of the variable field, logo, or text label that contains the hyperlink. The system updates the first object found that matches your entry for this parameter.
Section	Optional. Enter the name of the image.
Form	Optional. Enter the name of the form.
Key2	Optional. Enter the name of the Key2 group.
ObjectType	Enter the type of object, such as (variable) Field, Logo, or Text (label). The default is Field.

Keep in mind...

- The object (variable field, logo, or text label) referenced by SetLink must have an initial hyperlink setting.
- You must make sure the Target and Parms parameters contain valid HTML syntax.

Example

Here is an example:

```
SETLINK("http://www.skywiresoftware.com", "target=new",
"Section2256", "FormQ1331TPG", , , "Text")
```

2283
PPS

USING THE ISPRINTOBJECT FUNCTION

Use the new IsPrintObject DAL function during banner processing or in another print operation to determine if the section (image), form, or group is printable. This determination is based on the current print recipient and the recipient copy count.

Syntax

```
IsPrintObject (Section,Form,Group);
```

Parameter	Description
Section	Enter the name of the section you want to check. If you omit this parameter, the system uses the current section.
Form	Enter the name of the form you want to check. If you omit this parameter, the system uses the current form.
Group	Enter the name of the group you want to check. If you omit this parameter, the system uses the current group.

NOTE: You can use this function outside of a print operation to determine if a section is printable, but a true (1) result is not a guarantee the section will print during the next print operation.

Example

Here is an example:

```
IsPrintObject();
```

This example checks the current section on the current form in the current group and returns a one (1) if that section is printable or a zero (0) if it is not.

2311
PPS

USING THE SETFORMDESC FUNCTION

Use this new DAL function to change the description of a form.

Syntax

```
SetFormDesc(NewDescription,Form,Group)
```

Parameter	Description
NewDescription	Enter the new description. The text you enter replaces any existing form description.
Form	Enter the name of the form for which you want to change its description. The default is the current form.
Group	Enter the name of the group which contains the form you specified in the Form parameter. The default is the current group.

This function returns one (1) if the form was found and the description was assigned. Otherwise, it returns 0 (zero) to indicate that no form was found based upon the parameters provided.

Example

Here is an example:

```
SetFormDesc("Cover Page",Form1,Group2)
```