Shared Objects

Documaker Shared Objects

Version 11.2

Features and Enhancements

Docucorp International, a subsidiary of Skywire Software, announces the release of Documaker Shared Objects— version 11.2 — for use with Docupresentment version 2.1 and Documaker Server version 11.2. This document describes the new shared objects features.

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. Classes are taught hands-on at Skywire Software's training facilities in Atlanta. For a list of courses, including fees and availability, please visit our web site at www.skywiresoftware.com.

Installation:

Please follow the installation instructions included in the readme file on the CD-ROM to install version 11.2.

2401 Internet Boulevard
Suite 201
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.) 866 844 8780**
(EMEA) +44 (0) 1372 366 222

contactus@skywiresoftware.com

CONTENTS

Features and Enhancements

Documaker Shared Objects version 11.2

Docucorp International, a subsidiary of Skywire Software, proudly announces the initial release of the Documaker Shared Objects version 11.2.

Documaker Shared Objects are features which work with multiple Skywire Software products. This release includes functionality that enhances Docupresentment version 2.1's ability to work with Documaker Server version 11.2.

This document provides detailed information on the specific features and enhancements included in this shared object release.

NOTE: The initial release of Documaker Shared Objects version 11.2 is for Windows Vista, Windows XP, Windows 2000, Windows 2003 Server, and UNIX.

System Enhancements

The following new and improved features are released as Documaker Shared Objects, version 11.2. If the license column is blank, the feature is used in both RPS and IDS. Otherwise, the License column indicates that the enhancement is used only by IDS or iDMS.

System	Description
iDMS	Features available only to iDocumaker Workstation license holders.
IDS	Features available only to Docupresentment license holders.
RPS	Features available only to Documaker license holders.

If you have any questions about your license, please contact your sales representative.

Feature	License	For more information, see
1734	iDMS	Automatically Updating iDocumaker Workstation on page 3
1736	IDS	Running Multiple Instances of the WIP Edit Plug-in on page 9
1844	IDS	Using IDS TPD Rules to Print Multi-page TIFF Files into PDF Files on page 9
1860	IDS	Checking WIP Records that are not on the User List on page 10
1926	RPS	Meeting the PDF for Archive Specification on page 12
2069	IDS	Using Manually-Edited HTML Forms with Real-Time HTML Processing on page 14
2070	IDS	Accessing IDS Attachment Variables in GenData on page 15
2075	IDS	Adding Form Line and Form Description Line Information on page 16
2078	IDS	Setting the Starting Record for the DPRUpdateFromMRL Rule on page 18
2081	IDS	Passing a WIP Record ID to the MergeWIP Rule on page 19
2082	IDS	Using the New DPRGetConfigList Rule on page 19
2090	IDS	Printer Support for WIP Edit on page 21
2092	IDS	Using WIP Edit with SiteMinder• on page 22
2093	IDS	Customizing the Execution of Documaker Server under Docupresentment on page 23
2100	IDS	Using the DPRUpdateFormsetFromXML Rule with HTML Entry on page 24
2101	IDS	Designating Read-Only Multi-Line Text Field Paragraphs on page 25
2102	IDS	Automatically Determining the PRINTBATCHES Value on page 25
2104	IDS	Using Enhanced DAL Functions for WIP Column Access with Docupresentment on page 26
2105	IDS	Generating File Names Based on Transaction Values on page 27

Feature	License	For more information, see
2114	IDS	Automatic Detection of Import File Type by Documaker Bridge Rules on page 31
2115	IDS	Using the DPRWipBatchPrint Rule on page 31
2120	IDS	Enhanced Stability when Running Documaker under Docupresentment on page 35
2124	IDS	Automatically Printing Upon Completion on page 36
2125	IDS	Using the New WIP Fields in V2 and XML Import Files on page 38
2129	IDS	Printing on Your Workstation Printer on page 43
2148	IDS	Outputting WIP Field Data Onto the XML Tree on page 43
2149	iDMS	Preventing the Session from Expiring When Working on a Form on page 45
2151	IDS	Using the New DPRGetUserList and DPRModifyUser Rules on page 45
2174	IDS	Embedding a Subset of a Font on page 52

1734
IDMS

Automatically Updating iDocumaker Workstation

You can use IDS to update a user's workstation with a new version of the iDocumaker Workstation executables. Users are notified of an update when a document is opened if a new version has been made available by the administrator.

NOTE: You must be using version 11.2 or higher of iDocumaker Workstation to use the automatic update feature.

The user can then begin the installation by clicking the Begin Installation button. If the user clicks Exit, the update is skipped until he or she opens the next document.

Configuring IDS to Update iDocumaker Workstation

To configure IDS to update iDocumaker Workstation, follow these steps:

1 Select the location where the iDocumaker Workstation executables will be kept under IDS. This example shows the default location if IDS is running from the \docserv directory.

```
Docserv\data\CONFIG\wipedit
```

If you need to change this you can set the following INI option in the configuration-specific INI file:

```
< WIPEdit >
    ExecDir = d:\docserv\data\sampco\wipedit
```

Option	Description
ExecDir	This option tells the update utility (VERSUPD) where the executables are located.

2 Copy the executables for the iDocumaker Workstation plugin into this directory.

IDS keeps a file that contains version information for these executables. The contents of this file are included inside the DPW file. The presence of this file determines whether the plugin update process occurs. Here is the default path for this file:

```
Docserv\data\CONFIG\CONFIG.wipedit
```

If this is not an acceptable location you must set the following INI option in the CONFIG.INI file to the appropriate location:

```
< WIPEdit >
    VersionFile = d:\docserv\data\sampco
```

3 The update program needs to know the location of the installation file from the web server so you must set up the following INI options. To set up these options, you must know the web site address and the relative path to the installation file within the web site.

```
< INI2XML >
    DownloadURL        = localhost
    DownloadScript     = doc-prog/data/sampco/wipedit.dpi
    DownloadUserID     = (user ID)
    DownloadPassword   = (password)
```

Option	Description
DownloadURL	You can enter the web site address or a machine name on the network. For example, you could enter localhost, pd.docucorp.com, www.docucorp.com, or an IP address. The exact value is specific to your implementation. This option is similar to the PUTURL option which may already be in the INI2XML control group.
DownloadScript	This is the part of the URL which points to the location within the host for the installation file. It should contain the name of the installation file. This is similar to the SCRIPT option which usually points to the wipsave.asp or wipsave.jsp file for iDocumaker Workstation. The exact value is specific to your implementation.
DownloadUserID	Enter the user ID for authentication purposes. This entry may be encrypted.
DownloadPassword	Enter the password for authentication purposes. This entry may be encrypted.

Here is an example of how you can use the CRYRUW32 utility at a command prompt to encrypt the data:

```
C:\docserv1.8>cryruw32 password
Encrypted string (2XAUnkxUYlx7i5AnQ4m4E1m00)
```

4 Next, use the VERSUPD utility to build the installation file and version file.

Using the VERSUPD Utility

Use this utility to create the installation file and the version file. These files are created by the VERSUPD utility:

- A version file which contains XML entries for version number, patch level, and accumulated CRC for files without patches.
- An installation file which has all of the executables for iDocumaker Workstation compressed into one file. The executable is installed on client machines via the UPDWDT utility.

NOTE: The UPDWDT utility is typically executed by iDocumaker Workstation when needed. You do not have to run it.

Both the version file and the installation file need to be created where the IDS rules expect them to be. By default, the VERSUPD utility creates them in same location IDS expects to find them.

Program names

Windows	versupd.exe
UNIX	versupd

Syntax

```
versupd /config /ini /versionfile /installation /base /debug
```

Parameter	Description
/config	Enter the name of the configuration, such as SAMPCO.
/ini	(Optional) Enter the path to the configuration INI file used by IDS. Specifying an INI file helps you make sure IDS and the VERSUPD utility look for shared files in the same location.
/versionfile	(Optional) Enter the path to the version file. This overrides the entry in the INI file.
/installation	(Optional) Enter the path to the installation file. This overrides the entry in the INI file.
/base	(Optional) Enter the path to the executables for iDocumaker Workstation. By default, this utility looks for executables in the data\config\wipedit directory under the current directory. This overrides the entry in the INI file
/debug	(Optional) Include this option to send a list of each file included in the installation to stdout.

Here is an example:

```
versupd /config=SAMPCO
```

This command puts both files in the following path:

```
c:\docserv\Data\SAMPCO
```

Here is another example:

```
versupd /config=SAMPCO /versionfile=c:\docserv\VersionControl
    /installation=c:\docserv\WIPEditInstallation
```

This command creates these files:

File type	Name and path
Version	c:\docserv\VersionControl
Installation	c:\docserv\WIPEditInstallation

INI options

These INI options from the CONFIG.INI files are read by the VERSUPD utility:

```
< WIPEdit >
    ExecDir     =
    VersionFile =
```

Option	Description
ExecDir	Enter the location for iDocumaker Workstation's executables.
VersionFile	Enter the location of the file that contains the version information.

Error messages

The following error messages may be generated by versupd. These errors will go both to stdout and to a file named *trace* that will be in the current directory of the VERSUPD utility.

```
Could not create installation file (version file path)
Could not find files to build installation in directory (iDocuMaker
Workstation directory)
Not able to add file (executable name) to installation (installation
file name)
Could not access directory where the iDocuMaker Workstation
executables are suppose to be.
Could not access directory where custom wipedit executables are
suppose to be (custom executable directory)
Unable to create version file (installation file)
Unable to retrieve version information from directory (iDocumaker
Workstation directory)
Unable to create document (version file)
Could not lock version file (version file name).
```

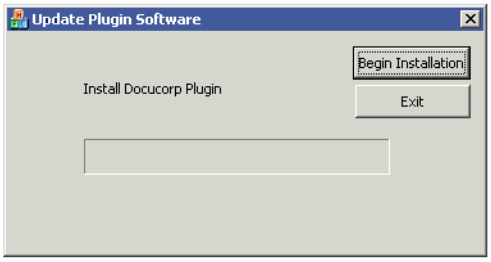
On the Client Side

When iDocumaker workstation is installed the version information is stored in registry. When iDocumaker Workstation parses the DPW file, it finds the most current version according to IDS. It then compares the version stored under IDS and the local version. If there is a discrepancy, iDocumaker Workstation's installation tool starts.

- The installation tool performs these steps:
- Downloads the archived file of iDocumaker Workstation executables.
- Backs up the current iDocumaker Workstation executable directory.

- Makes sure you have write access to all of the program files for iDocumaker Workstation.
- Erases all of the files in iDocumaker Workstation's executable directory.
- Installs the new executables.
- Updates the local version information in the registry.

Once you have set up IDS to automatically update iDocumaker Workstation, those computers with version 11.2 or higher of iDocumaker Workstation display the following window when you open a document.



Click the Begin Installation button to update iDocumaker Workstation. Click Exit to skip the update. The next time the user tries to open a document, IDS will again prompt the user to update iDocumaker Workstation.

New Components and APIs

This feature includes these new components and APIs:

- The VERS2REG utility gets the local version information and updates the registry. This utility executes on the workstation side from a command prompt. Typically, it is only executed during the original installation of iDocumaker Workstation. You can, however, start it from a command prompt. You can use the /v and /i parameters to determine which files and patch levels are in the current installation.

  vers2reg /v /i /p /r

Parameter	Description
/v	This parameter writes to stdout the values it will put in the registry.
/i	This parameter tells the utility to simply display the patch and CRC information and not change the registry.
/p	Use this parameter to set the path to the iDocumaker Workstation executables instead of using the registry to find the installed location
/r	This parameter indicates the registry key where the utility can find iDocumaker Workstation executables.

NOTE: CRC (Cyclic Redundancy Check) is a way to check for data transmission errors.

- The UPDWDT utility gets the iDocumaker Workstation installation file from the IDS and then updates iDocumaker Workstation.

NOTE: The UPDWDT utility is typically executed by iDocumaker Workstation when needed. Information about running this utility is only included in case you are having problems with the iDocumaker Workstation and need to update your system.

updwdt /b /i /r

Parameter	Description
/b	This tells the utility to copy the rebootbackup directory contents to the installation directory. This option is used when there are files to update during the reboot process.
/i	This tells the utility to install from a file specified by the next parameter. This is used if the installation file is already present on the local machine.
/r	The registry location where iDocumaker Workstation has been installed. The defaults is: `HKEY_CLASSES_ROOT\\wipedit.Document\\protocol\\StdFileEditing\\server`

- Documaker bridge rules to get the version information and CRC from the iDocumaker Workstation executables. You should have a separate location for each CONFIG value for these executables.

Checking Version Information

You can use the new WDTValidateDPI API to check the version information for iDocumaker Workstation from a menu. Place this API function in the WIPEDIT.RES file. When you use this function, these tests are performed:

1. Make sure local version information has been created. This identifies whether the VERS2REG utility has been run during the install process. If the version information does not exist, this message appears:

   ```
   Local version information does not exist for plug-in
   ```

2. Make sure the server version information has been updated. This indicates that server information was created and downloaded in the DPW file. If the version information cannot be found, this message appears:

   ```
   Version information does not exist on the server for plug-in
   ```

3. Compare server side information with local version information. If the version information matches, this message appears:

   ```
   You are running the correct version of the plug-in
   ```

   If the versions do not match, this message appears:

   ```
   Incorrect version of the plug-in - please update
   ```

4 Make sure the compressed file can be downloaded from the web server based on the download information in the registry. If it cannot, this message appears:

```
Not able to locate the installation file on the web server - (followed by web address attempted)
```

5 Check the format of the installation file. If there is a problem, this message appears:

```
Plug-in installation file is corrupt contact server administrator
```

If everything is Ok, you will see at least two messages. If tests 1, 2, and 3 pass the following message appears.

```
You are running the correct version of the plug-in
```

If tests 1, 2, and 3 fail but tests 4 and 5 pass, this message appears:

```
Installation file can be accessed successfully
```

1736 IDS

Running Multiple Instances of the WIP Edit Plug-in

You can now have multiple browser windows open, all running the WIP Edit plug-in. Note, however, that if the web application is not designed for multiple browser access by the same user, you will still experience problems. This feature only affects the WIP Edit plug-in.

You can get related debugging information when running multiple instances of the WIP Edit plug-in by setting this environment variable:

```
WIPCTLDEBUG=Y
```

The debugging information is placed in the wipctl.log file in the system's TMP directory or the directory specified by WIPEDITTMP.

Note: While this is not an issue for iDocumaker Workstation, some implementations in which iDocumaker Workstation is integrated with other web applications may be affected by this problem.

1844 IDS

Using IDS TPD Rules to Print Multi-page TIFF Files into PDF Files

Originally, the TPD rules could only print single page TIFF files into a PDF file. The system embedded CCITT Group 4 single strip TIFF files into the PDF file for performance reasons and stored other types of compressed and uncompressed TIFF file data directly into the PDF file.

This feature lets you process multi-page CCITT Group 4 single strip TIFF files and other types of multi-page TIFF files. So now the system can print single page, multi-page, and a combination of single and multi-page TIFF files into a PDF file, including color TIFF, dual resolution TIFF, and 32-bit TIFF files.

The TPD rules have also been enhanced to submit a combination of TIF, BMP, and JPG bitmap files in one request by specifying their types. The input attachment variables NAME and TYPE are sent to IDS with a full file name and type for each bitmap. If the bitmap file name is sent to IDS without the bitmap type, the TPD rule checks for the source type attachment variable SRCTYPE. If this variable does not exist, TIF is used as the default type.

Here is an example of the request type set up:

```
[ReqType:INI]
    function = tpdw32->TPDInitRule
[ReqType:TPD]
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = tpdw32->TPDCreateFormset
    function = tpdw32->TPDPrintFormset
```

Here are the input attachment variables:

Variable	Description
CONFIG	This is the name of the configuration, such as *SAMPCO*.
TIFFNAME	This is the number of input bitmap files.
SRCTYPE	This is the source bitmap file type, such as *TIF*.
TIFFNAME1.NAME	This is the name and path of the first bitmap file, such as *d:\docserv\mstrres\sampco\tif1.tif*.
TIFFNAME1.TYPE	The type of the first bitmap file.
TIFFNAME2.NAME	This is the name and path of the second bitmap file.
TIFFNAME2.TYPE	The type of the second bitmap file.
TIFFNAME3.NAME	This is the name and path of the third bitmap file.
TIFFNAME3.TYPE	The type of the third bitmap file.

NOTE: You can have as many TIFFNAME#.NAME/TIFFNAME#.TYPE variables as necessary.

1860
IDS

CHECKING WIP RECORDS THAT ARE NOT ON THE USER LIST

Now you can create a list of orphan WIP records. Orphan records are records that are not assigned to a valid user in the user database. Previously, you could only get a WIP list based on the Report To User assignments, which did not show orphan records.

Prior to this enhancement, when the system got the user list from the USERINFO file or from the IDS input attachment variable USERLIST, it compared the user ID in the record against the WIP user list and only output the records if there was a match.

This feature changes these rules so that if this input attachment variable:

```
CURRUSER = ~UNKNOWN~
```

is found, the following rules now perform as described in the table below:

Rule	Description of change
DPRGetWipList	If you specify this input attachment variable: `CURRUSER=~UNKNOWN~` this rule now lists the records that do not belong to users found in the valid user list.
DPRFindWipRecords (DPRSearchWip)	If you specify this input attachment variable: `CURRUSER=~UNKNOWN~` this rule now searches for records that do not belong to users found in the valid user list. Do not use field names such as RECORDID as the search criteria if you want to list the unknown user WIP records. This rule still checks the input attachment variable USEREPORTTOLIST as before and it has no effect if you specify *CURRUSER=~UNKNOWN~*.
DPRFindWipRecordsByUser	This rule does not support USEREPORTTOLIST and paging. If, however, you specify: `CURRUSER=~UNKNOWN~` the system generates the same unknown user WIP list as does the DPRFindWipRecords rule.
DPRCheckWipRecords	This rule searches the WIP records without checking the valid user list. If, however, you specify: `CURRUSER=~UNKNOWN~` the system checks the user IDs against the valid user list and reorders the list of unknown users.

NOTE: If *CURRUSER = ~UNKNOWN~* is not found, these rules work as before.

Here are some examples:

```
[ ReqType:WLT]
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRGetWipList

[ ReqType:WFD]
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRFindWipRecords

[ ReqType:WBU]
```

```
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRFindWipRecordsByUser

[ ReqType:WCR]
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRCheckWipRecords
```

1926 RPS

Meeting the PDF for Archive Specification

The PDF Print Driver now complies with the requirements of the PDF for Archive (PDF/A) specification. This specification was designed by the International Standards Organization (ISO) and is intended to facilitate the long-term storage of electronic documents. The specification (ISO 19005-1) outlines the restrictions to which conforming files must adhere.

There are two sets of requirements. The PDF/A-1a set of requirements is more rigorous than the PDF/A-1b set. The Docucorp PDF Print Driver implements PDF/A-1b compliance.

To set up your system to generate PDF/A-1b compliant documents, add the following INI option:

```
< PrtType:PDF >
    PDFAOptions = PDFA
```

Option	Description
PDFAOptions	Include this option if you want to produce PDF/A-1b compliant PDF files. Your entry identifies the INI control group that contains PDF/A-specific options. The default is PDFA.

If you use the default, the system automatically assumes the PDFA control group has the following options and values, so you do not have to actually include it — unless you want to use different values:

```
< PDFA >
    ICCProfile         = srgb.icc
    OutputCondition    = Generic CRT Monitor
    OutputConditionID  = sRGB IEC61966-2.1
```

Option	Description
ICCProfile	This specifies the name of the International Color Consortium (ICC) profile. This file describes the color attributes of a device or viewing requirement by defining a mapping between the source or target color space and a profile connection space (PCS). The file you specify must reside in the directory defined by the DefLib option in the MasterResource control group. The default is srg.icc. This file is embedded in the source code and will be used unless you include this control group and option and enter a different value in this field.
OutputCondition	A text string that describes the intended output device or production condition in human-readable form.
OutputConditionID	A text string identifier for the output condition.

NOTE: The PDFAOptions option tells the system you want to produce PDF/A-1b compliant PDF files. The system looks at the value you enter for that option and tries to find a control group by that name. If it cannot find a control group that matches, it uses the default PDFA values.

Keep in mind...

- The PDF file header must begin at byte offset zero (0) in the file. This is not a concern unless DAL scripts are used to place comments at the start of the file. If you enable PDF/A in a system that does emit comments at the beginning of the PDF file, the PDF driver will issue a warning but will still emit the comments.
- The document cannot be encrypted. If encryption and PDF/A support are both enabled, a warning appears and the document will not be encrypted.
- Colors must be specified in a device-independent manner. Currently, the Docucorp system supports the RGB scheme of specifying colors, which is device-dependent.

  To implement device-independence, the PDF driver must embed an ICC profile in the file. ICC profiles are widely available for an extensive array of devices. There may be licensing requirements if color profiles are to be embedded in PDF files. Adobe offers a royalty-free license for embedding the color profiles located at:

  www.adobe.com/digitalimag/adobergb.html

  The embedded color profile must be an RGB-based profile. The Docucorp PDF Print Driver defaults to using a file called srgb.icc, which is a generic color profile with a wide range of colors (gamut).

NOTE: ICC profiles vary in size from a few hundred bytes to a megabyte or more. If the system is configured to produce compressed PDF files, these are compressed as well. Be aware, however, that they add to the size of the file.

- To be PDF/A compliant, all fonts, TrueType or Postscript, must be embedded into the PDF file. This means the font files for every font used must be available on disk. Further, this requirement demands that all fonts used in a document be legally embeddable. It is up to you to handle any font licensing issues that may arise.

  Keep in mind that embedding fonts increases the size of the PDF file. Enabling font compression in the PDF driver will minimize this to an extent.

- The document catalog must contain a metadata key referencing a valid xmp metadata stream. The metadata stream must contain 2-4 KB of padding at the end to allow for in-place updating by applications that may not be PDF-aware. The Docucorp driver uses 3KB. Additionally, the metadata stream must be unfiltered — specifically, uncompressed. Therefore, metadata will increase the size of PDF files by a minimum of 3kB.

NOTE: Metadata is generated by the driver and is not configurable.

Important Notice

1. Docucorp's software may include an ICC Profile for use in producing PDF/A-compliant output. Docucorp obtained the ICC Profile from Adobe Systems Incorporated ("Adobe"), but this ICC Profile and others can also be obtained directly from Adobe's Support Web site and then used with the Docucorp software.

2. Docucorp has not modified the Adobe ICC Profile, however, Customer acknowledges that (a) Adobe disclaims all warranties and conditions, express or implied; (b) Adobe excludes any and all liability for damages related to the use of the ICC Profile as provided by Docucorp; and (c) the terms and conditions with respect to the license granted herein for the Adobe ICC Profile are offered by Docucorp alone and not by Adobe. The ICC Profile is Copyrighted 2006 by Adobe Systems Incorporated.

2069
IDS

USING MANUALLY-EDITED HTML FORMS WITH REAL-TIME HTML PROCESSING

This feature lets Docupresentment's Documaker Bridge return manually-edited HTML forms instead of performing a real-time conversion of FAP to HTML. It does not affect all FAP files, only the FAP files you would like to handle this way.

This feature is useful when you have FAP files that are using DAL scripts and similar logic is needed on HTML forms. If the FAP files do not change, you can convert specific FAP files into HTML manually, edit the HTML files, write Java scripts and so on, and have IDS return the HTML files instead of doing a real-time conversion of FAP to HTML.

Use the HTMLForms option in the CONFIG.INI file to specify the directory where the HTML files are located:

```
< MasterResource >
    HTMLForms =
```

Option	Description

HTMLForms	Enter the directory and path where the HTML forms reside. Documaker Bridge checks this directory for *filename*.htm and *filename*.html before deciding to convert FAP files into HTML files.
	For multi-page FAP files, each page has to be in a separate file. This naming convention is used:
	`filename_pagenumber.htm`
	For example, *myfile_2.htm* indicates the second page of multi-page FAP file called *myfile.fap*.
	If you need version/revision numbers on the HTML files, use the naming convention Studio uses for FAP files checked out of the library:
	`filename_versionrevision_effdate.htm`
	Here is an example:
	`CANC201B_0000300005_20060101.htm`
	This references FAP file *CANC201B* version 3, revision 5, with an effective date of 1/1/2006. If you need to add a page number to denote the second page, do so at the end, as shown here:
	`CANC201B_0000300005_19800101_2.htm`
	The system first checks for the file name with version, revision, and effective date information. If not found, it then checks for just the file name. Each check is done for both the *HTM* and *HTML* extensions.
	If image does not have version/revision information the check for file name with version/revision is omitted.

NOTE: While it is possible, it is not recommended to use this feature for all FAP files in your library as it will increase the amount of maintenance you must perform.

Use this option in the CONFIG.INI file to help resolve problems:

```
< Debug >
    DPRGetHTMLForms = Yes
```

Option	Description
DPRGetHTMLForms	Enter Yes to create the log file with information about which file names were checked and which files were found.

2070 IDS

ACCESSING IDS ATTACHMENT VARIABLES IN GENDATA

There are times when the GenData program needs access to data passed from IDS which is not in the extract file. This feature lets the GenData program access IDS attachment variables as GVM variables. If a GVM variable with the same name already exists, its value does not change.

Here is how it works:

On the IDS side

The RPDCreateJob rule adds any input attachment variables to the XML tree (job ticket) besides the existing variables, such as MsgFile, ErrFile, ExtrFile, LogFile, DbLogFile, NaFile, PolFile, NewTrn, PrtLog, PrtType, ExtrPath, PrintPath, PrintBatches, BatchFiles, IniOptions, EWPSRequest, EWPSResults, ShowErrors, WIPRECORDID, XMLOUTPUT, and so on.

For example, if an attachment variable called RPDTEST is located and it has a value of *This is a test*, it is added to the XML tree as shown here:

```
<DOCUMENT>
<JOBTICKET>
. . .
<RPDTEST>This is a test</RPDTEST>
</JOBTICKET>
</DOCUMENT>
```

On the Documaker Server side

After the ServerJobProc rule receives the XML tree (job ticket), its child elements are used to update INI values or create GVM variables or both.

2075 IDS

Adding Form Line and Form Description Line Information

You can use the new DSAddFormLines function with IDS import rules to add field Form Line and Form Desc Line information. This enhancement affects Documaker Workstation and the following rules:

- DPRUpdateFormsetFromXML rule (IDS)
- DPRLoadImportFile rule (IDS)
- RULImportXMLExtract rule (Documaker Server)
- RULImportXMLFile rule (Documaker Server)

Use these INI options to enable this feature:

```
< Control>
    DoFormLines    = Yes
    DoFormDescLines=Yes
```

Option	Description
DoFormLines	Enter Yes to include form lines. The default is Yes.
DoFormDescLines	Enter Yes to include form description lines. The default is Yes.

Use these INI options to control how the information is presented:

```
< FormDescTable >
    BoldKey2          =
    ColumnFormat      =
    IncludeKey2       =
    IncludeKey2Name   =
    Key2Prefix        =
    Key2PreInc        =
```

```
Key2PostInc        =
IncludeFormName    =
IncludeDuplicateForms=
ExcludedGroup      =
ExcludedForm       =
```

Option	Description
BoldKey2	Use this option to present Key2 descriptions in a bold font. The system determines which font to use by querying the font defined on the field and selecting its bold equivalent. The fonts of normal Form Description Lines fields (not assigned a Key2 name) will be changed to their non-bold counterparts. If you enable this feature, the system will query the font associated with each Form Description Line field and request either the bold or non-bold equivalent from the same font family and size. If the requested font is not available, the system does not change the field's font. To choose the bold and non-bold equivalent of a font, the system uses your FXR file, which must be defined properly. Each font listed in the FXR file has a stroke weight assigned to it. The stroke weight indicates the boldness of the font.
ColumnFormat	Set this option to No to have the system append the form description to the end of the form name, separated by two spaces. The default is Yes, which formats the form lines in a columnar fashion.
IncludeKey2	Use this option to enable or disable Key2 descriptions. To enable Key2 descriptions, set this option to *Yes*. The default is *No*.
IncludeKey2Name	Like the previous option, you can use this option to tell the system whether or not to include the Key2 group name obtained from the table in the returned description. The default is No. When you use the table feature, the system ignores the Key2Prefix option mentioned previously. The text for Key2 must appear in the table as you want it to appear on the form.If you include the Key2 name in the description text, the system pads the name with spaces up to the maximum length of a form name. This helps make sure the Form Description Lines appear as columns of data if you use a fixed pitch or non-proportional font (like Courier) to define the fields. If you do not want Key2 names included on the description line, set this option to Yes.

Key2Prefix	Use this option to specify a text string which will appear before each Key2 description line. The system automatically appends a single space after the text string. By default, this option is blank and does not affect the description lines. Here is an example of how you can use this option:
	`Key2Prefix = Forms Applicable -`
	By setting the option as shown above, the system prefixes all Key2 descriptions with the specified text. For instance, the output might look like this:
	`Forms Applicable - General Liability Coverage`

Option	Description

Key2PreInc
Key2PostInc
	Use these options to add blank lines between the Key2 descriptions and the form descriptions. If you set both of these options to one (1), your output might look like this:
	`Forms Applicable - COMMON POLICY`
	`DEC PAGE Common Policy Declarations`
	`FIL 1010 04 92 Supplemental Declarations`
	`Forms Applicable GENERAL LIABILITY`
	`CG DEC General Liability Declarations`
	If you include Key2 descriptions, the first text will always represent the first form grouping. This first Key2 description will not use the Key2PreInc option to include blank lines before the text. Subsequent groups, however, will have the specified number of blank lines before their text descriptions.

IncludeFormName	When you include form description lines on a form, the system typically returns both the form name and description. The IncludeFormName option lets you omit the form name and only print the description for each form.The default is Yes. This tells the system to include the form name and description, as shown in this example: `DEC PAGE Policy declarations for all coverage` The form's name (*DEC PAGE*) is included in the first portion of the string. The system pads the name with extra spaces equivalent to the maximum length of the form name to make sure the form description lines appear as columns of data if you use fixed pitch or non-proportional font (like Courier) to define the fields. To omit form names on the description line, set this option to No. Although this option is in the FormDescTable control group, you can use this option even when you do not use an associated table file to provide longer descriptions than those provided in the FORM.DAT.
IncludeDuplicateForms	When you use the Formset, Duplicate Form option to duplicate a form, the system excludes the duplicate forms from the form description lines. If you want the system to include the duplicate forms, change this option to Yes.
ExcludedGroup	Enter the name of the groups (as defined in the Key2 field) you want to exclude.
ExcludedForm	Enter the name of the forms you want to exclude.

2078 IDS — Setting the Starting Record for the

DPRUpdateFromMRL Rule

You can now specify the starting record for the DPRUpdateFromMRL rule to use when returning a form list. If you include the new STARTRECORD attachment variable, it will override the starting record set by PAGE attachment variable.

Variable	Description
STARTRECORD	Enter the record you with which you want the rule to start.

2081 IDS

Passing a WIP Record ID to the MergeWIP Rule

This enhancement lets iDocumaker Workstation designate a single WIP transaction to be processed in Documaker Server. IDS then passes the WIP record ID to Documaker Server so the MergeWIP rule will process that record.

This table describes how it works:

On the...	This happens
IDS side	The RPDCreateJob rule checks the WIPRECORDID input attachment variable and adds the XML element ⟨WIPRECORDID⟩ to the job ticket. Keep in mind that the WIPRECORDID input attachment variable is required when the RPD request is submitted. This requirement is in addition to the normal requirements for running Documaker Server as a subordinate process of IDS.
Documaker Server side	The ServerJobProc rule receives the job ticket and looks for the WIPRECORDID variable. If WIPRECORDID is found, the rule creates the WIPRECORDID GVM. The MergeWIP rule uses the WIPRECORDID GVM to retrieve the WIP record for batch processing by Documaker Server.

Note: This enhancement only affects Documaker Server when you are running Documaker Server via IDS.

For more information about the MergeWIP rule, see the Rules Reference. For more information about running Documaker Server as a subordinate process of IDS, see the Documaker Server Reference Guide.

2082 IDS

Using the New DPRGetConfigList Rule

Use this rule to get a listing of the configuration information in the DAP.INI file.

Syntax

```
long  _DSIAPI DPRGetConfigList(DSIHANDLE hdsi,
```

```
                                        char * pszParms,
                                        ULONG ulMsg,
                                        ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hInstance	DSI instance handle
char * pszParms	pointer to rule parameter string
ULONG ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
ULONG ulOptions	options

Input attachments

None

Output attachments

Attachment	Description
RESULTS	Success or failure.
CONFIGLIST	List of configuration information from the DAP.INI file.

Errors

Error	Description
DPR0013	The initialization file, FILENAME could not be loaded.

Example

Here is the request type for docserv.xml:

```
<section name="ReqType:ewps_doGetLibraries">
    <entry name="function">atcw32-&gt;ATCLogTransaction</entry>
    <entry name="function">atcw32-&gt;ATCLoadAttachment</entry>
    <entry name="function">atcw32-&gt;ATCUnloadAttachment</entry>
    <entry name="function">dprw32-&gt;DPRGetConfigList</entry>
</section>
```

Here is the request type for the docserv.ini file:

```
[ ReqType:ewps_doGetLibraries ]
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRGetConfigList
```

Here is an example of returned attachment variables:

```
RESULTS                 SUCCESS
SERVERTIMESPENT         0.010
CONFIGLIST              11
CONFIGLIST1.CONFIG      AFP2PDF
CONFIGLIST2.CONFIG      amergen
CONFIGLIST3.CONFIG      DOCUMERGE
CONFIGLIST4.CONFIG      EBPPTEST
```

```
CONFIGLIST5.CONFIG     FINANCE
CONFIGLIST6.CONFIG     INSURE
CONFIGLIST7.CONFIG     PPDemo
CONFIGLIST8.CONFIG     RPEX1
CONFIGLIST9.CONFIG     sampco
CONFIGLIST10.CONFIG    TIFF2PDF
CONFIGLIST11.CONFIG    UTILITY
```

2090 IDS

Printer Support for WIP Edit

This feature lets you set up your print selections from the WIP Edit plug-in. WIP Edit gets the needed fonts from Docupresentment automatically, using the GETRESOURCE request. This helps insure better fidelity of printed copies by using PCL or PostScript printers.

To use this feature, make sure...

- The WIPEDIT.RES file includes the print option.
- The WIPEDIT.INI file has the print types set up in the same manner as those in Documaker Workstation or PPS.

NOTE: The INITOKEN option in the File2Dpw control group of the CONFIG.INI file must be set before options in the WIPEDIT.INI file can take effect.

- The GETRESOURCE type is in the docserv.xml or DOCSERV.INI file.
- The FontLib option in the MasterResource control group is set in the CONFIG.INI file.

Here is an example of the WIPEDIT.RES file:

```
POPUP "&Print" 1070 "Print"
BEGIN
MENUITEM "Pri&nt Formset..." 1065 "NULL" "NULL"
MENUITEM "&Form..." 1066 "NULL" "NULL"
MENUITEM "Pa&ge..." 1067 "NULL" "NULL"
END
```

Here are examples for the WIPEDIT.INI file:

```
:PostScript examples.
< PrtType:PST >
    DownloadFonts = Yes,Enabled
    Module = PSTW32
    PrintFunc = PSTPrint
    Resolution = 300
;   SendOverlays = Yes,Enabled
    SendOverlays = No,Disabled
< PrtType:PXL >
    DownloadFonts = Yes,Enabled
    Module = PXLW32
    PageNumbers = Yes
```

```
    PrintFunc = PXLPrint
    SendOverlays = No,Enabled
< PXL >
    Device = \\Atl1dc01\YEL_HP8000_A
    DownloadFonts = Yes
    Module = PXLW32
    PrintFunc = PXLPrint
< PrtType:PCL >
    DownloadFonts = Yes,Enabled
    Module = PCLW32
    MultipleCopies = Yes
    PrintFunc = PCLPrint
    SendOverlays = No,Enabled
```

Here is an example the docserv.xml file:

```
<section name="ReqType:GETRESOURCE">
<entry name="function">atcw32->ATCLogTransaction</entry>
<entry name="function">atcw32->ATCLoadAttachment</entry>
<entry name="function">atcw32->ATCUnloadAttachment</entry>
<entry name="function">dprw32->DPRSetConfig</entry>
<!-- entry name="function">dprw32->DPRDecryptLogin</entry -->
<!-- entry name="function">dprw32->DPRDefaultLogin</entry -->
<!-- entry name="function">dprw32->DPRCheckLogin</entry -->
<entry name="function">atcw32-
>ATCSendFile,RETURNFILE,RETURNFILE,Binary</entry>
<entry name="function"<dprw32->DPRGetResource,RETURNFILE</entry>
<!-- -->
</section>
```

Here is an example of the DOCSERV.INI file:

```
[ReqType:GETRESOURCE]
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSetConfig
function = dprw32->DPRDecryptLogin
function = dprw32->DPRDefaultLogin
function = dprw32->DPRCheckLogin
function = atcw32->ATCSendFile,RETURNFILE,RETURNFILE,Binary
function = dprw32->DPRGetResource,RETURNFILE
```

Here is an example of how you would set the FontLib option:

```
< MasterResource >
    FontLib = mstrres\sampco\fmres
```

For more information on setting printing options, see the Documaker Server Reference Guide or the Documaker Workstation Supervisor Guide.

2092 IDS

Using WIP Edit with SiteMinder▸

This feature lets the WIP Edit plug-in work with web sites protected by SiteMinder• and allows WIP Edit to work with web sites that use clustered web servers.

2093 IDS

Customizing the Execution of Documaker Server under Docupresentment

This feature affects running multi-step Documaker Server as a subordinate process to Docupresentment. You can now introduce special steps which occur between the GenTrn, GenData, and GenPrint steps. You can use these steps, for instance, to

- Sort the TRNFILE or recipient batches
- Copy files to different locations
- Send files to the printer
- Notify an operator that steps were completed

INI options

This feature modifies the RPDRunRP rule so it can run a custom executable after each step in the process. Use these INI options to define the custom executable name and path:

```
< RPRun >
    PostGenTrnExecutable   =
    PostGenDataExecutable  =
    PostGenPrintExecutable =
```

Option	Description
PostGenTrnExecutable	Enter the name and path of the custom executable, such as a batch file or shell script, you want the system to run after it completes the GenTrn step.
PostGenDataExecutable	Enter the name and path of the custom executable, such as a batch file or shell script, you want the system to run after it completes the GenData step.
PostGenPrintExecutable	Enter the name and path of the custom executable, such as a batch file or shell script, you want the system to run after it completes the GenPrint step.

Parameters

By default, the following information is passed as parameters to each executable:

```
GenTrn - INI file, trnfile
GenData - INIfile, recipient batches
GenPrint - INI file, print file
```

These INI options are read from the FSIUSER.INI file to create the parameter list. The FSIUSER.INI file is created by the RPDRunRP rule. This INI file is passed to Documaker Server for each step. Internally, the RPDRunRP rule loads the FSIUSER.INI file and gets parameter information from it.

GenTrn parameters:

```
< Data >
    TrnFile = (parameter trnfile)
```

GenData parameters:

The system reads all of the options under the Print_Batches control group to determine the recipient batches:

```
< Print_Batches >
    Batch# = (paramter receipient batches )
```

GenPrint parameters:

The system reads all of the print batches to determine the printer used and passes the port value for the printer as the parameter.

```
< Print_Batches >
    Batch1 = (value ignored)
< Batch1 >
    Printer = Printer1
< Printer1 >
    Port = (parameter print file)
```

Error messages

If any of the INI options are missing, the system logs an error. It will, however, try to run the post process without the missing parameter. Memory and list allocation errors result in failure and the system will not attempt to execute the outside process.

Here is a list of the potential errors:

```
Could not get INI option <Data> TrnFile GenTrn step (non-fatal)
Could not create VMM list GenTrn step (fatal)
Could not load INI file GenTrn step (fatal)
Could not get INI context GenTrn step (fatal)
Could not create VMM list GenData step (fatal)
Could not load INI file GenData step (fatal)
Could not get INI context GenData step (fatal)
Could not create VMM list GenPrint step (fatal)
Could not load INI file GenPrint step (fatal)
Could not get INI context GenPrint step (fatal)
Could not get INI option GenData step <group> <option> (non fatal)
Could not start process:  [executable name and command line] - fatal
Memory re-allocation failed (fatal)
Memory allocation failed (fatal)
PROCStartProcess failed: (command line)
PROCWaitProcess failed: [executable name and command line] - non
fatal
PROCExitCodeProcess failed: [executable name and command line] - non
fatal
```

2100 IDS

Using the DPRUpdateFormsetFromXML Rule with HTML Entry

The DPRUpdateFormsetFromXML rule was originally designed for use only with the WIP Edit plug-in. This enhancement lets you use it with HTML entry as well. Now when this rule is used with HTML entry it acts just like the DPRLoadImportFile rule.

The DPRUpdateFormsetFromXML rule now uses the attachment variable DPRIFORMSPROTOCOL to determine if only forms are changed or if image and field information is affected as well.

- When the value of DPRIFORMSPROTOCOL is blank, missing, or *PLUGIN*, only form information is updated, so the rule can add, remove, and change order of forms. Image and field information is ignored.
- If the DPRIFORMSPROTOCOL value is something else, like *IDS* or *RDBMS*, this rule acts similar to the DPRLoadImportFile rule when importing XML files. The form set is replaced with the information in the XML file, including forms, images, fields, and so on.

iPPS was enhanced to provide the DPRIFORMSPROTOCOL value automatically so you do not have to do anything to make this work.

NOTE: For more information on these rules, see the SDK Reference.

2101 IDS

DESIGNATING READ-ONLY MULTI-LINE TEXT FIELD PARAGRAPHS

The following attributes are now included in the XML export file on the ‹P› tag for read-only multi-line text field paragraphs:

```
contenteditable="false"
unselectable="on"
```

These attributes are used by iPPS and iDocumaker Workstation TERSUB functionality to prevent a user from selecting or modifying the paragraphs.

Here is an example:

```
<P contenteditable="false" unselectable="yes" />
```

2102 IDS

AUTOMATICALLY DETERMINING THE PRINTBATCHES VALUE

You no longer need to set the PRINTBATCHES attachment variable. If it is missing when the Documaker Bridge rules are executed, the rules will now find the Documaker INI files and determine the correct value for PRINTBATCHES.

To use this feature, simply stop passing the PRINTBATCHES attachment variable to Docupresentment.

The system determines the PRINTBATCHES value based on the number of Printer options in the PrinterInfo control group of your Documaker Server (GenData) INI files (FSIUSER.INI and FSISYS.INI):

```
< PrinterInfo >
    Printer =
```

Since these INI options exist, you do not have to make any changes.

NOTE: If the PRINTBATCHES value is provided, the rules work as before, so existing implementations are not affected by this change.

2104 IDS

USING ENHANCED DAL FUNCTIONS FOR WIP COLUMN ACCESS WITH DOCUPRESENTMENT

You can now use the following DAL functions to set or retrieve WIP field data when Docupresentment processes WIP or archived transactions:

Function	Enhancement
WIPKEY1	Returns the value of the Key1 WIP field. Requires no input parameters. Here is an example: `Val=WIPKEY1()`
WIPKEY2	Returns the value of the Key2 WIP field. Requires no input parameters. Here is an example: `Val=WIPKEY2()`
WIPKEYID	Returns the value of the KeyID WIP field. Requires no input parameters. Here is an example: `Val=WIPKEYID()`
WIPFLD	Returns the value of the specified WIP field data. This field must be defined in the WIP.DFD file. Requires one input parameter to indicate which key value to return. Here is an example: `Val=WIPFLD("TranCode")`

SETWIPFLD	Sets the value of the specified WIP field key and keeps it in memory until the job finishes. For example, this DAL script sets/changes CURRUSER to DEMO1 and returns it:
	`SETWIPFLD("CURRUSER","DEMO1");`
	`Val=WIPFLD("CURRUSER");`
	`Return Val;`

There are several ways to run the DAL script, here are two examples:

- Using the ~DALRUN built-in INI function following a DAL script file, as shown in this example:

```
~DALRUN wipkey.dal
```

- Using the DPRExecuteDAL rule, as shown in the following request type:

```
[ ReqType:i_WipTest]
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
```

```
function = dprw32->DPRGetWipFormset
function = dprW32->DPRExecuteDAL,wipkey.dal,RUNF
```

You can also now use the following built-in INI functions to retrieve WIP field data when Docupresentment processes WIP or archived transactions:

- ~KEY1
- ~KEY2
- ~KEYID
- ~ORIGUSER
- ~CREATETIME
- ~MODIFYTIME
- ~FORMSETID
- ~ORIGFSID
- ~TRANCODE
- ~DESC
- ~WIPFIELD

All of these built-in functions except WIPFIELD do not require input parameters.

The WIPFLD function requires an input parameter that indicates the field data to return. Here is an example:

```
~WIPFLD FORMSETID
```

In this example, FORMSETID is the input parameter that specifies the field name.

2105 IDS

Generating File Names Based on Transaction Values

Docupresentment's Documaker Bridge rules (DPRPrint and DPRUnloadExportFile) have been enhanced so you can specify output names based on transaction data when Docupresentment processes WIP and archived transactions. This is done using INI options and built-in INI functions.

This enhancement lets you have full control over output file names and can be used, for example, when you need to interface to 3rd party system that requires specific file naming conventions.

NOTE: You must make sure the generated file names are unique. If you set up the system so that it generates the same name multiple times, the files are going to be overwritten. Use with caution.

Here is an example of how you can use a built-in INI function and DAL function to specify the output file while printing a transaction from WIP:

You need this request type:

```
< ReqType:i_WipPrint >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRGetWipFormset
    function = dprw32->DPRPrint
```

You need these input attachment variables:

Variable	Description
CONFIG	Configuration
RECORDID	A WIP record ID. Used to identify and retrieve a WIP record. The variable can also be RECNUM or Unique_ID, depending on the type of database.
PRTTYPE	XXX. If the INI option is not found, the system uses the default printer type of PDF.
PRINTPATH	The full path for the print file.
PRINTFILE	The output file name with or without a full path. If PRINTFILE includes a file name, a path, and a file extension, the system ignores the PrtType:XXX control group options (FileName, FileExt, and FileDir). If PRINTFILE does not include a path, the system checks PRINTPATH. If PRINTPATH does not exist, the system checks the FileDir option. If PRINTFILE does not include an extension, the system checks the FileExt option. If there is no entry for the FileExt option, the system defaults to the PRTTYPE for the extension. If both PRINTFILE is omitted and the PrtType:XXX control group options are omitted, the system creates a unique name.

You need these INI options:

```
< Printer >
    PrtType = XXX
< PrtType:XXX >
    FileName =
    FileExt =
    FileDir =
```

Option	Description
Printer control group	
PrtType	Optional. Specify the printer type.
PrtType:XXX control group	

Option	Description

FileName	Enter the output file name, with or without a full path. You can get the file name using built-in INI functions, as shown here: ~DALRUN, ~Key1, ~Key2, or ~KeyID, and so on.
	If you use the ~DALRUN built-in INI function to specify the file name, you can set the FileName option as shown here:
	`FileName = ~DALRUN wipkey.dal`
	Where *wipkey.dal* is the DAL script you want the ~DALRUN built-in INI function to execute. Here is an example of a DAL script:
	`Val=WIPKEY();`
	`return Val;`
	You can also use WIPKEY1(), WIPKEY2(), or WIPFLD("FieldName").
	If you use the ~Key1, ~Key2, or ~KeyID built-in INI function, you can set the FileName option as shown here:
	`FileName = ~KeyID`

FileExt	(Optional) Enter the appropriate file extension for the file type. The system uses your entry if it cannot find the PRTTYPE input attachment variable.
FileDir	Enter the name of the directory into which the output file should be placed. The system uses your entry if the FileName option does not include a full path and it cannot find the PRINTPATH input attachment variable.

Here is another example of how you can use a built-in INI function to specify the output file while exporting a transaction from WIP:

You need this request type:

```
< ReqType:i_WipExport >
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRGetWipFormset
function = dprw32->DPRUnloadExportFile
```

You need these input attachments:

Variable	Description
CONFIG	Configuration
RECORDID	A WIP record ID. Used to identify and retrieve a WIP record. The variable can also be RECNUM or Unique_ID, depending on the type of database.
FILETYPE	The file type. The default is V2.
EXPORT	The output file name, with or without a full path. If omitted, the system checks the INI options in the following control groups to determine the file type. - For V2, check the ExpFile_CD control group - For CMBNA, check the ImpExpCombined control group - For XML, check the XML_IMP_EXP control group If the INI options are omitted, the system creates a unique file name.

You need these INI options to export a V2 file:

```
< ExpFile_CD >
    File    =
    Ext     =
    Path    =
```

Option	Description

File	Enter the output file name, with or without a full path.
	If you use the ~DALRUN built-in INI function to specify the file name, you can set it as shown here:
	`File = ~DALRUN wipkey.dal`
	If you use the ~Key1, ~Key2, or ~KeyID built-in INI function to specify the file name, you can set it as shown here:
	`File = ~KeyID`

Ex t	(Optional) Enter the file extension. The default is *out*.
Path	Enter the path of the output file. This option is ignored if you enter a full path in the File option.

You need these INI options to export a CMBNA file:

```
< ImpExpCombined >
    File    =
    Ext     =
    Path    =
```

Option	Description

File	Enter the output file name, with or without a full path.
	If you use the ~DALRUN built-in INI function to specify the file name, you can set it as shown here:
	`File = ~DALRUN wipkey.dal`
	If you use the ~Key1, ~Key2, or ~KeyID built-in INI function to specify the file name, you can set it as shown here:
	`File = ~KeyID`

Ex t	(Optional) Enter the file extension. The default is *ds*.
Path	Enter the path of the output file. This option is ignored if you enter a full path in the File option.

You need these INI options to export an XML file:

```
< XML_IMP_EXP >
    File    =
    Ext     =
    Path    =
```

Option	Description

File	Enter the output file name, with or without a full path.
	If you use the ~DALRUN built-in INI function to specify the file name, you can set it as shown here:
	`File = ~DALRUN wipkey.dal`
	If you use the ~Key1, ~Key2, or ~KeyID built-in INI function to specify the file name, you can set it as shown here:
	`File = ~KeyID`

Ex t	(Optional) Enter the file extension. The default is *xml*.
Path	Enter the path of the output file. This option is ignored if you enter a full path in the File option.

2114 IDS

Automatic Detection of Import File Type by Documaker Bridge Rules

This enhancement lets you use the same request type and the same attachment variables to import all supported import file types into Documaker Server. To determine the import file type, the beginning of the input file is checked:

For this kind of import	The file should begin with
XML file	‹*?xml*
Combined NA/POL file	*WIP=*
V2 import	if not *WIP=* or ‹*?xml*, the system assumes the file is a V2 import format file

NOTE: This change affects DPRLoadImportFile rule and is only applicable if the FILETYPE attachment variable is blank or omitted. If this variable is passed in with a value of *XML* or *CMBNA,* that format is assumed and no automatic check occurs.

2115 IDS

Using the DPRWipBatchPrint Rule

Use this rule to print multiple transactions from WIP. This rule is used with iDocumaker Workstation or iPPS to produce non-PDF output when all transactions are output into one print-ready file. The print types are PCL, PCL6 (PXL), or PostScript.

Syntax

```
long _DSIAPI DPRWipBatchPrint ( DSIHANDLE hdsi,
                                char * pszParms,
```

```
                                    ULONG ulMsg,
                                    ULONG ulOptions )
```

Parameters

Parameter	Description
DSIHANDLE hdsi	pointer to the rule data
char * pszParms	pointer to rule parameter string
ULONG ulMsg	DSI message
ULONG ulOptions	options

Attachment inputs

This rule expects these input attachment variables:

Variable	Description
PrtType	Optional. This specifies the print type, such as PCL, PCL6 (PXL), or PostScript. If not present, the system checks the Printer control group. The default is PCL.
PrtDevice	Optional. This is the name of the print device.
PrintFile	Optional. The name of the print file. If PrtDevice present, this variable is ignored. By default, the system creates a 46-byte unique file name.
PrintPath	Optional. This path points to the location of the print file.
DPRProofLogo	Optional. Enter Yes if you want to include a logo. See the discussion of the DPRAddLogo rule for setup options.
RecordIDs	The number of records or a list of record IDs delimited by commas. Here is an example of how you can use RecordIDs to specify a list of record IDs: `RecordIDs 00000001,00000002,00000003, ...` You can also use this variable to specify the total number of record IDs and then list those IDs using RecordIDs*X*, as shown in the RecordIDs*X* discussion.

RecordIDs*X*	A record ID, where *X* denotes a record index from one (1) to the number of records. Include this variable if RecordIDs contains the total number of records.
	Here is an example of how you would specify the number of records (using RecordIDs) and the actual record IDs (using RecordIDs*X*):
	`RecordIDs 10`
	`RecordIDs1 00000001`
	`RecordIDs2 00000002`
	`...`
	`RecordIDs10 00000010`

Variable	Description
RecNums	The number of records or a list of record IDs. This variable is ignored if RecordIDs exists. Here is an example of how you can use RecNums to specify a list of record IDs: `RecNums 00000001,00000002,00000003...` You can also use this variable to specify the total number of record IDs and then list those IDs using RecNums*X*, as shown in the RecNums*X* discussion.

RecNums*X*	This is a record ID, where *X* denotes a record index from one (1) to the number of records. Include this variable if RecNums contains the total number of records.
	Here is an example of how you would specify the number of records (using RecNums) and the actual record IDs (using RecNums*X*):
	`RecNums 10`
	`RecNums1 00000001`
	`RecNums2 00000002`
	`...`
	`RecNums10 00000010`

AllRecipients	Optional. If present, all recipients copies are printed to the print file.
Recipient	Enter a list of recipients delimited by commas. Here is an example: `AGENT,COMPANY,INSURED` Recipient is ignored if you include AllRecipients.

NOTE: You can use either RecordIDs or RecNums, both accomplish the same purpose. Both are provided for your convenience.

Keep in mind that the values passed in via RecordIDs or RecNums are the record numbers if the WIP index is in xBase or the values in the UNIQUE_ID column if the WIP index is in an SQL database, depending on your setup.

INI options

You can use these INI options:

```
< Printer >
    PrtType    =
< Attachments >
    PrintPath  =
```

Option	Description
PrtType	Optional. This specifies the print type, such as PCL, PCL6 (PXL), or PostScript. If not present, the system checks the Printer control group. The system ignores this option if the input attachment variable PrtType is present. The default is PCL.
PrintPath	Optional. The name of the print device. The system ignores this option if input attachment variable PrintPath is present.

You may also need to set up INI options for WIP record retrieval and printers in the PrtType:XXX control group and also define recipients in the Recip_Names control group.

To reduce the number of PCL fonts being downloaded into the print stream, which optimizes the size of the output file, set these INI options:

```
< PrtType:PCL >
    InitFunc       = PCLInit
    TermFunc       = PCLTerm
    DownloadFonts  = Yes
```

This makes sure each font is downloaded only once and only when needed.

In addition, if you want to add a logo you can add the AddLogo control group to the master resource INI file. Here is an example of the INI options you could use:

```
< AddLogo >
  Logo  = TRSEAL
  Top   = 600
  Left  = 1200
  Pages = 1
  Color = 16711680
```

Option	Description
Logo	The name of the logo you want to use. Store this logo in the FORMS directory of the master resource library.
Top	Contains the top coordinate (position) of the logo in FAP units (2400 units per inch)
Left	Contains the left coordinate (position) of the logo in FAP units (2400 units per inch)
Pages	(Optional) The default is to add the logo on all pages. Use this option to set the number of pages on which you want the logo to appear. If you set this option to 1, the system adds a logo to the first page only.
Color	(Optional) Default is to display the logo as a black and white logo (value of zero). This number is a 24-bit RGB color. The lowest 8 bits represent the amount of red color, the next 8 bits represent the amount of green color, and the subsequent 8 bits represent the amount of blue color. A color setting of 255 (lowest 8 bits are all on) would indicate the full amount of red and no green or blue. A color setting of 65535 (lowest 16 bits are on) indicates the full amount of red and green but no amount of blue. This results in yellow.

Returns

Success or failure

Errors

Error	Message
DPR0001	Cannot locate the variable #VARIABLE,# in the attachment list.
DPR0022	Cannot add the variable #VARIABLE,# to the attachment list.

Error	Message
DPR0069	API #APINAME,# failed to print the form set at location #LOCAION,#.
DPR0084	Failed to get the current record #RECORDID,# in #LOCATION,#.
DPR0086	Failed to load the WIP form set.

Example

Here is an example request type:

```
[ReqType:i_WipBatchPrint]
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRWipBatchPrint
```

Here are some example input attachments:

```
CONFIG SAMPCO
USERID DOCUMAKER
PRTTYPE PCL
PRINTFILE TMP.PCL
PRINTPATH d:\docserv\mstrres\sampco
RECORDIDS 3
RECORDIDS1 1
RECORDIDS2 2
RECORDIDS3 3
ALLRECIPIENTS YES
```

2120
IDS

Enhanced Stability when Running Documaker under Docupresentment

Docupresentment now restarts Documaker Server (GenData) if it encounters a critical error and resubmits the transaction. This change was made to help counteract situations where you have sporadic memory access problems in custom or 3rd-party code.

Before this change when the GenData program started, the RPDProcessJob rule communicated with GenData via TCP/IP, sending the job ticket message to GenData and receiving a job log response. If the TCP/IP communication failed, the RPDProcessJob rule forced GenData to stop. This would prepare IDS for the next request.

This change adds the ability to automatically restart GenData after the process described above. After it confirms that GenData has been stopped, the RPDProcessJob rule calls the RPDCheckRPRun rule to restart GenData and then calls itself to communicate with GenData and send the same job ticket.

To keep a copy of each transaction, an eight-digit index number is added to the job ticket and job log file names when they are downloaded for information in debug modes.

Also, the system now includes these error messages which can appear if there is a TCP/IP failure:

Message	Description
RPD0011	Unexpected program termination of GenData.
RPD0012	Socket connection failure.
RPD0013	Can not unload job ticket to the msg buffer.
RPD0014	Can not load the msg buffer to the job log.
RPD0016	Socket time-out.
RPD0017	Time exceeded the MaxWaitForStart specification.
RPD0018	GenData failure.

The system now includes additional information in the log trace file in case of failure. This includes the job ticket, the input attachment variables, and error messages. On the IDS side, this file is named *dprtrc.log*. On the GenData side, this file is the trace file.

2124
IDS

Automatically Printing Upon Completion

This feature lets you automatically print a transaction (usually in PDF format) when you complete the transaction using iPPS. You can, for instance, use this feature to generate a Home Office PDF copy and automatically create a Home Office export file which you can later import into an agency management system.

The DPRPrint rule has been enhanced to look for the following new print type:

```
PRTYPE=COMPLETE
```

When you set the print type to COMPLETE, the DPRPrint rule automatically calls the new DPRComplete rule. The DPRComplete rule checks the CompleteType option in the Complete control group to get the actual print type, print file name, print path, file extension, and auto print recipients. You can have multiple complete types.

The DPRComplete rule then sets the appropriate attachment variables for PRTTYPE and PRINTFILE and then calls DPRPrint rule.

The DPRComplete rule expects these DSI variables and input attachment variables:

Variable	Description
DPRFORMSET	DSI variable. The form set to print, created by another rule, such as DPRLoadImportFile, DPRGetWipFormset, MTCLoadFormset, and so on.
PRTTYPE	Attachment variable. For DPRPrint to call DPRComplete, set this to COMPLETE.

These INI options are required:

```
<Complete>
    CompleteType = XXX
<Complete:XXX>
    FileType =
    FileName =
    FileExt =
    FilePath =
    Recipient =
```

Option	Description
Complete control group	
CompleteType	Specify a CompleteType. In this example, the XXX tells the system to look in the Complete:XXX control group.
Complete:XXX control group	
FileType	Enter a print file type. You can choose from PDF, PCL, XML, and so on. The default is PDF.
FileName	Enter an output file name. If you omit this option, the system creates a 46-byte unique file name.
FileExt	Enter a file extension. The default is based on your entry in the FileType option.
FilePath	Enter the print path.
Recipient	Enter the auto print recipients. You can enter a single recipient, multiple recipients separated by commas, or ALLRECIPIENTS.

NOTE: If the Complete control group includes multiple complete types, the DPRComplete rule processes each complete type.

The Recip_Names control group is required. The Printer INI options are also required, unless you are printing to XML, V2, or some other non-printer device.

Errors

Message	Description
DPR0022	The attachment variable cannot be located
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

Example

Here is an example of the request type:

```
[ ReqType:i_WipComplete]
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRGetWipFormset
    function = dprw32->DPRPrint
```

Here is an example of the input attachments:

```
CONFIG          SAMPCO
USERID          DOCUCORP
PASSWORD        DOCUCORP
RECNUM          279
PRTTYPE         COMPLETE
```

Here is an example of the INI options:

```
< Complete >
    CompleteType = COMP1
    CompleteType = COMP2
    CompleteType = COMP3
< Complete:COMP1 >
    FileType = PDF
    FileName =
    FileExt =
    FilePath =
    Recipient = HOME OFFICE,INSURED
< Complete:COMP2 >
    FileType = XML
    FileName =
    FileExt =
    FilePath =
    Recipient = INSURED,AGENT
< Complete:COMP3 >
    FileType = PCL
    FileName =
    FileExt =
    FilePath =
    Recipient = ALLRECIPIENTS
```

2125
IDS
(Available in IDS version 2.1, Patch 02)

Using the New WIP Fields in V2 and XML Import Files

Now you can output the following WIP fields in V2 and XML format when exported or printed from PPS or Docupresentment:

- Key1
- Key2
- KeyID
- TranCode
- StatusCode
- Desc
- GuidKey
- TrnName
- LocID
- SubLocID

- Jurisdictn
- QueueID

NOTE: These fields are included in the standard WIPDFD file, however, you must define them in a custom WIPDFD file if you want to use them.

For V2

When Docupresentment exports or prints transaction data to V2 format, the system checks input attachment variables for field data. If these variables are not located, the system gets the field data from WIP.

If the new fields (GuidKey, TrnName, LocID, SubLocID, Jurisdictn, and QueueID) are not defined or the field data are not found in WIP, the system looks in the WIPData control group to get a list of mapped fields and tries again to get field data from WIP.

If one or more new fields are present, the header line in the V2 file should have a record format as shown here:

```
;Key1;Key2;KeyID;TranCode;StatusCode;Desc;GuidKey;TrnName;LocID;
SubLocID;Jurisdictn;QueueID;
```

NOTE: If no values are found, the system omits all of the new fields from the header line.

For XML

When Docupresentment exports or prints transaction data into XML format, the system inserts these fields into the XML tree under the DOCSET tag. Those XML elements are output as children of DOCSET. Each element has an attribute NAME. Its attribute value is the mapped key name if it is defined in the WIPData control group, otherwise, its value is itself.

For WIP data, the system exports the WIPKEYS element with its children based on the WIP field definition. For archived data, it exports the ARCHIVEKEYS element with its children based on the archive field definition. As mentioned previously, each child element has an attribute NAME and the attribute value is from the key mapping.

If some fields are not defined in the standard WIPDFD or APPIDXDFD files, those fields are output to CUSTOMKEYS.

Here is an example of a typical XML format of output WIP fields:

```
<?xml version="1.0" encoding="UTF-8" ?>
<DOCUMENT TYPE="RPWIP" VERSION="11.3">
    <DOCSET NAME="">
    <LIBRARY NAME="" CONFIG="SAMPCO">SAMPCO</LIBRARY>
    <ARCEFFECTIVEDATE NAME="RUNDATE">20050831
    </ARCEFFECTIVEDATE>
    <KEY1 NAME="COMPANY">SAMPCO</KEY1>
    <KEY2 NAME="LOB">MULTI-PRL</KEY2>
    <KEYID NAME="POLICYNUM">205776255</KEYID>
    <TRANCODE NAME="TRANCODE">NB</TRANCODE>
    <STATUSCODE NAME="STATUSCODE">W</STATUSCODE>
    <DESC NAME="DESC">XML FIELDS EXPORT EXAMPLE</DESC>
    <LOCID NAME="" />
```

```
        <SUBLOCID NAME="" />
        <JURISDICTN NAME="" />
        <TRNNAME NAME="" />
        <QUEUEID NAME="" />
        <GUIDKEY NAME="GUIDKEY">85F0B38055624FABBB0870699BF919C7
        </GUIDKEY>
        <WIPKEYS>
            <KEY1 NAME="COMPANY">SAMPCO</KEY1>
            <KEY2 NAME="LOB">MULTI-PRL</KEY2>
            <KEYID NAME="POLICYNUM">205776255</KEYID>
            <RECTYPE NAME="RECTYPE">00</RECTYPE>
            <CREATETIME NAME="CREATETIME">20050831</CREATETIME>
            <ORIGUSER NAME="ORIGUSER">DEMO1</ORIGUSER>
            <CURRUSER NAME="CURRUSER">DEMO1</CURRUSER>
            <MODIFYTIME NAME="MODIFYTIME">20050831</MODIFYTIME>
            <FORMSETID NAME="FORMSETID">00000008</FORMSETID>
            <TRANCODE NAME="TRANCODE"">NB</TRANCODE>
            <STATUSCODE NAME="STATUSCODE">W</STATUSCODE>
            <FROMUSER NAME="FROMUSER">DEMO1</FROMUSER>
            <FROMTIME NAME="FROMTIME" />
            <TOUSER NAME="TOUSER" />
            <TOTIME NAME="TOTIME" />
            <DESC NAME="DESC">XML FIELDS EXPORT EXAMPLE</DESC>
            <INUSE NAME="" />
            <ARCKEY NAME="ARCKEY" />
            <APPDATA NAME="APPDATA" />
            <RECNUM NAME="RECNUM">3</RECNUM>
            <LOCID NAME="" />
            <SUBLOCID NAME="" />
            <JURISDICTN NAME="" />
            <TRNNAME NAME="" />
            <QUEUEID NAME="" />
            <GUIDKEY NAME="" />
            <CUSTOMKEYS>
                <KEY NAME="RECINUSE" />
                <KEY NAME="RUNDATE">20050831</KEY>
            </CUSTOMKEYS>
    </WIPKEYS>
    ...
    </DOCSET>
    </DOCUMENT>
```

This WIPData control group is set:

```
    < WIPData >
        (Standard WIP key) =
        (Standard WIP key or custom WIP field) =
        DefaultKeys2XML =
        WIPKeys2XML =
```

Option	Description
(Standard WIP key)	Use this option to map a corresponding custom WIP field so that when a XML tree is exported, the standard WIP key becomes the XML element tag name and the mapped custom WIP field becomes the value of its attribute NAME.
(*Standard WIP key or custom WIP field)*	Use this option to exclude a standard WIP key or a custom WIP field from the exported XML tree.
DefaultKeys2XML	Enter No to prevent the system from exporting all of the new WIP fields (LocID, SubLocID, Jurisdictn, TrnName, QueueID, and GuidKey). The default is Yes.
WIPKeys2XML	Enter No to prevent the system from exporting <WIPKEYS> and its children. The default is Yes.

NOTE: Keep in mind that for archived data, the control group is ArcRet and name of the fourth INI option is ARCKeys2XML instead of WIPKeys2XML. Other options are the same as those shown in this table.

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

```
< ArcRet >
    APPIDX          = d:\docserv\mstrres\Sampco\arc\appidx
    ARCPath         = d:\docserv\mstrres\Sampco\arc\
    CARFile         = d:\docserv\mstrres\Sampco\arc\archive
    Catalog         = d:\docserv\mstrres\Sampco\arc\catalog
    CARPath         = d:\docserv\mstrres\Sampco\arc
    APPIDXDFD       = d:\docserv\mstrres\Sampco\arc\AppIdx.Dfd
    Key1            = Key1
    Key2            = Key2
    KeyID           = KeyID
    Desc            = Desc
    TranCode        = TranCode
    StatusCode      = StatusCode
    GuidKey         = GuidKey
    LocID           = LocID
    SubLocID        = SubLocID
    Jurisdictn      = Jurisdictn
    TrnName         = Exclude
    QueueID         = QueueID
    ARCKeys2XML     = No
    DefaultKeys2XML= No
;   LBLimit         = 500
;   TempIDX         = d:\docserv\mstrres\Sampco\ARC\TEMP

< WIPData >
    Path            = [CONFIG:Sampco] WIPPath =
    WIPDFDFile      = d:\docserv\mstrres\sampco\DefLib\Wip.Dfd
    File            = d:\docserv\mstrres\Sampco\wip\WIP
    Key1            = Key1
    Key2            = Key2
    KeyID           = KeyID
```

```
Desc            = Desc
TranCode        = TranCode
StatusCode      = StatusCode
GuidKey         = GuidKey
LocID           = LocID
SubLocID        = SubLocID
Jurisdictn      = Jurisdictn
TrnName         = Exclude
QueueID         = QueueID
WIPKeys2XML     = No
DefaultKeys2XML= No
```

Here are some notes on selected options:

Field	Notes
ArcRet control group	
Key1	Use these options to map a corresponding custom ARC field so that when an XML tree is exported, the standard ARC key becomes the XML element tag name and the mapped custom ARC field becomes the value of its attribute NAME.
TrnName	Enter Exclude to omit a standard ARC key or a custom ARC key.
ARCKeys2XML	Enter No to prevent the system from exporting ARCKeys and their children. The default is Yes.
DefaultKeys2XML	Enter No to prevent the system from exporting all of the new ARC fields (LocID, SubLocID, Jurisdictn, TrnName, QueueID, and GuidKey). The default is Yes.
WIPData control group	
Key1	Use these options to map a corresponding custom WIP field so that when a XML tree is exported, the standard WIP key becomes the XML element tag name and the mapped custom WIP field becomes the value of its attribute NAME.
WIPDFDFile	This option defines a custom WIPDFD. Key1, Key2, and KeyID are standard WIP keys used to map these custom WIP fields: COMPANY, LOB, and POLICYNUM.
TrnName	Enter Exclude to omit a standard WIP key or a custom WIP key.
WIPKeys2XML	Enter No to prevent the system from exporting WIPKeys and their children. The default is Yes.
DefaultKeys2XML	Enter No to prevent the system from exporting all of the new WIP fields (LocID, SubLocID, Jurisdictn, TrnName, QueueID, and GuidKey). The default is Yes.

NOTE: You must map the custom fields to the standard WIP keys. Otherwise, the system exports them under CUSTOMKEYS as shown in the example XML tree format.

Keep in mind that Key fields are always exported according to the standard WIP keys. If a standard WIP key does not map to a custom WIP field, it exported as an empty tag and its attribute NAME does not have a value, such as INUSE, LOCID, and so on. Custom WIP fields that are not mapped to standard WIP keys are grouped into CUSTOMKEYS.

2129 IDS

PRINTING ON YOUR WORKSTATION PRINTER

When using iPPS or iDocumaker Workstation, you can now send print files to your workstation printers. The print files are created on the server, downloaded, and then printed on your workstation's printer.

The system displays the Printer window so you can select the printer you want to use or cancel the print job. The printer you select must support either PCL or PostScript.

The print files will have a DPP file extension and will be in PCL or PostScript format. This DPP file is generated by IDS via a request from iPPS or iDocumaker Workstation. There are no changes on the client side (plug-in) you need to make.

NOTE: When you install or update iPPS or iDocumaker Workstation, the installation process creates the necessary file association.

2148 IDS

OUTPUTTING WIP FIELD DATA ONTO THE XML TREE

Documaker Server can now export these WIP-related transaction fields onto the XML tree:

- Key1
- Key2
- KeyID
- TranCode
- StatusCode
- Desc
- GuidKey
- TrnName

- LocID
- SubLocID
- Jurisdictn
- QueueID

The XML print driver (print type XMP) will now include WIP field data in the output when it is generated from GenData's PrintFormset rule or the GenPrint program. You use the Trigger2WIP control group to map the field information. This WIP field information is included in the resulting XML tree under the DOCSET tag.

NOTE: The transaction batch record is defined by the DFD which is defined via the RCBDFDFL setting. The mapped WIP fields must be defined in the WIP DFD file or the internal WIP definition if an external DFD is not used.

Here is an example of the Trigger2WIP control group set up for field mapping:

```
< Trigger2WIP >
    Company         = Key1
    LOB             = Key2
    PolicyNum       = KeyID
    TransactionType= TranCode
```

The output XML tree should have this format:

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT TYPE="RPWIP" VERSION="11.2">
    <DOCSET NAME="">
        <LIBRARY NAME="" CONFIG="Batch Processing">Batch
Processing</LIBRARY>
        <ARCEFFECTIVEDATE>20061115</ ARCEFFECTIVEDATE>
        <KEY1 NAME="COMPANY">SAMPCO</KEY1>
        <KEY2 NAME="LOB">LB1</KEY2>
        <KEYID NAME="PolicyNum">1234567</KEYID>
        <TRANCODE NAME="TRANSACTIONTYPE">T1</TRANCODE>
        <STATUSCODE NAME="STATUSCODE"/>
        <DESC NAME="DESC"/>
. . .
    </DOCSET>
</DOCUMENT>
```

2149
IDWS

Preventing the Session from Expiring When Working on a Form

Now you can prevent the web server session from timing out when you are working on documents in iDocumaker Workstation. The time-out occurs if you leave the document open in iDocumaker Workstation for an interval longer than the time the web server allows a session to remain active. For instance, when a session times out, a save request will fail.

The session is kept current by telling iDocumaker Workstation to contact the web server when you change the current page. To use this enhancement, you must set this INI option in the CONFIG.INI file:

```
< INI2XML >
    RefreshScript = iwip18/test.htm
```

NOTE: The value of the RefreshScript option is specfic to your installation. The value shown above is only an example.

By default, iDocumaker Workstation, will not contact the web server if it has done so in the last five minutes. You can change this interval using this INI option in the WIPEDIT.INI file.

```
< WIPEdit >
    RefreshSessionTime = 600
```

Specify the interval in seconds. The example above specifies an interval of 600 seconds, or 10 minutes.

2151
IDS

Using the New DPRGetUserList and DPRModifyUser

Rules

You can use these two new rules to retrieve and modify user information from the userinfo database.

DPRGetUserList

Use this rule to retrieve user information from a user database. For every record this rule retrieves, it returns all columns except the password. This table lists the columns and the maximum amount of data the column can contain, as of version 11.2.

Column	Maximum size
SECURITY	64 bytes
PASSWORD	64 bytes
LEVEL	1byte
REPORTTO	64 bytes
USERNAME	25 bytes
INUSE	1 byte
MESSAGE	128 bytes

Syntax

```
long _DSIAPI DPRGetUserList ( DSIHANDLE hInstance,
                    char * pszParms,
                    unsigned long  ulMsg,
                    unsigned long  ulOptions )
```

Parameters

Parameter	Description
DSIHANDLE hInstance	DSI instance handle
char * pszParms	pointer to rule parameter string
unsigned long ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
unsigned long ulOptions	options

Input attachments

Variable	Description
CONFIG	Configuration

Output attachments

Variable	Description
RESULTS	Success or an error code.
RECORDS	The total number of user records.
RECORDS*X*.ID	The user ID of the Xth user record.
RECORDS*X*.USERNAME	The user name for the Xth user record.
RECORDS*X*.LEVEL	The level of user rights for the Xth user record.
RECORDS*X*.REPORTTO	The user report-to ID for the Xth user record.
RECORDS*X*.SECURITY	The user security for the Xth user record.
RECORDS*X*.INUSE	The user's InUse status for the Xth user record.
RECORDS*X*.MESSAGE	The user message for the Xth user record.

X denotes record index from 1 to the total number of user records.

INI options

These INI options are required:

```
< UserInfo >
    File = UserInfo file name
    Path = Path to locate UserInfo file
```

or

```
< UserInfo >
    UserInfo = UserInfo file name with a full path
```

Option	Description
File	Enter the name of the UserInfo file.
Path	Enter the path to the UserInfo file you entered in the File option.
UserInfo	Enter the name and full path of the UserInfo file.

NOTE: You must enter either the File and Path options or the UserInfo option.

Returns

Success or failure

Errors

These error messages may appear:

Error	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0003	The user information database, #FILENAME, # could not be opened.
DPR0004	The user ID #USERID,# is invalid.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0025	Cannot add variable #VARIABLE,# to the attachment record #RECORD,#

Example

For this example, you need this input attachment variable:

Variable	Description
CONFIG	Configuration

Here is an example of the request types:

```
[ ReqType:i_DPRGetUserList ]
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = dprw32->DPRGetUserList
```

Here is an example of the results:

```
CONFIG                SAMPCO
RECORDS               3
RECORDS1.ID           DOCUCORP
RECORDS1.INUSE        Y
RECORDS1.LEVEL        0
RECORDS1.MESSAGE
RECORDS1.REPORTTO
RECORDS1.SECURITY
RECORDS1.USERNAME
RECORDS2.ID           FORMAKER
RECORDS2.INUSE        Y
RECORDS2.LEVEL        0
RECORDS2.MESSAGE
RECORDS2.REPORTTO     DOCUCORP
RECORDS2.SECURITY
RECORDS2.USERNAME
RECORDS3.ID           USER1
RECORDS3.INUSE
RECORDS3.LEVEL        9
RECORDS3.MESSAGE
RECORDS3.REPORTTO     DOCUCORP
RECORDS3.SECURITY
RECORDS3.USERNAME
RESULTS               SUCCESS
```

DPRModifyUser

Use this rule to modify a single record or multiple user records in a user database. With this rule you can update, add, and delete information.

Syntax

```
long _DSIAPI DPRModifyUser ( DSIHANDLE hInstance,
                             char * pszParms,
                             unsigned long  ulMsg,
                             unsigned long  ulOptions )
```

Parameters

Parameter	Description
DSIHANDLE hInstance	DSI instance handle
char * pszParms	pointer to rule parameter string
unsigned long ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
unsigned long ulOptions	options

Input attachments

Variable	Description
CONFIG	Configuration
ACTION	The type of action you want performed, such as Update, Add or Delete. You can perform one action at a time.
USERS	The number of user records you want to update, add, or delete. The default is one (1).
USERS*X*.FieldName	Field name of Xth user record. ID is a required field and others are optional.

NEWUSER*X*.FieldName	The field name of Xth new user record to modify. Here are the field names and lengths:
	`SECURITY = 64 bytes`
	`PASSWORD = 64 bytes`
	`LEVEL = 1 byte`
	`REPORTTO = 64 bytes`
	`USERNAME = 25 bytes`
	`INUSE = 1 byte`
	`MESSAGE = 128 bytes`
	The field's length should not exceed its definition.

Where *X* denotes record index from 1 to the total number of user records.

To update the user record, USERSX.ID is the only required input field. It is used to locate the user record. NEWUSERSX.FieldNames specify fields to update with. You can optionally update these fields:

```
SECURITY    PASSWORD
LEVEL       REPORTTO
```

```
USERNAME    INUSE
MESSAGE
```

NOTE: You cannot update the ID.

Here is an example of input attachment variables to update user records:

Variable	Contents	Description
CONFIG	SAMPCO	Configuration
ACTION	Update	Tells the system you are updating records
USERS	2	Specifies that there are two user records to update.
USERS1.ID	USER1	The ID is only required to locate the first user record.
NEWUSERS1. PASSWORD	1234567890	Updates the first user's password with new password 1234567890
USERS2.ID	USER2	Specifies the ID of the second user record.
NEWUSERS2. LEVEL	5	Updates the second user's rights level to 5.
NEWUSERS2. USERNAME	Guest	Changes the second user's name to Guest.

To add user records, you must enter the total number of user records. You can then optionally enter these fields:

```
ID          PASSWORD
LEVEL       REPORTTO
USERNAME    SECURITY
MESSAGE
```

NOTE: Only ID is required. This prevents you from repeatedly adding the same record.

Here is an example of input attachment variables to add user records:

```
CONFIG              SAMPCO
ACTION              ADD
USERS               1
USERS1.ID           USER2
USERS1.PASSWORD     USER2468
USERS1.LEVEL        9
USERS1.USERNAME     Demo
```

To delete user records, you are required to enter the total number of user records and ID of each user record to be deleted.

Here is an example of input attachment variables to delete user records:

```
CONFIG              SAMPCO
ACTION              DELETE
```

```
USERS             3
USERS1.ID         USER1
USERS2.ID         USER2
USERS3.ID         USER3
```

Here is an example of the request types you could use:

```
[ ReqType:i_DPRModifyUser]
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = dprw32->DPRModifyUser
```

These INI options are required:

```
< UserInfo >
    File = UserInfo file name
    Path = Path to locate UserInfo file
```

or

```
< UserInfo >
    UserInfo = UserInfo file name with a full path
```

Option	Description
File	Enter the name of the UserInfo file.
Path	Enter the path to the UserInfo file you entered in the File option.
UserInfo	Enter the name and full path of the UserInfo file.

You must enter either the File and Path options or the UserInfo option.

Returns

Success or failure

Errors

These error messages may appear:

Error	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0003	The user information database #FILENAME,# could not be opened.
DPR0004	The user ID #USERID,# is invalid.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0025	Cannot add variable #VARIABLE,# to the attachment record #RECORD,#
DPR0089	Unable to update #USERID,# to the userinfo database.
DPR0090	Unable to add #USERID,# to the userinfo database.

Error	Description
DPR0091	User #USERID,# exists already. Cannot add it again.
DPR0092	Unable to delete #USERID,# from the userinfo database.
DPR0093	Cannot modify user records requested by #USERID,# because the action mode was not specified.

2174
IDS

Embedding a Subset of a Font

The PDF Print Driver now embeds only those glyphs that appear in the document when you tell it to embed fonts. The glyphs are typographical symbols, such as letters of the alphabet and punctuation symbols.

Embedding only the glyphs used in the document will result in smaller PDF files. This change affects all PDF files created by the PDF Print Driver, including PDF/A-1b-compliant PDF files.

You can, however, tell the PDF Print Driver to include all of the glyphs — not just those used in the document — by adding the SubsetAllEmbeddedFonts option to your PDF printer control group and setting it to No, as shown in this example:

```
< PrtType:PDF >
    SubsetAllEmbeddedFonts = No
```

Option	Description
SubsetAllEmbeddedFonts	If this option is set to Yes, which is the default, the PDF Printer Driver embeds only the font glyphs used in the document when it creates a PDF file. This is true for both normal PDF files and PDF/A-1b-compliant files. If you enter No, the PDF Printer Driver embeds all of the glyphs for the fonts if embedding is also enabled.