ORACLE®

INSURANCE

Oracle® Documaker

Using the Documaker Bridge

with Docupresentment version 2.2

Part number: E14902-01

August 2009

ORACLE®

Contents

Chapter 1

Using the Documaker Bridge

Via the Internet Document Server (IDS), Docupresentment lets users connect to the server via the Internet; however, executing back-end applications requires additional components, called *bridges*. The bridge components provide software rules, document templates, and other files necessary to process documents. Documaker Bridge provides a link to Documaker.

This chapter provides an overview of how the bridge works and discusses...

In addition, there are a variety of rules you can use to customize how Documaker Bridge works. For more information on these rules, see Documaker Bridge Rules on page 57.

NOTE: See the SDK Reference for information on rules you can use to control IDS.

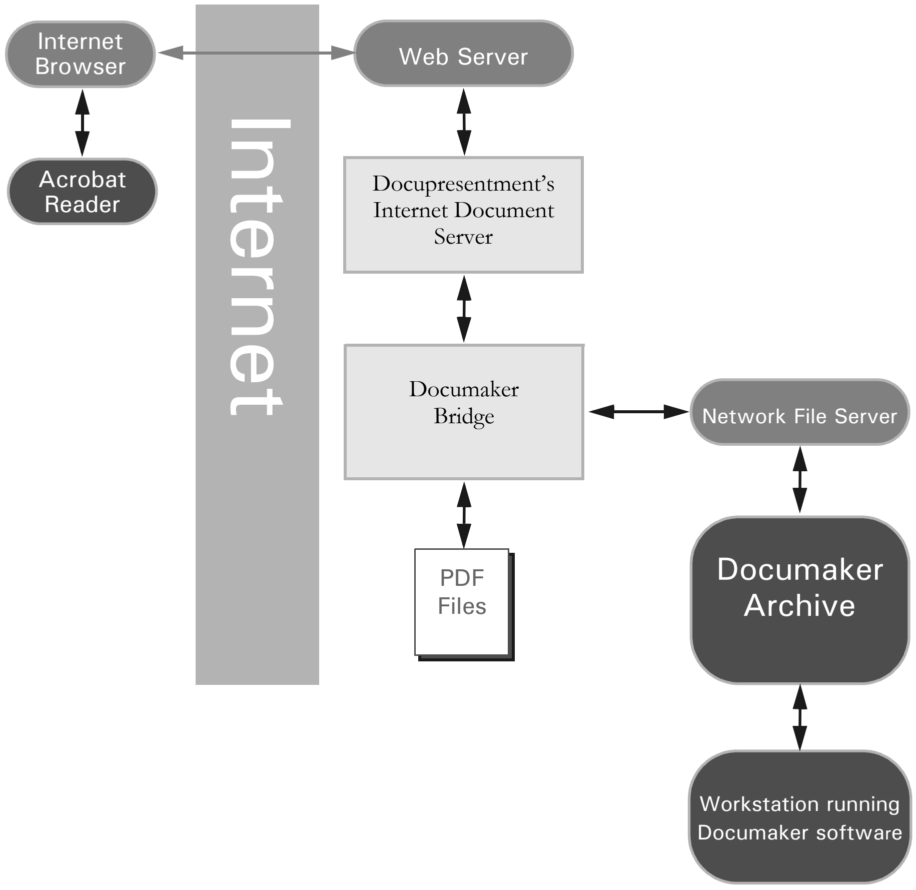
OVERVIEW

This chapter provides information on the Documaker Bridge, including archives stored in DB2 and SQL Server. It covers the capabilities of Documaker Bridge, its architecture, and product installation and setup.

The Documaker Bridge lets users retrieve archived Documaker form sets via the Internet using a web browser. Viewing the retrieved form sets requires the Adobe Acrobat Reader. The Acrobat Reader lets users view documents on-screen, just as the documents would look if they were printed.

Documaker Bridge workflow

Here is a illustration which shows how the components work together:



When a user starts his or her internet browser and connects to the Internet Document Server, a login screen appears. After logging in, the user can search for archived forms using the search screen. The Internet Document Server displays the results and the user then selects a specific form set for retrieval. The selected form set is then displayed via the Acrobat Reader.

Here is a more detailed, step-by-step discussion of how it all happens:

1. The user's browser connects to the login dialog.
2. The user submits the user ID and password. Once logged in, the user's client module communicates with the Internet Document Server.
3. The client module submits the request to the server.

4 The server processes, and accepts or rejects the log-in request. The results are posted for the client module to respond to the user.

5 If the log-in request is unsuccessful, the client module sends an error to the user. If the log-in request is successful, the client module sends the first archive query dialog to the user.

6 The user completes and submits the first archive query dialog. This dialog supplies key information used in searching the archive indexes.

7 The client module submits the request to the server.

8 The server processes the query request, finding and returning a set of matching records.

9 The client module builds a query results dialog and returns it to the user.

10 The user either requests additional records, or selects a record.

11 Via the client module, the server receives the request and fulfills it. When a request is for a specific record, the server uses Documaker Bridge to retrieve the set and examine it. A list of eligible recipients for that document set is returned to the user, via the client module.

12 The user selects a specific recipient.

13 Once the request is received by the server, via the client module, Documaker Bridge retrieves the document set and generates a PDF file. The URL reference to the temporary PDF file is returned to the user, via the server and the client module.

14 The user selects the URL of the PDF document, and the user's browser starts Acrobat Reader, which communicates to the web server, loads, and displays the PDF document set.

15 The user can view or print the displayed document using Acrobat Reader.

Checking Your Installation

To make sure Documaker Bridge is operating properly, this topic describes how to connect to the server using a browser. Once connected, you use the sample utility archive environment to retrieve and view a form set. This makes sure Documaker Bridge is operating properly.

1 Make sure your NT server and web server are up and running. The Internet Document Server (IDS) may already be running. If it is already running, you must restart IDS for the new Documaker Bridge files to take effect. If the server is not running, start IDS by accessing a command prompt on the server and entering the following:

Operating system	Location	Enter
Windows	cd (installation_directory)	cd \docserv docserver.bat
UNIX	cd (installation_directory)	cd /home/docc/int022/docserv docserver or ./docserver.sh

A message appears to tell you the Internet Document Server is ready.

2 Run the DSIEX program to test IDS. Enter one of these commands:

Operating system	Location	Enter
Windows	cd (installation_directory)	cd \docserv dsiexw32.exe
UNIX	cd (installation_directory)	cd /home/docc/int022/docserv ./dsiex or ./dsiex.sh

You should see 30 or more lines of server statistics and library data written to sysout. If you get a missing INI file diagnostic, ignore it—the defaults will suffice for this test.

If you do not see any output or the program stalls, the connection to the server is probably not being completed. If the DSIEX program fails to work at all, the most cause is one or more missing DSOs. Check the files in the directory to make sure you have everything you need, then try again.

Once you get the DOCSRV program running, leave it running while you complete the remaining steps.

3 Start your web browser and enter the following URL:

```
http://xxx.xxxxxxx.xxx/doc-html/login.htm
```

The actual URL you enter will be specific to your web server's domain name or IP address. For example, if your web server has a domain name of www.anycompany.com, you would enter:

```
http://www.anycompany.com/doc-html/login.htm
```

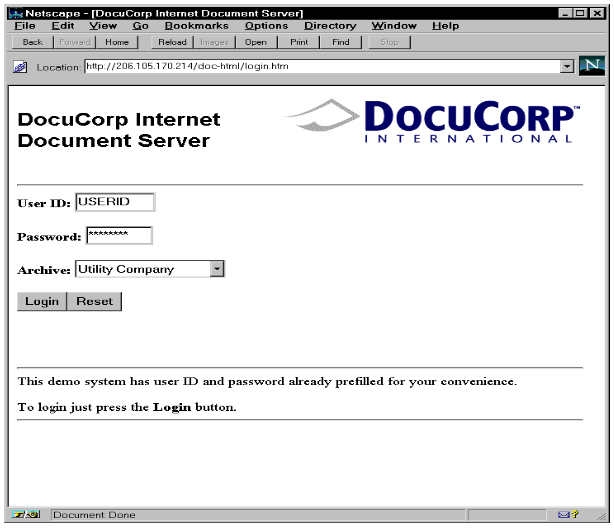
You might also have a numeric IP address rather than a domain name. It that case, enter the numeric IP address. For example, if your IP address is 255.100.101.102, you would enter:

```
http://255.100.101.102/doc-html/login.htm
```

If you are using the *UserDir* directive of Apache and your web server has a domain name of *www.myapacheserver.com* and a virtual directory of *~myuserid* you would enter:

```
http://www.myapacheserver.com/~myuserid/login.htm
```

You should now see a window similar to the following window:


Netscape - [DocuCorp Internet Document Server]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://206.105.170.214/doc-html/login.htm
DOCUCORP INTERNATIONAL
Login Reset
Document: Done

DocuCorp Internet Document Server

User ID: USERID

Password: ********

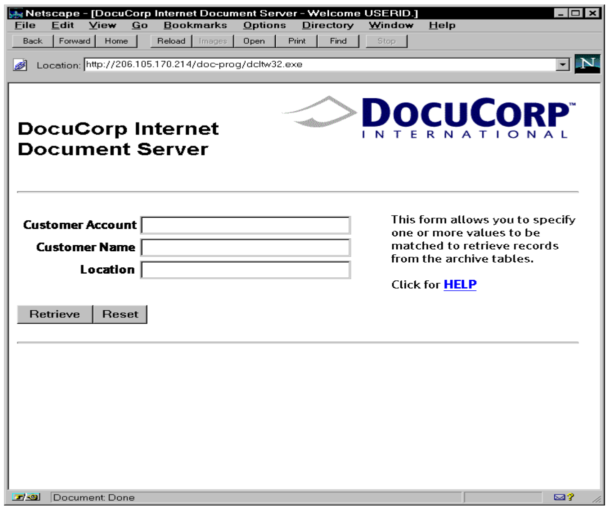
Archive: Utility Company

This demo system has user ID and password already prefilled for your convenience.

To login just press the **Login** button.

This is the example login screen. The user ID and password are filled for you. In the Archive field, you can select from several samples which show utility or financial examples.

4 Select Utility Company and click the Login button. A window similar to this appears:


Netscape - [DocuCorp Internet Document Server - Welcome USERID]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://206.105.170.214/doc-prog/dcltw32.exe
DOCUCORP INTERNATIONAL
Retrieve Reset
Document: Done

DocuCorp Internet Document Server

Customer Account

Customer Name

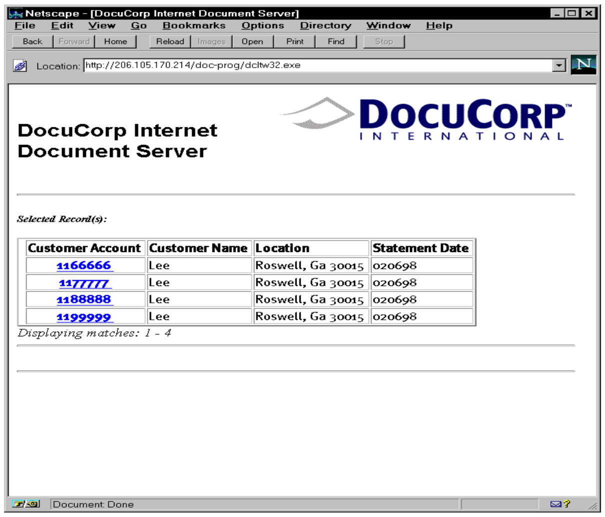
Location

This form allows you to specify one or more values to be matched to retrieve records from the archive tables.

Click for HELP

This window lets you look up specific transactions.

5 Click the Retrieve button to see all utility company examples. A window similar to the following window appears.


Netscape - [DocuCorp Internet Document Server]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://206.105.170.214/doc-prog/dcltw32.exe
DocuCorp International
Document: Done

DocuCorp Internet Document Server

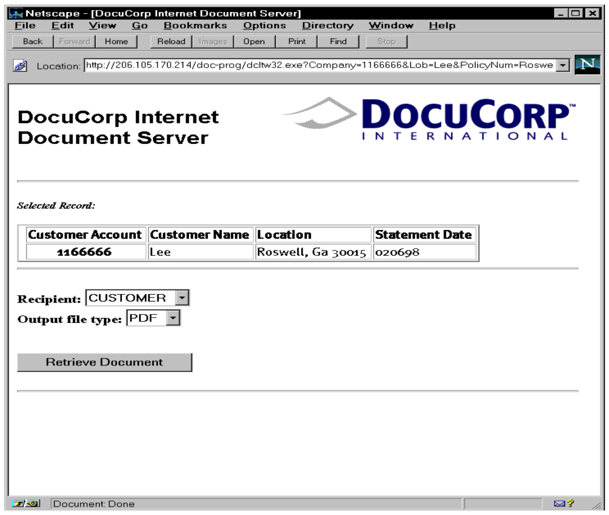
Selected Record(s):

Customer Account	Customer Name	Location	Statement Date
1166666	Lee	Roswell, Ga 30015	020698
1177777	Lee	Roswell, Ga 30015	020698
1188888	Lee	Roswell, Ga 30015	020698
1199999	Lee	Roswell, Ga 30015	020698

Displaying matches: 1 - 4

This window shows you the archive records that matched the search criteria you entered.

6 Click on the first customer account, the one at the top of the table with an account number of 1166666. You should now see a window similar to the one below:


Netscape - [DocuCorp Internet Document Server]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://206.105.170.214/doc-prog/dcltw32.exe?Company=1166666&Lob=Lee&PolicyNum=Roswe
DocuCorp International
Retrieve Document
Document: Done

DocuCorp Internet Document Server

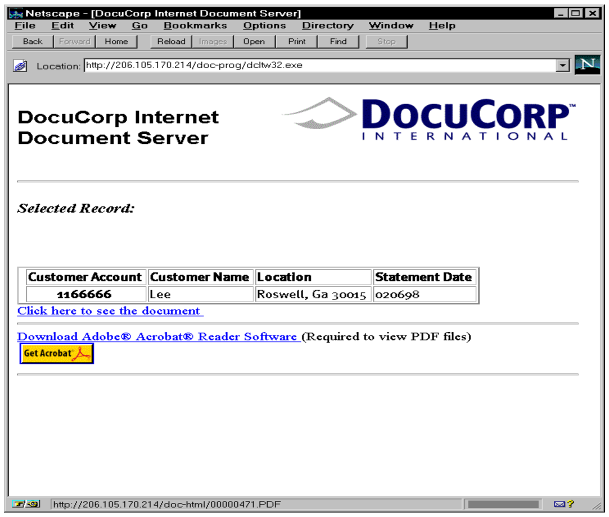
Selected Record:

Customer Account	Customer Name	Location	Statement Date
1166666	Lee	Roswell, Ga 30015	020698

Recipient: CUSTOMER
Output file type: PDF

This window lets you select which recipient's form set to view. In the case of the sample utility archive, there is only a single recipient.

7 Click the Retrieve Document button. You should now see a window similar to the one below:


Netscape - [DocuCorp Internet Document Server]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://206.105.170.214/doc-prog/dcltw32.exe
DOCUCORP INTERNATIONAL
Get Acrobat
http://206.105.170.214/doc-html/00000471.PDF

DocuCorp Internet Document Server

Selected Record:

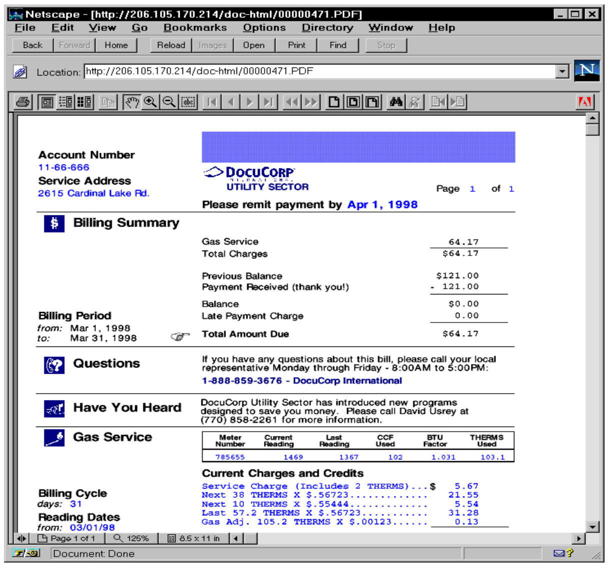
Customer Account	Customer Name	Location	Statement Date
1166666	Lee	Roswell, Ga 30015	020698

Click here to see the document

Download Adobe® Acrobat® Reader Software (Required to view PDF files)

When this window appears, Documaker Bridge has created an Adobe Acrobat file of the form set you selected.

8 Click on the highlighted text, *Click here to see document*. A window similar to the following window appears:


Netscape - [http://206.105.170.214/doc-html/00000471.PDF]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://206.105.170.214/doc-html/00000471.PDF
DOCUCORP
Page 1 of 1
125%
8.5 x 11 in
Document: Done

Account Number
11-66-666
Service Address
2615 Cardinal Lake Rd.

UTILITY SECTOR

Page 1 of 1

Please remit payment by Apr 1, 1998

Billing Summary

Gas Service	64.17
Total Charges	$64.17
Previous Balance	$121.00
Payment Received (thank you!)	- 121.00
Balance	$0.00
Late Payment Charge	0.00
Total Amount Due	$64.17

Billing Period
from: Mar 1, 1998
to: Mar 31, 1998

Questions

If you have any questions about this bill, please call your local representative Monday through Friday - 8:00AM to 5:00PM:
1-888-859-3676 - DocuCorp International

Have You Heard

DocuCorp Utility Sector has introduced new programs designed to save you money. Please call David Usrey at (770) 858-2261 for more information.

Gas Service

Meter Number	Current Reading	Last Reading	CCF Used	BTU Factor	THERMS Used
785655	1469	1367	102	1.031	103.1

Current Charges and Credits

```
Service Charge (Includes 2 THERMS)...$    5.67
Next 38 THERMS X $.56723............     21.55
Next 10 THERMS X $.55444............      5.54
Last 57.2 THERMS X $.56723..........     31.28
Gas Adj. 105.2 THERMS X $.00123.....      0.13
```

Billing Cycle
days: 31
Reading Dates
from: 03/01/98

This window shows the retrieved form using the Adobe Acrobat Reader.

Once you perform these steps, you have verified the proper operation of the Documaker Bridge.

Setting Up Your Resources

This topic explains how to modify your master resources for use with Documaker Bridge. This includes copying your master resources and updating HTML files.

Follow these steps:

1 Copy your resources into a subdirectory of the docserv\mstrres directory in the installation. For example, if you are adding RPEX1 resources to Documaker Bridge, then create a subdirectory named *rpex1* and copy your RPEX1 files and subdirectories into that new directory, as shown here:

For	Use this directory
Windows	c:\docserv\mstrres\rpex1
UNIX/Linux	cd /home/docc/int022/mstrres/rpex1

2 In the C:\DOCSERV\HTML directory, edit the LOGIN.HTM file. Edit the login.htm file located by default in the installation's docserv\html directory. The actual location of this file used by the web server is based on the setup of the web server's Virtual Directory for IDS content, examples:

For	Web server virtual directory	Physical directory
Windows	doc-html	c:\docserv\html
UNIX/Linux	~docc	/home/docc/public_html/

Add the following line after the statement...

```
<SELECT NAME="CONFIG">
```

....but change the bolded text to reflect your resource specific names:

```
<OPTION VALUE = "RPEX1">Rules Processor Example1
```

3 Create an HTML subdirectory in your resource directory. You can copy the HTML files from another set of master resources, such as the \utility\html directory for utility resources, into your resource directory and use those HTML files. For example, if you are adding RPEX1, add this subdirectory to your resource:

For	Use this directory
Windows	c:\docserv\mstrres\rpex1\html
UNIX/Linux	cd /home/docc/int022/mstrres/rpex1/html

For information on customizing the HTML files, see Customizing the HTML Templates on page 33.

Updating INI Files

You must update two IDS INI files and create a master resource INI files. Keep in mind that the master resource INI options (located in the RPEX1.INI file in the example below) take precedence over the options in the DAP.INI file.

NOTE: You will find (or create) the following INI files in the installation directory.

Client files

Add this section and replace the bolded text (RPEX1) with your resource directory name in the following files. In the docclint.ini file, add:

```
< DOCClient:RPEX1 >
    HTMLPath = mstrres\rpex1\html\
```

In the docclient.xml file, add:

```
<section name="DOCClient:RPEX1">
    <entry name="HTMLPath">mstrres\rpex1\html\</entry>
</section>
```

DAP.INI file

Add these lines and replace the bolded text (RPEX1) with your resource directory name.

```
[ Config:RPEX1 ]
    INIFile = RPEX1.INI
[ Configurations ]
    Config = RPEX1
```

RPEX1.INI file

You must create this file. Be sure to add these lines and replace the bolded text with your resource specific information. The name of this file must reflect your entry in the INIFile option in the DAP.INI file.

```
< Archival >
    ArchiveMem= Yes
< ArcRet >
    AppIdx = APPIDX
    AppIdx = c:\docserv\mstrres\rpex1\deflib\Appidx.dfd
    CARFILE = ARCHIVE
    CARPath =
    CATALOG = CATALOG
    RestartTable = RESTART
[ DB2_FileConvert ]
    APPIDX = DAP100_APP_R1
    ARCHIVE = DAP100_ARC_R1
    CATALOG = DAP100_CAT_R1
    RESTART = DAP100_RES_R1
[ DBHANDLER:DB2 ]
    CreateTable = No
    CreateIndex = No
    Database = ARCDBL
    UserID = Userid
    PassWd = Passwd
[ DBTable:APPIDX ]
    DBHANDLER = DB2
[ DBTable:ARCHIVE ]
    DBHANDLER = DB2
```

```
[ DBTable:CATALOG ]
    DBHANDLER = DB2
[ DBTable:RESTART ]
    DBHANDLER = DB2
<MasterResource>
    XRFFile = rel95sm
    DefLib = mstrres\rpex1\deflib\
    FormLib = mstrres\rpex1\forms\
    LbyLib = mstrres\rpex1\forms\
    FormFile =
<Control>
    XrfExt = .fxr
    ImageEXT = .fap
    DateFormat = 24%
< Trigger2Archive >
    Company = Company
    Lob = Lob
    PolicyNum = PolicyNum
    RunDate = RunDate
< UserInfo >
    userinfo = userinfo\userinfo
```

Database-specific Administration Tasks

If you are retrieving form sets from DB2, your DB2 System Administrator must *bind* Docupresentment's DB2 package to the DB2 system as shown below.

NOTE: If you omit this bind operation, you will receive an SQL return code -805 when you try to retrieve archived form sets.

From the DOCSERV installation directory, issue these statements:

This statement	Is used to...
DB2CMD (Windows) DB2 (UNIX)	Invoke the DB2 command line processor
DB2 CONNECT TO *ARCDB*	Connect to the DB2 Database (substitute your database name for ARCDB)
DB2 BIND DB2LIB.BND	Bind the DB2LIB Package to DB2

Customizing the Bridge

There are several tasks you can do to customize how the Documaker Bridge works with IDS and iPPS or iDocumaker. These include:

- Returning Error Messages in Attachment Variables on page 11
- Switching to Another DBMS on page 12
- Using Library Manager on page 13
- Preserving Output Files on page 14
- Automatically Printing Upon Completion on page 14

Returning Error Messages in Attachment Variables

You can return Documaker error messages in IDS attachment variables if you provide this additional attachment variable:

```
ShowErrors = Yes
```

The RPDCreateJob rule checks the input attachment variable ShowErrors is set to Yes. The RULServerJobProc rule translates and write errors into the JOBLOG.XML file if it finds errors. The ERRFILE list errors encountered during Documaker processing as shown here:

```
-------------------------------------------------------------------
GenData

Transaction Error Report - System timestamp: Mon Jul 08 15:20:06 2002

------------------------------------------------
FormMaker Data Generation (Base)

Transaction:  1234567
Symbol:  SCO
Module:  M1
State:  GA
Company Name (after ini conversion):  SAMPCO
Line of Business (after ini conversion):  LB1
Trans Type:  T1
Run Date:     19980223
------------------------------------------------
DM12050:  Error in RPProcessOneField(): Unable to
RPLocateFieldRule(pRPS, <NOSUCHTHING>).
DM12048:  Error in RPProcessFields(): Unable to
RPProcessOneField(pRPS) <FORMSET PAGE NUM OF>. Processing will
continue for image <q1vrfl>. See INI group:< GenDataStopOn > option:
FieldErrors.
==> Warning count:    0
==> Error   count:    2
End of Transaction Error Report - System timestamp: Mon Jul 08
15:20:07 2002
Elapsed Time: 1 seconds
-------------------------------------------------------------------
```

The error messages will be translated from the MSGFILE and written to the output queue as shown here:

```
RPD002010
RPD00201.ErrorTransaction:  1234567
RPD00202.ErrorSymbol:  SCO
RPD00203.ErrorModule:  M1
RPD00204.ErrorState:  GA
RPD00205.ErrorCompany Name (after ini conversion):  SAMPCO
RPD00206.ErrorLine of Business (after ini conversion):  LB1
RPD00207.ErrorTrans Type:  T1
RPD00208.ErrorRun Date:     19980223
RPD00209.ErrorDM12050:  Error in RPProcessOneField(): Unable to
RPLocateFieldRule(pRPS, <NOSUCHTHING>).
RPD002010.ErrorDM12048:  Error in RPProcessFields(): Unable to
RPProcessOneField(pRPS) <FORMSET PAGE NUM OF>. Processing will
continue for image <q1vrfl>. See INI group:< GenDataStopOn > option:
FieldErrors.
```

Switching to Another DBMS

By default, xBase is used for Documaker Bridge retrieval from archive. You can, however, override this default using INI options.

You must add DBTable control group options to archive and retrieve information from non-xBase DBMS systems such as DB2 and Oracle. Here are some examples:

For Documaker, change the FSIUSER.INI file to switch from xBase:

```
< DBTable:APPIDX >
   DBHandler = DB2 (or ORA or ODBC)
< DBTable:CATALOG >
   DBHandler = DB2 (or ORA or ODBC)
   {and any other tables used}
```

For IDS, change the RPEX1.INI file to switch from xBase to another DBMS:

```
< DBTable:APPIDX >
   DBHandler = ORA (or DB2 or ODBC)
< DBTable:CATALOG >
   DBHandler = ORA (or DB2 or ODBC)
   {and any other tables used}
```

Note: *ORA* only applies to UNIX implementations. *ODBC* only applies to Windows Implementations. For information on setting up Documaker, see the Documaker Server System Reference.

Using an ODBC driver

You can make the ODBC driver disconnect and connect again after it has been idle for specified time. To specify the time periods, use this INI option:

```
< DBHandler:ODBC >
   ConnectionTimer = 300
```

Option	Description
ConnectionTimer	Enter the number of seconds you want the driver to remain idle before reconnecting.

Also, the ODBC driver returns a specific error code if there is a communication error and the Documaker Bridge forces IDS to restart in case this error is detected. For instance, if you are writing custom code you can check for:

```
DB_ERROR_CONNECTION_FAILURE returned by DBGetLastError()
```

If found, it means the ODBC connection failed. The Documaker Bridge checks this value and causes IDS to restart.

NOTE: Only specific error codes are expected, so some communication errors might not be detected.

Recovering from ODBC errors

The Documaker Bridge restarts IDS after an ODBC connection error. This lets it automatically recover from lost connections which can occur, for example, when the SQL server is restarted. In this scenario, IDS must be restarted because although Documaker keeps the connection open for performance reasons, it does not recover if the connection is dropped.

When a transaction is executed and the Documaker Bridge encounters an ODBC error, it restarts IDS. The current transaction gets an error, but the next transaction is executed correctly if the ODBC connection is restored. If the ODBC connection cannot be restored, the next transaction gets an error as well. The connection is restored when IDS restarts.

NOTE: The Documaker Bridge only looks for specific ODBC errors. If you encounter an error which does not trigger an IDS restart and you feel a restart should occur after this error, contact Support so we can evaluate the situation and possibly add the error.

DB2 communication errors

When the DB2 connection is used by Documaker Bridge via IDS, the rules check for DB2 communication errors. DB2 servers return SQL error codes when there are communication failures. Those SQL codes are mapped to an error code and returned by the DBGetLastError function.

IDS restarts if a communication failure occurs and the DB2 connection is re-established.

USING LIBRARY MANAGER

If you are using an xBase library — not a DBMS like DB2 or Oracle— use these INI options to indicate you are using Library Manager:

```
< MasterResource >
    DDTFile     = master.lby
    FormFile    = master.lby
    LogoFile    = master.lby
    LbyLib      = ..\rpex1\
```

where *master.lby* is the name of your library and the LbyLib option points to the directory where the library resides.

You can also turn on tracing of the Library Manager component by specifying these INI options:

```
< Debug_Switches >
    Enable_Debug_Options   = Yes
    LbyLib                 = Yes
```

With these options set, the system creates a trace file you can use to resolve problems.

Preserving Output Files

You can set up Documaker Bridge so it will retain output files after they are printed or after a complete process is run. This is helpful when you need to create output files for use in third-party systems, such as archiving or policy management systems.

Note: See Automatically Printing Upon Completion on page 14 for more information on how to set up the complete process.

To give you more control of the file clean up process from the client side, the DPRPrint rule checks the DPRPERSISTOUTPUT attachment variable. If this variable is set to Yes, the output file is not registered for clean up at a later time.

For the complete process under Documaker Bridge, you can use the PersistOutput option to control file cleanup for each file type:

```
< Complete:XXX >
    PersistOutput =
```

Option	Description
PersistOutput	Enter Yes if you want Docupresentment to retain output files after they are printed or after a complete process is run. This is helpful when you need to create output files for use in third-party systems. The default is No which means these files are registered for cleanup by Docupresentment.

Keep in mind that if you set up Docupresentment to retain output files for use by third-party systems, you should set up the third-party system to clean up these files when they are no longer needed.

Automatically Printing Upon Completion

You can automatically print a transaction (usually in PDF format) when you complete the transaction using iPPS or iDocumaker. 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 looks for the following 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
```

Using Image Origins with XML Import

An output file produced from an import process can have the same image positions as an output file created from Documaker Server.

The Documaker Bridge applies image origin (position) information during XML import. The origin specified in the form definition takes precedence over the origin specified in the FAP image itself.

NOTE: This only works with master resource libraries (MRLs) built using Documaker Studio. These MRLs include the FOR, GRP, and BDF files introduced with Documaker Studio and contain origin information. The legacy model has separate DDT files that are not executed during XML import.

Detecting the Import File Type

You can use the same request type and the same attachment variables to import all supported import file types into Documaker. 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 affects the 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.

For more information, see DPRLoadImportFile on page 186.

Setting Up the DAP.INI File

The DAP.INI file is loaded by Documaker-related rules. These rules do not have access to the DOCSERV.INI file. The DOCSERV.INI file is the INI file used by the Internet Document Server. If you need to change an option used by the Documaker system, you must change the DAP.INI file.

Dynamic Configuration - Using the Config Control Group

These control groups specify the INI files to load at the transaction level. This lets you keep transaction-specific resources localized and separate from the server resources. To turn on transaction-based INI loading, be sure to include the DPRSetConfig rule in the DOCSERV.INI file. For more information, see DPRSetConfig on page 255.

For each Config:XXX control group, you must place an entry in the Configurations control group. You can have multiple values specified by the INIFile option for each of the Config:XXX control groups.

```
< Config:UTILITY >
    INIFile     = utility.ini
< Configurations >
    Config      = UTILITY
```

PDF File Creation Options

The next control groups, Printer, PrtType, and PDFFileCache, affect the creation of PDF files. For more information on PDF support, including limitations and tips on improving quality, see the Internet Document Server Guide.

Compression option

You can choose from these PDF compression methods:

Choose	For
0 (zero)	no compression
1	best speed
2	default compression
3	best compression

To override the default, add the Compression option in the PrtType:PDF control group in the DAP.INI file.

```
< PrtType:PDF >
    Compression = 3
```

BookMark option

The Bookmark option contains two values, on/off flag and bookmark level, which are separated by a comma (,). Here is an example:

```
< PrtType:PDF >
    Bookmark = Yes, Form
```

If no value is specified, the option will be set to No. The first value could be Yes or No, or simply Y or N, and it's not case sensitive. If you enter a string other than Yes, No, Y, or N, the option is set to No. The second value can be *Formset*, *Group*, *Form*, or *Page*. This value determines the lowest level the bookmarks will be set to.

For example, if you enter *Form*, bookmarks will be set for each form set, for each group in all form sets, and for each form in all groups. You can add spaces before and after the value and it is not case sensitive. If you enter anything other than *Formset*, *Group*, *Form*, or *Page*, the value will be set to *Page*.

```
< Printer >
    PrtType         = PDF
< PrtType:PDF >
    Compression     = 3
    BookMark        = Yes,Page
    Device          = NUL
    DownloadFonts   = No,Enabled
    LanguageLevel   = Level2
    Module          = PDFw32
    PageNumbers     = Yes
    PrintFunc       = PDFPrint
    SendColor       = Yes,Enabled
    SendOverlays    = No,Disabled
```

TimeOut option

Use this option to tell the system how long it should allow a PDF file to remain on disk before removing it. The default is 7200 seconds, or two hours.

```
< PDFFileCache >
    TimeOut = 7200
```

You can specify this option in the DAP.INI file or in the each of configuration INI files.

Configuring INI Files for Each Config Control Group

These control groups supply information needed to access the Documaker archive module:

```
< MasterResource >
    XRFFile     = intlsm
    DefLib      = mstrres\utility\deflib\
    FormLib     = mstrres\utility\forms\
    LbyLib      = mstrres\utility\forms\
    FormFile    = master.lby
< Control >
    XRFExt      = .fxr
    ImageEXT    = .fap
    DateFormat  = 24%
< ArcRet >
    Appidx      = mstrres\utility\arc\appidx
    Catalog     = mstrres\utility\arc\catalog
    CARPath     = mstrres\utility\arc\
< UserInfo >
    Userinfo    = userinfo\userinfo
```

NOTE: For archives stored in DB/2, Oracle, and SQL Server, there are other required INI options, such as:

```
< Archival >
```

```
ArchiveMem = Yes
```

See the archive chapter in the Documaker Server System Reference for more information.

Client Rules for Processing Requests and Results

Most of the requests and results on the CGI client are processed by the same set of rules.

Request Rules	Description
ATCUnloadAttachment	Saves attachment variables into the queue record
IRCRequest	Posts the request into the Doc Server queue
Result Rules	**Description**
ATCLoadAttachment	Parses the result queue record read from the result queue into the attachment
ATCAppend2Attachment	Appends additional values to the attachment list in memory
IRCResult	Checks the attachment list for errors returned from Doc Server.
IRCUnloadPage	Unloads the HTML page by processing HTML template that is sent back to the web browser by the web server.

Setting Up the DOCSERV.XML File

The docserv.xml file is used by the Internet Document Server to configure certain options. While this file is optional when the Internet Document Server is installed, it is required to use any of the optional bridge components.

Basic Options

The rules executed for each request are specified in this configuration file. The rules are organized by request type, as shown here.

```
<section name="ReqType:INI">
    <entry name="function">irlw32->;IRLInit</entry>
    <entry name="function">dprw32->;DPRInit</entry>
    <entry name="function">Tpdw32->;TPDInitRule</entry>
</section>
<section name="ReqType:THREADINI">
    <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
    <entry name="function">DSICoRul->;Init</entry>
    <entry name="function">DSICoRul->;Invoke,DocuCorp_IDS_DPRCo.DPR-
>;DPRCoLoginInit</entry>
</section>
<section name="ReqType:ADM">
    <entry name="function">atcw32->;ATCLogTransaction</entry>
    <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">irlw32->;IRLAdmin</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
</section>
```

Advanced Options

Running timer rules

You can use the AutorunInterval option in the configuration to set the interval at which to run the periodic timer request. The request run is SAR and can be used for occasional operations such as purging the file cache. The time is in seconds and the default is 3600, or one hour.

Scheduling when request types are run

You can use the Timers subsection in the configuration to schedule when request types are sent to IDS.

Here is an example that includes the periodic and timed requests. It is in the BusinessLogicProcessor section, messaging subsection, timed subsection:

```
<section name="timed">
    <entry name="AutoRunIntervalSeconds">3600</entry>
    <section name="Timers">
        <entry name="RRRR">Wed 10:15:00 AM</entry>
        <entry name="JJJ">09:45:00 PM</entry>
        <entry name="RBCD">23:10:00</entry>
    </section>
</section>
```

The first line tells the system to run, or send to IDS, request type RRRR each Wednesday at 10:15 AM.

The second line tells the system to run JJJ every day at 9:45 PM.

The third line tells the system to run RBCD every day at 11:10 PM.

You can spell out the day of the week if you like, just be sure to leave a space between the day and the time. You must enter the time in HH:MM:SS format. You can enter the time using a 24- or 12-hour clock. If you use the 12-hour clock, include AM or PM, as necessary.

NOTE: The actual time the request type is run may differ from the time you specify if IDS is busy processing other requests.

If the request time occurs before IDS is started, the request is postponed until the following day. After a request is executed, it is marked as executed and will not be executed again until the following day. There will be no results posted to the result queue. Here are some more examples of how you can enter the time:

If you enter	IDS treats this as
Friday 13:00:00 AM	Fri 1:00:00 PM
Tue 15:00:00 PM	Tue 3:00:00 PM
Thur 17:00:00	Thu 5:00 PM
19:00:00	7:00 PM every day

Setting Up the Client Configuration Files

The client configuration files, docclnt.ini and docclient.xml, are initialization files used by client programs.The docclnt.ini file is used by the CGI executable program and the docclient.xml file is used by other executables, Java client programs, and Microsoft ActiveX controls and Active Server Pages.

NOTE: Before version 2.0, installations of Internet Document Server used a docclnt.ini file; the 2.0 install procedure can convert this file into an docclient.xml file.

Basic Options

Using the docclnt.ini file, you can change the location of your HTML templates. The default location is in the TMPL directory, located in the current directory. To specify the location of HTML templates, use the following INI option:

```
< DOCClient >
    HTMLPath = tmpl\
```

NOTE: This directory should always include ERRORS.HTM template, so if the CGI client program detects the error it can produce an error message and send it to the web browser.

To specify the location of CONFIG specific HTML templates, use:

```
< DOCClient:UTILITY >
    HTMLPath = mstrres\utility\html\
```

UTILITY is the value of HTML form variable CONFIG. If this variable value is set, the CGI client program looks for HTML templates in this directory.

You can also specify the name and location of the request queue. This value should be the same as the value set for the Internet Document Server. See Setting Up the DOCSERV.XML File on page 21 for more information.

```
< RequestQ >
    Name = REQUESTQ
```

Similarly, you can also specify the name and location of the result queue:

```
< ResultQ >
    Name = RESULTQ
```

To specify a list of rules to run on all requests, use:

```
< REQType:Default >
    function = atcw32->ATCUnloadAttachment
    function = ircltw32->IRCRequest
< RESType:Default >
    function = ATCw32->ATCLoadAttachment
    function = ATCw32->ATCAppend2Attachment
    function = ircltw32->IRCResult
    function = ircltw32->IRCUnloadPage
```

To specify a list of rules to run on a PRT request, use:

```
< RESType:PRT >
```

```
        function = ATCw32->ATCLoadAttachment
        function = ATCw32->ATCAppend2Attachment
        function = ircltw32->IRCResult
        function = ircltw32->IRCPrint
        function = ircltw32->IRCUnloadPage
```

NOTE: The PRT request will not execute rules in the REQTYPE:Default control group because it has to run one extra rule, IRCPrint. For more information about this rule, see the SDK Reference.

To specify a list of rules to run on an ERR request, use:

```
    < RESType:ERR >
        function = ircltw32->IRCUnloadPage
```

NOTE: An ERR request indicates a processing error and is posted by the Internet Document Server. It should not be coming from an HTML page as the value.

To specify a list of rules to run on CAD (Client Administration) request, use:

```
    < REQType:CAD >
        function= ircltw32->IRCAdmin
        function= atcw32->ATCUnloadAttachment
        Post    = N
```

(INI value Post = N has to be set for this request. It means that the request is not posted to the Internet Document Server, it is processed by the client.)

```
    < RESType:CAD >
        function = atcw32->ATCLoadAttachment
        function = atcw32->ATCAppend2Attachment
        function = ircltw32->IRCUnloadPage
```

To specify a list of rules to run on SCS (Client Statistics) request, use:

```
    <REQType:SCS>
        function= ircltw32->IRCSendVersion
        function= atcw32->ATCUnloadAttachment
        Post    = N
```

You must set the Post option to N for this request. It means that the request is not posted to the Internet Document Server, instead it is processed by the client.

```
    < RESType:SCS >
        function= atcw32->ATCLoadAttachment
        function= ircltw32->IRCUnloadPage
```

To specify the location and names of HTML templates for different request types, use…

```
    < DOCClient:CONFIG >
        HTMLPath = (in this case config is UTILITY)
```

…or in the directory, specified in these control groups:

```
    < DOCClient >
        HTMLPath =
    < HTMLTemplates:LGN >
```

```
       Page = search.htm
  < HTMLTemplates:DEFAULT >
       Page = errors.htm
  < HTMLTemplates:SCH >
      Page = records.htm
  < HTMLTemplates:RCP >
      Page = recips.htm
  < HTMLTemplates:PRT >
      Page = printout.htm
  < HTMLTemplates:ERR >
      Page = tmpl\errors.htm
  < HTMLTemplates:SSS >
      Page = srvstats.htm
  < HTMLTemplates:ESS >
      Page = srvstats.htm
  < HTMLTemplates:RSS >
      Page = srvstats.htm
  < HTMLTemplates:RRP >
      Page = printout.htm
  < HTMLTemplates:ADM >
      Page = tmpl\currini.htm
  < HTMLTemplates:CAD >
      Page = tmpl\currini.htm
  < HTMLTemplates:RAD >
      Page = tmpl\currini.htm
  < HTMLTemplates:SCS >
      Page = cltstats.htm
```

Each value can have fully specified path with the name or just the name. If it does not have an explicit path, the CGI client program looks for the file in the directory specified in the INI control group.

The values specified in the Attach:DEFAULT control group specify the values to append to the attachment if you use the ATCAppend2Attachment rule. These values include location of the HTML files, name of the CGI client program, location of output PDF files, and so on. When the CGI client program processes the HTML template, it replaces values like #BASELOCATION,# with values from this group with the same option name.

```
  < Attach:DEFAULT >
      Baselocation= http://206.105.170.214/doc-html/
      Exename     = /doc-prog/dcltw32.exe
      PrintPath   = html\
      DocType     = DAP
```

Advanced Options

Generating unique IDs

To specify the name and location of database table for generating unique IDs, use:

```
  < UniqueDB >
      name = .\UNIQDB
```

This file can be different for the client and the server. The default is *UNIQDB*.

Setting time-out values

You can specify the time-out value for the client program in each of the request type INI control groups. This value is set in seconds and is defaulted to 60, or one minute. If you get errors because the client program times out and does not receive results from the Internet Document Server, try increasing this value.

Decreasing this value will not make the Internet Document Server or the CGI client run faster. Adjust this value only if needed. When you change the default time-out value for a request type in the DOCCLNT.INI file, the request type should call these rules:

```
atcw32->ATCUnloadAttachment
ircltw32->IRCRequest
```

If the request type has no rules, the time-out value setting is skipped and the ReqType default time-out (60 seconds) is used. For example to change the time-out value to two minutes, set the INI options as shown here:

```
< ReqType:XXXX >
    atcw32->ATCUnloadAttachment
    ircltw32->IRCRequest
    Timeout = 120
```

You can also set up global time-out settings, so even if the ASP page specifies some other value, you can overwrite it. You specify global time-out settings using these options:

```
< ResultQ >
    DefaultTimeout = 60000L
    MaxTimeout     = 90000L
    MinTimeout     = 60000L
```

Option	Description
DefaultTimeout	Enter, in milliseconds, the time-out to use if the application did not specify one. The default is 15000L or 15 seconds.
MaxTimeout	Enter, in milliseconds, the maximum amount of time to wait. If the application specifies a longer time-out period, the system uses this value instead. This option lets you handle situations which can occur when ASP pages specify a time-out that exceeds the IIS global setting limits. Setting this option to the same value as IIS script time-out keeps you from having to edit all ASP pages where the time-out was specified as too long.
MinTimeout	Enter, in milliseconds, the minimum amount of time to wait. If the application requests a time-out that is less than this value, the system uses this value instead. Setting this option keeps you from having to edit all ASP pages where the time-out was specified as too short.

NOTE: If you set the DefaultTimeout option outside the limits set for the MinTimeout and MaxTimeout options, the system uses the values for the MinTimeout and MaxTimeout options.

Debugging information

The CGI client program debug option defaults to No. If you set this value to Yes, the client program creates the DCLNTTRC.LOG file which contains additional debugging information. This option is used for debugging purposes only.

```
< DOCClient >
    Debug = Yes (or No)
```

Here is an example DCLNTTRC.LOG file produced with request type LGN:

```
1. DCLTReadPostData Method POST: <57>
2. Successfully read post data
3. DCLTParsePostData received:
<REQTYPE=LGN&CONFIG=UTILITY&USERID=USERID&PASSWORD=PASSWORD>
4. Successfully parsed post data
5. Located request type : LGN
6. Successfully loaded function for request
7. Successfully ran pre-post functions
8. Successfully posted data
9. Waiting for server. Sleep is set to 1000 msec
10. Got results. Now processing
11. Successfully loaded post-post rules
12. IRCUnloadPage found CONFIG value, using INI group
<DOCCLIENT:UTILITY>
Successfully ran post-post rules
```

AUTHENTICATING USERS

You can make sure all users are authenticated before they view content which contains confidential information or client data. This authentication must be repeated each time a user views a page. To authenticate users, you will use these rules:

- DPRCheckLogin on page 87
- DPRDecryptLogin on page 97
- DPRDefaultLogin on page 100
- DPRLoginUser on page 196
- DPRGenerateSeedValue on page 128

User IDs and passwords are not authenticated on the HTTP server. Authentication is performed on application server (IDS) in the network.

NOTE: The password is case sensitive. If you need the password to not be case sensitive, make the client application convert the password to uppercase before it submits the password to IDS.

The authentication token includes the user ID and a password hash value. For browsers that accept cookies, you can store the token as a cookie. For browsers that do not accept cookies, the token information is carried in the HTTP request.

Cookies should be set to expire in 30 minutes, although each request can reset the cookie an additional 30 minutes. At a predetermined time each day, such as at 2:00 AM, the salt value is reset and all existing password hashes become invalid.

All subsequent login attempts pass the authentication token, which includes user ID and password hash value. For token-based authentication, the internal application (IDS) compares the past password's hash value to the user's computed password hash value. Token-based failures return the client to login screen (without a login failed message). If token values are missing, the user should be redirected to login screen.

Initial login flow

Here is the initial login flow:

- Internet application submits the USERID and PASSWORD values to IDS.
  - If these values are encrypted, they will be decrypted later.
  - If these values are not encrypted, the Internet application should also provide this value:

    ```
    PASSWORDENCRYPTED=NO
    ```
- IDS preprocessing (message DSI_MSGRUNF) begins.
  - The DPRDecryptLogin rule decrypts USERID and PASSWORD and adds the clear text version of USERID to the input attachment. Password hash is created and added to the input attachment and clear text version is removed.
  - The DPRDefaultLogin rule uses the USERID value from input attachment and locates a matching record in the user table. By default, the rule uses the USERINFO table. The values of USERID and PASSWORD from that table are added to the input attachment as REALUSERID and REALPASSWORD.

    - The DPRLoginUser rule creates a hash value from REALPASSWORD and compares USERID with REALUSERID and the hash value in PASSWORD with hash value of REALPASSWORD.

- IDS post processing (message DSIMSG_RUNR) begins.
    - The DPRLoginUser rule adds the LOGINRESULT value to the output attachment.
    - The DPRDefaultLogin rule removes the values for REALUSERID and REALPASSWORD from the input and output attachments.
    - The DPRDecryptLogin rule encrypts the value for USERID, adds the password hash to it and encrypts the resulting string again. The new value is the *authentication token*. This value is appended to the output attachment as the USERID. The Internet application passes the USERID to IDS on all subsequent requests.
- In case of error, the rules create the attachment variable LOGINRESULT with the value FAILURE and call the DSIErrorMessage API. The internet application can check for a specific error code in the attachment variable RESULTS, but it is best to simply redirect the user to the login screen if LOGINRESULT is not SUCCESS.
- In case of error, the value for the authentication token is omitted from the output attachment.

Data request flow

Here is a summary of the data request flow:

- The Internet application submits the authentication token. This token is returned to IDS by the initial login processing as *USERID*.
- IDS preprocessing (message DSI_MSGRUNF) begins.
    - The DPRDecryptLogin rule decrypts the authentication token and splits it into the USERID and PASSWORD hash. The rule then decrypts the USERID value and adds the clear text USERID and PASSWORD hash to the input attachment as *USERID* and *PASSWORD*.
    - The DPRDefaultLogin rule uses the USERID value from input attachment to locate a matching record in a user table, by default the USERINFO table. The rule adds the USERID and PASSWORD values from the table to the input attachment as REALUSERID and REALPASSWORD.
    - The DPRCheckLogin rule creates a hash value from REALPASSWORD and compares USERID with REALUSERID and the hash value in PASSWORD with hash value of REALPASSWORD.
- IDS post processing (message DSI_MSGRUNR) begins.
    - The DPRCheckLogin rule adds the value of LOGINRESULT to the output attachment.
    - The DPRDefaultLogin rule removes the values for REALUSERID and REALPASSWORD from the input and output attachments.
    - The DPRDecryptLogin rule recreates the authentication token and adds it to the output attachment as USERID.

The internet application should pass this value to IDS on all subsequent requests. This token is the same as the token passed to the Internet application on the initial login.

- If there are errors, the rules create the attachment variable LOGINRESULT with the value FAILURE and call the DSIErrorMessage API. The internet application can check for a specific error code in the attachment variable RESULTS, but it is best to just redirect the user back to the login window if LOGINRESULT is not SUCCESS.

- If there are errors, the value for USERID (authentication token) is missing from the output attachment.

Changing seed value for the password hash

You can change the password hash seed value on the timer request. Once the value is changed, none of those generated with different seed value authentication tokens are valid.

Use the DPRGenerateSeedValue rule to reset the seed. You should execute this rule at least once a day.

Example

IDS rules use global data APIs to store the seed value, so IDS servers should be set up for global data APIs. The configuration options for all IDS servers, are shown here:

```
<section name="GlobalData">
  <entry name="Path"> </entry>
</section>
```

should point to the same valid directory. This option defaults to .*global*\, so if you use the default, create the directory global under the directory where IDS is running.

Here are example INI options for implementing the authentication schema with the sample Documaker archive/retrieval setup. Note the use of the DPRSetConfig rule before the login rules, this is done so you can specify the location of the Documaker USERINFO table for each setup.

```
<section name="ReqType:LGN">
    <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">irlw32->;IRLCopyAttachment</entry>
    <entry name="function">dprw32->;DPRDecryptLogin</entry>
    <entry name="function">dprw32->;DPRDefaultLogin</entry>
    <entry name="function">dprw32->;DPRLoginUser</entry>
</section>
<section name="ReqType:PRT">
    <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">dprw32->;DPRInitLby</entry>
    <entry name="function">dprw32->;DPRPrintFormset</entry>
</section>
<section name="ReqType:RCP">
```

```
        <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">dprw32->;DPRGetRecipients</entry>
    </section>
```

Use these configuration options to reset the seed value every day at 3:00 AM.

```
<section name="Timer">
  <entry name="ResetSeed">3:00:00 AM</entry>
</section>
<section name="ReqType:RESETSEED">
  <entry name="function">dprw32->;DPRGenerateSeedValue</entry>
</section>
```

Customizing the login process

The best way to customize the login process is to replace the DPRDefaultLogin rule. Use the rest of the rules as designed. If you create a custom login rule to replace the DPRDefaultLogin rule, the custom rule should do the following:

- Preprocessing (message DSI_MSGRUNF)
    - Check the LOGINRESULT value in the input attachment. If it exists and is not SUCCESS, do nothing.
    - Locate the USERID in the input attachment.
    - Determine the password for the user ID. For example, you could query a custom user table and add the password value to the input attachment as REALPASSWORD and the user ID as REALUSERID.
    - If there are errors, issue an error message and add LOGINRESULT to the input attachment with the value FAILURE.
- Post processing (message DSI_MSGRUNR)
    - Remove the REALUSERID and REALPASSWORD from the input and output attachments. If these values are missing, do not issue an error message.

Using Manually-Edited HTML Forms with Real-Time HTML Processing

Documaker Bridge can 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 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 the FAP file 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 option 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.

Customizing the HTML Templates

The example HTML templates provided with Documaker Bridge work with the sample Documaker archive environment (UTILITY). Implementing Documaker Bridge with an existing Documaker archive requires either modifying the example HTML files or creating new HTML files for your archive.

This topic discusses the modifications typically necessary to use the example HTML files with another Documaker archive. Even if it is necessary to create new HTML files, rather than modify the example files, this topic points out the archive-specific components of the HTML files.

The architecture of the archive retrieval process was demonstrated when you verified the proper installation of Documaker Bridge earlier in this chapter. In summary:

1. The first HTML display is the Login page.
2. Once the user is verified, the bridge presents a search page which contains fields for the Company, Line of Business, and Policy Number. When this page is submitted, the bridge searches the archive based on the data entered in these fields.
3. The next HTML page sent by the bridge returns the search results in table format. Each table entry has an HTML link for that particular form set.
4. When a user selects one of the table entries, the bridge returns the recipient selection page, which lets the user select from the recipients originally available for that particular form set.
5. Once the user selects the recipient, the bridge generates the PDF file and sends the next HTML page, which again displays the form set information and contains a HTML link to the PDF file.
6. The user can then click the file link and view the PDF file using Acrobat Reader.

The key to interfacing the bridge to a particular Documaker archive is the archive-specific components in the various HTML files. There are several example HTML files included with Documaker Bridge:

- LOGIN.HTM
- SEARCH.HTM
- RECORDS.HTM
- RECIPS.HTM
- PRINTOUT.HTM

Assuming you installed Documaker Bridge without changing the directory structure, the LOGIN.HTM file will be located in the DOCSERV\HTML directory, and the other HTML files will be in the DOCSERV\MSTRRES\UTILITY\HTML directory.

We'll now look at each of these HTML files to explain how they work and what will need to be changed to make these example files work with another Documaker archive.

Setting Up the Login Page

The login HTML page is the first page that the browser loads when it connects to Documaker Bridge. The login page can contain any information you want, including links to other pages, links to email, and so on.

This page is not processed by the Internet Document Server and is not an HTML template, rather is it a content page that is automatically sent to the browser. The only part of the login page Documaker Bridge needs is the HTML form with these required fields:

- USERID
- PASSWORD
- REQTYPE

and the optional value:

- CONFIG

Documaker Bridge uses these fields to verify the user, using the USERINFO table, and to start the retrieval process.

The following is part of the LOGIN.HTM HTML page, which shows the form that lets you enter your user ID and password:

```
<FORM METHOD=POST ACTION=”/doc-prog/dcltw32.exe”>
<INPUT NAME=”REQTYPE” value=”LGN” TYPE=”HIDDEN”> <BR><BR>
<INPUT NAME=”CONFIG” value=”UTILITY” TYPE=”HIDDEN”> <BR><BR>
<tr>
<b>User ID: </b> <INPUT SIZE=10 MAXLENGTH=8 NAME="USERID"
value="USERID"> <BR><BR>
</tr>
<tr>
<b>Password: </b> <INPUT TYPE=PASSWORD SIZE=8 MAXLENGTH=8
NAME="PASSWORD" VALUE=PASSWORD><P>
</tr>
<P>
<INPUT TYPE="submit" VALUE="Login"><INPUT TYPE="reset"
VALUE="Reset"><P>
</FORM>
```

The name of executable in…

```
ACTION=”/doc-prog/dcltw32.exe”
```

… is the name and location of the client program. Note that this name is relative to the web server's root directory or Virtual Directory Alias for the CGI-BIN program execution. For information on the web server root directory location and setup, consult your web server manuals. Here are some examples:

For	Enter
Windows	ACTION=”/doc-prog/dcltw32.exe”
UNIX/Linux	ACTION=”~docc/cgi-bin/doccgi.sh”

This part of the login page defines these HTML form variables:

Variable	Description
USERID	The entry field for the user ID
PASSWORD	The entry field for the user’s password. Asterisks (*) appear as the user types the password.
REQTYPE	Hidden, the value is LGN.
CONFIG	Hidden, the value is UTILITY.

and two buttons:

Button	Description
LOGIN	The user presses this button after entering values in the USERID and PASSWORD fields and selecting the CONFIG value.
RESET	The user can press this button to reset the values in the fields to their original values—blank in this example.

When the user enters the values and clicks on LOGIN button, the web server invokes the client program specified in the ACTION attribute of the HTML FORM. The client program receives the values of all the form variables.

The client program receives the request of type REQTYPE, LGN in this case, and runs rules registered for this REQTYPE. The rules needed to process the LGN are stored in the docserv.xml file:

```
<section name="REQTYPE:LGN">
    <entry name="function">atcw32->;ATCLogTransaction</entry>
    <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">dprw32->;DPRSetConfig</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
    <entry name="function">dprw32->;DPRLogin</entry>
    <!-- -->
</section>
```

Processing the LGN request on the server creates these attachment variables:

- RESULTS - SUCCESS or error code
- USERID
- PASSWORD
- RIGHTS
- REPORTTO
- SECURITY
- USRMESSAGE

These fields are part of USERINFO record, and the values are set when the record matching USERID is found in the file.

NOTE: An attachment is a block of information accessed in the form of name/value pairs. Attachments are used to pass information between the client and the server rules, as well as the API.

If you are using WebSphere MQ or JMS queues, the size of all variables and attachments combined is limited to available memory or to the limit set by the messaging system.

Setting Up the Search Page

The search HTML page is returned by the client program after it processes the LGN request. The client creates this page by processing the SEARCH.HTM template. Generally, the search page can contain any information you want, including links to other pages, links to email, and so on. The only part Documaker Bridge needs is the HTML form with these required fields:

- USERID
- REQTYPE
- FIELDS

and optional values:

- CONFIG
- MAXRECORDS (the default is 20)
- PARTIALMATCH
- TABLEINIGROUP (the default is ArcRet)
- TABLEINIOPTION (the default is AppIdx)

NOTE: TABLEINIGROUP and TABLEINIOPTION are advanced values that should not be changed in most situations.

The values for each field in the FIELDS variable are required. For example, if the FIELDS value is *Company,Lob,PolicyNum*, then the values for the Company, Lob, and PolicyNum are required.

These FIELD variables are archive-specific and must match the archive keys contained in the application index (APPIDX) for the archive. For example, if your Documaker archive used the keys ACCOUNT, NAME, and POLNO instead of Company, LOB, and PolicyNum, the HTML template must be modified to use those key names to work with that archive.

Here is part of the SEARCH.HTM file that shows the relevant HTML form:

```
<FORM METHOD=POST ACTION="#EXENAME,#">
<INPUT NAME="REQTYPE" value="SCH" TYPE="HIDDEN">
<INPUT NAME="USERID" value="#USERID,#" TYPE="HIDDEN">
<INPUT NAME="DOCTYPE" value="#DOCTYPE,#" TYPE="HIDDEN">
<INPUT NAME="CONFIG" value="#CONFIG,#" TYPE="HIDDEN">
<INPUT NAME="FIELDS" value="Company,Lob,PolicyNum" TYPE="HIDDEN">
<INPUT NAME="PARTIALMATCH" value="YES" TYPE="HIDDEN">
<INPUT NAME="MAXRECORDS" value="15" TYPE="HIDDEN">
<table cellpadding=0>
<tr><td align=right>
<b>Customer Account</b></td>
<td> <INPUT SIZE=30 MAXLENGTH=30 NAME="Company" > <br>
</td></tr>
<tr><td align=right>
<b>Customer Name</b></td>
<td> <INPUT SIZE=30 MAXLENGTH=30 NAME="Lob" > <br>
</td> </tr>
<tr><td align=right>
<b>Location</b></td>
<td> <INPUT SIZE=30 MAXLENGTH=30 Name="PolicyNum"> <br>
</td> </tr>

</table>
</TD>

<TD WIDTH=30% VALIGN=TOP>
This form allows you to specify one or more values to be
matched to retrieve records from the archive tables.
<P>
Click for <a href="help.htm"><B>HELP</B> </a> <br>
</TD>
</TABLE>
<P>
<INPUT TYPE="submit" VALUE="Retrieve"><INPUT TYPE="reset"
VALUE="Reset"><P>
</FORM>
```

The name of executable in…

```
ACTION="#EXENAME,#"
```

…is the name and location of client program. This value is replaced with the actual executable name by the rules on LGN request. Note that this name is relative to web server root directory. For information on web server root directory location and setup, consult your web server documentation.

This part of the search page defines these HTML form variables:

Variable	Description
USERID	An entry field for user input
REQTYPE	A hidden field, invisible to the user (the value is SCH)
CONFIG	A hidden field, invisible to the user (the value is #CONFIG,#)
FIELDS	A hidden field, invisible to the user (the value is Company,Lob,PolicyNum)
Company	An archive field containing the Customer Account Number
Lob	An archive field containing the Customer Name
PolicyNum	An archive field containing the Customer Location

and two buttons:

Button	Description
RETRIEVE	Press after entering search criteria to retrieve the information.
RESET	Press to reset the values of the fields to their original values.

Remember that to make this template work with another Documaker archive, the archive-specific values must be changed to match those used by the Documaker archive. These values include the archive keys (as discussed earlier) and possibly drop-down selections.

When the user enters the values and clicks on the Retrieve button, the web server invokes the client program specified in the ACTION attribute of the HTML FORM. The client program receives the values of all the form variables.

The client program receives the request of type REQTYPE, SCH in this case, and runs rules registered for this REQTYPE. The query executed on the DOC server is presented as…

```
SELECT FROM IndexTable WHERE Company='CompanyValue' .AND. Lob =
'LobValue' .AND. PolicyNum = 'PolicyNumValue'.
```

…where CompanyValue, LobValue, and PolicyNumValue are the values entered into the entry fields. You have these optional values:

Value	Description
PARTIALMATCH	If this variable is present on the HTML form, the search is executed with partial match criteria, if this variable is not present, it means exact match.
TABLEINIGROUP	Name of the group in the INI file, that is supposed to specify table name. If you omit this value, the rule uses the ArcRet control group. Again, this is an advanced setting and should not be used in most situations.
TABLEINIOPTION	Name of the INI option specifying the actual table name. If this value is omitted, the system defaults to AppIdx. This is an advanced setting and should not be used in most situations.
MAXRECORDS	Maximum number of matching records to return. This value defaults to 20. If this value is too big, the waiting time for getting the query results might be too long. Adjust this value carefully.

The processing of the SCH request on the server creates these attachment variables:

- RESULTS - SUCCESS or reason for the error
- MORERECORDS - set to YES if there are more records than returned
- RECORDS - number of records in the result set
- RECORD##.FIELD1 - values for each of the columns in the table and for each returned record. FIELD1 is the actual column name.

Performance considerations

This search rule can be used on virtually any table in the supported database formats. Complex queries are not executed fast and could result in the client program timing out or just unacceptable wait times. The exact match query (omitted PARTIALMATCH HTML variable) is always faster than partial match. Consider using it, where possible. Some of the search values (such as state) might have limited range of values, so try creating the drop-down on the HTML page, for user to pick from, and use exact matches.

Not all the databases are equal in performance. For example, for faster performance codebase requires tags (indexes) created on all the search columns. If exact match is used, it will run faster if the combined index is created, for example, if the exact match search is specified on Company, LOB, and PolicyNum columns, the composite index would have to have an expression as Company+Lob+PolicyNum.

Setting Up the Records Page

The records HTML page is returned by client program after it processes the SCH request. This page shows the results of the search in tabular format. The client creates this page using the RECORDS.HTM template. The records page can contain any information you want, including links to other pages, links to email, and so on. The only part of it the Bridge needs is the HTML form with these required fields:

- USERID
- REQTYPE
- ARCKEY

and this optional value:

- CONFIG

Here is part of the RECORDS.HTM file:

```
<html>
<head>
<base href="#BASELOCATION,#">
<meta http-equiv="keywords" content="Docucorp Retrieve">
<title> RETRIEVE RECORD </title>
</head>
<body bgcolor = "#FFFFFF" link=#0000ff vlink=#2525b5
alink=#ff0000 text="#000000" >
<IMG ALIGN=RIGHT TOP SRC="nlogo30s.gif" alt=fsilogo> <BR>
<BR>
<H2><B> Document Retrieval  </B> </H2><BR>
<hr>
<BR>
<H5><B> <I> Selected Record(s): </I></B> </H5>
<TABLE BORDER=2 CELLPADDING=1 COLOR="blue">
<TR><TD><TH> <B> Customer Account </TH></TD>
    <TD> <B>  Customer Name </TD>
    <TD> <B>  Location </TD>
    <TD> <B>  Statement Date </TD>
</TR>
<!-- DCL BEGIN SECTION;NAME=RECORDS;SKIPONEMPTY; -->
<TR>
    <TD><TH>
    <A
HREF="#EXENAME,#?Company=#Company,URL#&Lob=#Lob,URL#&PolicyNum=#Pol
icyNum,URL#&RunDate=#RunDate,URL#&ARCKEY=#ARCKEY,URL#&USERID=#USERI
D,URL#&REQTYPE=RCP&DOCTYPE=#DOCTYPE,URL#&CONFIG=#CONFIG,URL#&DESC=#
DESC,URL#&CREATETIME=#CREATETIME,URL#&">
    #Company,%s# </TD> </TH>
    <TD> #Lob,%s# </TD>
    <TD> #PolicyNum,%s# </TD>
    <TD> #RunDate,%s# </TD>
</TR>
<!-- DCL END SECTION -->
</TABLE>

</body>
</html>
```

Notice that each company value in the table is a HTML link. When the user selects a record by clicking one of the links, the web server starts the client program you specified in the HREF attribute of the HTML page. The client program also receives the command line parameters you specified on the HREF line.

These variables are archive-specific and must match the archive keys contained in the application index (APPIDX) for that archive. For example, if a Documaker archive used the keys ACCOUNT, NAME, and POLNO instead of Company, Lob, and PolicyNum, you must modify the HTML template to use those key names to work with that archive. Also, you may need to change the table headings, such as Customer Account, Customer Name, Location, and so on, to match those used in a particular archive.

The client program receives the request of type REQTYPE, RCP in this case, and runs the rules registered for this REQTYPE.

If the search returns more than 20 records, the system displays the first 20 along with a Next button which lets the user display the next 20 records, and so on. When there are less than 20 records left to display, the system displays those records without the Next button.

The RECORDS.HTM template contains a HTML form for doing this. Here is the example of this HTML form:

```
<!-- DCL BEGIN SECTION;NAME=MORERECORDS;IF MORERECORDS=YES; -->
<FORM METHOD=POST ACTION="#EXENAME,#" >
<INPUT VALUE="Next #MAXRECORDS,# matches" TYPE="submit">
<INPUT NAME="REQTYPE" value="SCH" TYPE="HIDDEN">
<INPUT NAME="USERID" value="#USERID,%s#" TYPE="HIDDEN">
<INPUT NAME="CONFIG" value="#CONFIG,%s#" TYPE="HIDDEN">
<INPUT NAME="FIELDS" value="Company,Lob,PolicyNum" TYPE="HIDDEN">
<INPUT NAME="PARTIALMATCH" value="YES" TYPE="HIDDEN">
<INPUT NAME="RESTART" value="ARCKEY" TYPE="HIDDEN">
<INPUT NAME="ARCKEY" value="#RECORDS15.ARCKEY,#" TYPE="HIDDEN">
<INPUT NAME="Company" value="#.Company,#" TYPE="HIDDEN">
<INPUT NAME="Lob" value="#.Lob,#" TYPE="HIDDEN">
<INPUT NAME="PolicyNum" value="#.PolicyNum,#" TYPE="HIDDEN">
<INPUT NAME="MAXRECORDS" value="#MAXRECORDS,#" TYPE="HIDDEN">
<INPUT NAME="LASTRECORD" value="#LASTRECORD,#" TYPE="HIDDEN">
<INPUT NAME="FIRSTRECORD" value="#FIRSTRECORD,#" TYPE="HIDDEN">
</FORM>
<!-- DCL END SECTION -->
```

This form is used only if there was a variable MORERECORDS in the attachment with the value YES. This form is presented by the browser as one button. When the user selects the Next ## matches button, the web server executes the client program specified in the ACTION= part of this HTML form.

The client program gets the request type SCH. This form is similar to regular Search form, with the exception of RESTART value. The restart value specifies for IDS rules that tell it where to resume searching. Generally, restart value works the same way as FIELDS. The value specifies comma-delimited archive table column names. For each of these names there should be a HTML variable with a corresponding value. In this case, restart is ARCKEY and there is ARCKEY= in the HTML form.

Note the RECORDS15.ARCKEY,# value for the restart position. It means get the value from the record 20. This number should be changed to the same number as the MAXRECORDS value.

The rules needed by IDS (contained in the docserv.xml file) to process the RCP request and post the results to the DOC client are:

```
<section name="ReqType:RCP">
    <entry name="function">atcw32->;ATCLogTransaction</entry>
    <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">dprw32->;DPRSetConfig</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
    <entry name="function">dprw32->;DPRGetRecipients</entry>
    <!-- -->
    <!-- -->
</section>
```

The processing of the RCP request creates following attachment variables:

- RESULTS - SUCCESS or error code
- ARCKEY - the archive key value for the current selected transaction
- RECORDS - number of recipients in the returned result set
- RECORDS#.RECIPIENT - name of the recipient. This line is repeated as many times as there are recipients
- RECORDS#.DESCRIPTION - description of the recipient (as defined in the Recip_Names control group). This line is repeated as many times as there are recipients

Setting Up the Recipient Page

The recipient selection HTML page is the page returned by client program after the processing of the RCP request. The client creates this page by processing the RECIPS.HTM HTML template. Again, the recipient selection page can contain any information the user might want, including links to other pages, links to email, and so on. The only part Documaker Bridge needs is the HTML form with these required fields:

- USERID
- REQTYPE
- ARCKEY

and these optional values:

- CONFIG
- PRTTYPE (defaults to PDF)

Here is the relevant part of the RECIPS.HTM template:

```
<html>
<head>
<base href="#BASELOCATION,#">
<meta http-equiv="keywords" content="FormMaker Retrieve">
<title> RETRIEVE RECORD </title>
```

```
</head>
<body bgcolor = "#FFFFFF" link=#0000ff vlink=#2525b5
alink=#ff0000 text="#000000" >
<IMG ALIGN=RIGHT TOP SRC="nlogo30s.gif" alt=fsilogo> <BR>
<H2><B> Document Retrieval  </B> </H2><BR>
<hr>
<BR>
<H5><B> <I> Selected Record: </I></B> </H5>
<TABLE BORDER=2 CELLPADDING=1 COLOR="blue">
<TR><TD><TH> <B> Customer Account </TH></TD>
    <TD> <B>  Customer Name </TD>
    <TD> <B>  Location </TD>
    <TD> <B>  Statement Date </TD>
</TR>
<TR>

<TD><TH> #Company,%s# </TH></TD>
<TD> #Lob,%s# </TD>
<TD> #PolicyNum,%s# </TD>
<TD> #RunDate,%s# </TD>
</TR>
</TABLE>
<hr>
<BR>
<FORM METHOD=POST ACTION="#EXENAME,#">
<INPUT NAME="USERID" VALUE="#USERID,%s#" TYPE="HIDDEN">
<INPUT NAME="DOCTYPE" VALUE="#DOCTYPE,%s#" TYPE="HIDDEN">
<INPUT NAME="REQTYPE" VALUE="PRT" TYPE="HIDDEN">
<INPUT NAME="CONFIG" VALUE="#CONFIG,%s#" TYPE="HIDDEN">
<INPUT NAME="ARCKEY" VALUE="#ARCKEY,%s#" TYPE="HIDDEN">
<INPUT NAME="Company" VALUE="#Company,%s#" TYPE="HIDDEN">
<INPUT NAME="Lob" VALUE="#Lob,%s#" TYPE="HIDDEN">
<INPUT NAME="PolicyNum" VALUE="#PolicyNum,%s#" TYPE="HIDDEN">
<INPUT NAME="RunDate" VALUE="#RunDate,%s#" TYPE="HIDDEN">
<INPUT NAME="PRINTPATH" VALUE="#PRINTPATH,%s#" TYPE="HIDDEN">
<B>Recipient:</B>
<SELECT NAME="recipient">
<!-- DCL BEGIN SECTION;NAME=RECORDS; -->
<OPTION VALUE="#RECIPIENT,%s#">#DESCRIPTION,#
<!-- DCL END SECTION -->
</SELECT>

<BR>
<B> Output file type:</B>
<SELECT NAME="PRTTYPE">
<OPTION> PDF
</SELECT>
<BR>
<BR>
<BR>
<INPUT TYPE=submit VALUE="Retrieve Document"><P>
</FORM>
<hr>
</html>
```

Again, variables in the file are archive-specific and must match the archive keys contained in the application index (APPIDX) for the archive. For example, if a particular Documaker archive used the keys ACCOUNT, NAME, and POLNO instead of Company, LOB, and PolicyNum, you must modify the HTML template to use those key names. Also, you may need to change the table headings (Customer Account, Customer Name, Customer Location, and so on) to match those used in a particular archive.

The client program receives the request of type REQTYPE—PRT in this case—and runs rules registered for this REQTYPE. The rules needed on the client to process the request are default rules. The rules to process the result posted from the Internet Document Server on the client are not the default rules. The following rules are used:

```
< ResType:PRT >
   function = ATCw32->ATCLoadAttachment
   function = ATCw32->ATCAppend2Attachment
   function = ircltw32->IRCResult
   function = ircltw32->IRCPrint
   function = ircltw32->IRCUnloadPage
```

The rules needed on the Internet Document Server (contained in the docserv.xml file) to process the PRT request and post the results to the DOC Client are:

```
<section name="ReqType:PRT">
    <entry name="function">atcw32->;ATCLogTransaction</entry>
    <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">dprw32->;DPRSetConfig</entry>
    <entry name="function">dprw32->;DPRInitLby</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
    <entry name="function">dprw32->;DPRPrintFormset</entry>
</section>
```

Processing the PRT request by IDS creates the following attachment variables:

- RESULTS - SUCCESS or error code
- REMOTEPRINTFILE - full file name of the created PDF file. The client program changes this name to be relative to the web server HTML contents directory.

Effective dates

When using Library Manager with IDS, include the following attachment variable on the RECIPS.HTM page so IDS can compare the form's effective date to the archived run date when you retrieve a form using Library Manager.

```
<FORM METHOD=POST ACTION="#EXENAME,#">
<INPUT NAME="USERID" VALUE="#USERID,%s#" TYPE="HIDDEN">
<INPUT NAME="DOCTYPE" VALUE="#DOCTYPE,%s#" TYPE="HIDDEN">
<INPUT NAME="REQTYPE" VALUE="PRT" TYPE="HIDDEN">
<INPUT NAME="CONFIG" VALUE="#CONFIG,%s#" TYPE="HIDDEN">
<INPUT NAME="ARCKEY" VALUE="#ARCKEY,%s#" TYPE="HIDDEN">
<INPUT NAME="Company" VALUE="#Company,%s#" TYPE="HIDDEN">
<INPUT NAME="Lob" VALUE="#Lob,%s#" TYPE="HIDDEN">
<INPUT NAME="PolicyNum" VALUE="#PolicyNum,%s#" TYPE="HIDDEN">
<INPUT NAME="RunDate" VALUE="#RunDate,%s#" TYPE="HIDDEN">
<INPUT NAME="PRINTPATH" VALUE="#PRINTPATH,%s#" TYPE="HIDDEN">
<INPUT NAME="ARCEFFECTIVEDATE" VALUE="#RunDate,%s#" TYPE="HIDDEN">
```

Setting Up the Printout Page

The printout page is returned by the client program after the processing of the PRT request. The client creates this page by processing the PRINTOUT.HTM template. Again, the printout page can contain any information you want, including links to other pages, links to email, and so on.

The example template has a link to the Adobe internet site for downloading Acrobat Viewer browser plug-in. The printout page does not display the Adobe Acrobat PDF file, it contains a link to the PDF file.

The following is the PRINTOUT.HTM template:

```
<html>
<head>
<base href="#BASELOCATION,#">
<meta http-equiv="keywords" content="FormMaker Retrieve">
<title> RETRIEVE RECORD </title>
</head>
<body bgcolor = "#FFFFFF" link=#0000ff vlink=#2525b5
<alink=#ff0000 text="#000000" >
<body>
<IMG ALIGN=RIGHT TOP SRC="nlogo30s.gif" alt=fsilogo> <BR>
<H2><B> Document Retrieval  </B> </H2><BR>
<hr>
<BR>
<H3><B> <I> Selected Record: </I></B> </H3>
<TABLE BORDER=2 CELLPADDING=1 COLOR="blue">
<TR><TD><TH> <B> Customer Account </TH></TD>
    <TD> <B>  Customer Name </TD>
    <TD> <B>  Location </TD>
    <TD> <B>  Statement Date </TD>
</TR>
<TR>
    <TD><TH> #Company,# </TH></TD>
    <TD> #Lob,# </TD>
    <TD> #PolicyNum,# </TD>
    <TD> #RunDate,# </TD>
</TR>
<P><BR><BR><BR>

</TABLE>
<A HREF="#PRINTFILE,#"> Click here to see the document </A>
<hr>
<A align=right HREF="http://www.adobe.com/prodindex/acrobat/
readstep.html">
Download Adobe&#174 Acrobat&#174 Reader Software </A>(Required to
view PDF files)
</A>
<A align=right HREF="http://www.adobe.com/prodindex/acrobat/
readstep.html">
<IMG ALIGN=LEFT TOP SRC="getacro.gif" alt="Download Adobe logo"> </A>
<P><BR>
<hr>

<!--DCL BEGIN SECTION;NAME=ERRORFILE; -->
<A HREF="#ERRORFILE,#"> Click here to see the errors </A>
```

```
<!-- DCL END SECTION -->

</body>
</html>
```

Again, variables on in the file are archive-specific and must match the archive keys contained in the application index (APPIDX) for the archive. For example, if a particular Documaker archive used the keys ACCOUNT, NAME, and POLNO instead of Company, LOB, and PolicyNum, the HTML template must be modified to use those key names to work with that archive.

Also, you may need to change the table headings (Customer Account, Customer Name, Customer Location, and so on) to match those used in a particular archive.

When the user clicks on the HTML link to the Adobe Acrobat PDF file, the web server sends the PDF file to the browser, and the browser automatically displays the file using the Adobe Acrobat Reader, provided the viewer has been properly installed.

Using the IDS HTML Templates

The Web Retrieval System includes a client-side base rule called IRCUnloadPage which you can use to return information to the web server. This rule lets you:

- Replace variables in an HTML file with their values before sending the information to the web server.
- Conditionally repeat sections of an HTML file to display multiple records

This topic discusses changes you must make to HTML files to take advantage of these features.

Note: Since HTML template files are not well-formed HTML files until *after* they have been processed by the Bridge, you must be very careful when you edit them with an HTML editor. Many HTML editors try to *fix* the templates, which can instead introduce problems. Be sure to disable any features in your HTML editor that automatically make corrections to the HTML page being edited.

As it processes the HTML file, the IRCUnloadPage rule performs variable replacement by searching for strings of the form:

```
#VARIABLE,FORMAT#
```

VARIABLE is the name of an element in the client's global variable list, and FORMAT is the format in which to display the contents of VARIABLE. Acceptable values for FORMAT include the values %s, WIDTH, HEXTIME, LTIME, DATE, and HTML. The effect of each of these codes is described below.

- Percent signs (%) will cause the data to be formatted according to C language formatting rules.
- The WIDTH=<value> code tells the system to left-justify the data in a field with a width of <value>. Setting the width to zero is the same as specifying %s in the FORMAT field. If the string contained in VARIABLE is longer than the specified width, the output will be truncated. Note that web browsers interpret all contiguous white space (spaces, tabs, and so on) as a single space. For the width to be preserved on the web page, then, the HTML <pre> and </pre> tags should surround the entire #VARIABLE,FORMAT# statement.
- The HEXTIME=value> code causes the data contained in VARIABLE to be treated as a hex format date/time string. This data is converted to the format given by <value>. For a list of supported formats, see Time Formats on page 54.
- The LTIME=<value> format tag is identical to the HEXTIME format tag with the exception that the input data is expected to be in decimal format.
- The DATE=<value> code causes the data contained in VARIABLE to be treated as a data/time string in YYYYMMDD format, such as 19981231. This data is converted to the format given by <value>. For a list of supported formats, see Date Formats on page 52.
- The HTML code causes HTML-reserved and nonprintable characters in the contents of VARIABLE to be replaced with their corresponding hexadecimal values.

Note that FORMAT is optional and defaults to %s. The comma, however, is not optional, so use #VARIABLE,# to specify default formatting.

The IRCUnloadPage rule can also repeat sections of an HTML file if you want to display an unknown number of records. To declare that a section repeats, use a statement (preferably within an HTML comment) of the form.

```
DCL BEGIN SECTION;NAME=<sectionname>;[NEVERSKIP];[IF
<variable>=<value>;]
```

When the IRCUnloadPage rule reads this declaration, it searches the base attachment for a section named *sectionname* (see the discussion of the base attachment file format above).

If the section name is not present in the attachment or the record count for the section is zero, the section will, by default, be skipped. You can override this behavior by specifying the NEVERSKIP option, in which case a single instance of the section is placed in the output stream. To signal the end of a repeated section, use the statement (again, preferably within an HTML comment):

```
DCL END SECTION
```

To conditionally process sections, use the IF directive. If the condition <variable>=<value> is true, then the section will be processed.

Within each section, IRCUnloadPage will perform replacement according to the rules set out above. For stem variables, lines to be replaced should have the format:

```
#MEMBERNAME,FORMAT#
```

where MEMBERNAME is the desired member of the stem variable (see again the discussion of the base attachment file format). For simple sectioned data, such as non-stem variables), lines to be replaced should have the format

```
#.,FORMAT#
```

Here is an example. Assume that the base attachment for a transaction contains…

```
USERID;JOHNDOE
RESULT;SUCCESS
POLICYNUM;XX1234567
DATETIME;B4B8702F
RECIPIENTS;3
RECIPIENTS1;CUSTOMER
RECIPIENTS1.COPIES;1
RECIPIENTS1.VIEWABLE;YES
RECIPIENTS2;AGENT
RECIPIENTS2.COPIES;1
RECIPIENTS2.VIEWABLE;YES
RECIPIENTS3;COMPANY
RECIPIENTS3.COPIES;2
RECIPIENTS3.VIEWABLE;NO
```

…and that the HTML source corresponding to this transaction type is…

```
Content-type:text/html

<html>
<head>
    <title>List of recipients requested by #USERID,%s#</title>
</head>
<body>
    <p>#RECIPIENTS,%s# recipients were found for policy
       #POLICYNUM,%s#, #USERID,%s#. They are listed below, along
        with the copy counts for each.
    </p>
    <hr>
    <ul>
    <!-- DCL BEGIN SECTION;NAME=RECIPIENTS;IF VIEWABLE=YES; -->
        <!-- Note that COMPANY copy is not viewable -->
        <li><pre>#.,WIDTH=20#          #COPIES,%s#</pre>
    <!-- DCL END SECTION -->
    </ul>
    <hr>
    <p>#DATETIME,HEXTIME=14%#</p>
</body>
</html>
```

The output from this example would be…

```
Content-type:text/html

<html>
<head>
    <title>List of recipients requested by JOHNDOE</title>
</head>
<body>
    <p>3 recipients were found for policy XX1234567, BOBAFETT.
      They are listed below, along with the copy counts for each.
    </p>
    <hr>
    <ul>
    <li><pre>CUSTOMER            1</pre>
    <li><pre>AGENT               1</pre>
    </ul>
    <hr>
    <p>01/28/1996</p>
</body>
</html>
```

NOTE: To view the above in a web browser, copy the lines from <html> to </html> and paste into a new file. Save the new file with the extension *html* or *htm* and use your browsers File, Open option to load the new file.

Nested sections are also supported. Note that the '.' applies to the current section *and* nesting level. Assume that the attachment contains...

```
EMPLOYEES;3
EMPLOYEES1;Bob Dobbs
EMPLOYEES1.DEPENDENTS;2
EMPLOYEES1.DEPENDENTS1;Bob's Wife
EMPLOYEES1.DEPENDENTS1.BIRTHDATE;19650303
EMPLOYEES1.DEPENDENTS2;Bob's Child
EMPLOYEES1.DEPENDENTS2.BIRTHDATE;19850712
EMPLOYEES2;E.A. Robinson
EMPLOYEES3;H.R. Puffin
EMPLOYEES3.DEPENDENTS;2
EMPLOYEES3.DEPENDENTS1;Jimmy
EMPLOYEES3.DEPENDENTS1.BIRTHDATE;19520930
EMPLOYEES3.DEPENDENTS2;Freddy the Flute
```

```
EMPLOYEES3.DEPENDENTS2.BIRTHDATE;00000000
```

...and that the HTML template looks like this:

```
<html>
<head>
    <title>Some Employees</title>
</head>
<body>
    <p>Here is a list of #EMPLOYEES,%s# employees and their
        dependents.
    </p>
    <table>
        <tr><th>Employee Name</th>
            <th>Dependent Name</th>
            <th>Dependent Birthdate</th>
        </tr>
        <!-- DCL BEGIN SECTION;NAME=EMPLOYEES; -->
            <tr><td>#.,%s#</td></tr>
            <!-- DCL BEGIN SECTION;NAME=DEPENDENTS; -->
                <tr>
                    <td></td>
                    <td>#.,%s#</td>
                    <td>#BIRTHDATE,%s#</td>
                </tr>
            <!-- DCL END SECTION -->
        <!-- DCL END SECTION -->
    </table>
</body>
</html>
```

The data sent to the web server by the IRCUnloadPage rule will look like...

```
Content-type:text/html

<html>
<head>
    <title>Some Employees</title>
</head>
<body>
    <p>Here is a list of 3 employees and their
        dependents
    </p>
    <table cellpadding=7>
        <tr><th>Employee Name</th>
            <th>Dependent Name</th>
            <th>Dependent Birthdate</th>
        </tr>
            <tr><td>Bob Dobbs</td></tr>
                <tr>
                    <td></td>
                    <td>Bob's Wife</td>
                    <td>19650303</td>
                </tr>
                <tr>
                    <td></td>
                    <td>Bob's Child</td>
```

```
                        <td>19850712</td>
                    </tr>
                <tr><td>E.A. Robinson</td></tr>
                <tr><td>H.R. Puffin</td></tr>
                    <tr>
                        <td></td>
                        <td>Jimmy</td>
                        <td>19520930</td>
                    </tr>
                    <tr>
                        <td></td>
                        <td>Freddy Flute</td>
                        <td>00000000</td>
                    </tr>
        </table>
    </body>
    </html>
```

Note the use of 00000000 as the birth date for Freddy Flute. If a variable is not present to replace a parameter in an HTML template, the IRCUnloadPage rule assumes the parameter should be left as is. If the EMPLOYEES3.DEPENDENTS2.BIRTHDATE variable were left out of the attachment, the corresponding output would have been…

```
<tr>
    <td></td>
    <td>Freddy Flute</td>
    <td>#BIRTHDATE,%s#</td>
</tr>
```

…which is not the desired result. Note also the use of the '.' operator. In the section named EMPLOYEES, '.' represents the variable EMPLOYEESx (where x is replaced by the record number). In the section named DEPENDENTS, '.' represents the variable EMPLOYEESx.DEPENDENTSy (where x and y are replaced by the corresponding record numbers). Because DEPENDENTS is a subsection of EMPLOYEES, 'EMPLOYEES.' is prepended to all variables in the DEPENDENTS section.

At times, it you may need to have constant values added to the attachment for a given type of (or for every) transaction. For example, you may want the name of the machine on which the Web Retrieval Server is running. To add variables to the attachment list, use one of these INI options...

```
< Attach:<request type> >
    VARIABLE = VALUE
```

or

```
< Attach:DEFAULT >
    VARIABLE = VALUE
```

...and add the rule ATCXXX->Append2Attachment to the rule list for the transactions to which these values should apply. Note that the variables are added to the input attachment during IRP_MSGRUNF processing. This means that the client should run the rule during result processing rather than request processing. It also means that the rule must be run before any rules that use the appended values.

HTML Forms

The Web Retrieval client retrieves a large amount of its input from HTML forms. These forms let the HTML author specify variable name/value pairs via the INPUT tag. These variables are converted without modification by the Web Retrieval client into attachment variables. As an example, specifying…

```
< FORM METHOD=POST ACTION="/cgi-bin/dcltw32.exe" >
    <INPUT NAME="REQTYPE" value="ADM" TYPE="HIDDEN">
.
.
.
```

…tells the Web Retrieval client to create an attachment variable named REQTYPE and give it the value ADM. Attachment variables created this way will be passed to the Web Retrieval server to control transaction processing.

Required Fields

There is a single INPUT field, REQTYPE, which is required for every transaction. The REQTYPE field identifies the request type and allows the system to locate the rules and resources necessary for processing the transaction. Individual rules will have their own specific INPUT field requirements.

Removing HTML Pages

The example HTML pages included with the system are designed so that if you receive an error when producing the PDF file, you will see the link to the error messages on the same page (to see how this works, rename the logo files in the library you are using).

If there are no errors, this means users see an extra HTML page. If you want to remove this page, just put an extra tag into <HEAD> section of the PRINTOUT.HTM file.

Here is the tag:

```
<meta http-equiv="Refresh"
CONTENT="0;URL=#BASELOCATION,#/#PRINTFILE,#">
```

This tag redirects the page to the next screen, which is a PDF file. When you use this tag, the browser shows the printout page for a moment and then jumps to the Acrobat Reader. The zero (0) in this tag is the number of seconds the printout page will appear, before being redirected. You may want to change the printout page to say something like:

Please, wait a moment.

Date Formats

Date formats consist of these components, in this order:

```
(Format type)(Separator)(Year size)
```

Component	Description
Format type	1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,or K. The default format type is 1. You must include a format type if you want to specify a separator or a year size.

Component	Description
Separator	For the separator character, you can enter a backslash (/), a dash (-), a period (.), a comma (,), or B (or b), which indicates a blank space. You should only enter separator characters for format types which include separators (see the table of format types below). If the format type does not include separators, such as format type C, the system ignores any separator character you enter. The default separator is a backslash (/).
Year size	For the year size, you can specify either 2 (06) or 4 (2006) to indicate a two- or four-digit year. With the year 2000 approaching, we recommend that you use four-digit years. DAL functions will use a four-digit year unless the format or the input data specifies otherwise. For example, if you enter 1/2, you specify date format 1 and a two-digit year, such as 02/17/06.

Here is a discussion of the various formats you can use:

Format	Date Order	Description
1	MM/DD/YY	Month-Day-Year with leading zeros (02/17/2006)
2	DD/MM/YY	Day-Month-Year with leading zeros (17/02/2006)
3	YY/MM/DD	Year-Month-Day with leading zeros (2006/02/17)
4	Month D, Yr	Month name-Day-Year with no leading zeros (February 17, 2006)
5	bM/bD/YY	Month-Day-Year with spaces instead of leading zeros (2/17/2006)
6	bD/bM/YY	Day-Month-Year with spaces instead of leading zeros (17/ 2/2006)
7	YY/bM/bD	Year-Month-Day with spaces instead of leading zeros (2006/ 2/17)
8	M/D/YY	Month-Day-Year with no leading zeros (2/17/2006)
9	D/M/YY	Day-Month-Year with no leading zeros (17/2/2006)
A	YY/M/D	Year-Month-Day with no leading zeros (2006/2/17)
B	MMDDYY	Month-Day-Year with no separators (02172006)
C	DDMMYY	Day-Month-Year with no separators (17022006)
D	YYMMDD	Year-Month-Day with no separators (20060217)
E	MonDDYY	Month abbreviation-Day-Year with leading zeros (Feb172006)
F	DDMonYY	Day-Month abbreviation-Year with leading zeros (17Febl2006)
G	YYMonDD	Year-Month abbreviation-Day with leading zeros (2006Feb17)

Format	Date Order	Description
H	day/YY	Day of year (counting consecutively from January 1)-Year (48/2006)
I	YY/day	Year-Day of Year (counting consecutively from January 1—often called the Julian date format) (2006/48)
J	D Month, YYYY	Day-Month name-Year (17 February, 2006)
K	YYYY, Month D	Year-Month name-Day (2006, February 17)

NOTE: Month abbreviations consist of the first three characters of the month's name. Months with four-character names, such as June, are not abbreviated.

Here are some examples:

Format	Description	Result
1	Format type 1	12/18/06
1-	Format type 1 with dashes (-) as the separator characters	12-18-06
1/2	Format type 1 with backslashes (/) as the separator characters and a two-digit year	12/18/06
14	Format type 1 with a four-digit year (no separator specified but the format type includes separators so the default separator (/) will be used	12/18/06
B4	Format type B with a four-digit year (no separator specified and the format type does not include separators, so none will be included	1218106

Time Formats

Times can be entered in several formats. The time formats are explained in the table below.

Format	Time Segments	Description
1	HH:MM:SS	Time is based on a 24 hour system. This is frequently referred to as "military time". The 24 hour system is the default format. Example: 14:18:23
2	HH:MM:SS XM	Time is based on a 12 hour system. AM or PM is given. Example: 02:18:23PM
3	HH:MM	Time is based on a 24 hour system. Seconds are not given. Example: 14:18

Format	Time Segments	Description
4	HH:MM XM	Time is based on a 12 hour system. Seconds are not given. AM or PM is given. Example: 02:18PM

Chapter 2

Documaker Bridge Rules

The Documaker Bridge includes rules you can use to control what happens to data moving across the bridge. These rules are listed on the following pages and then discussed in alphabetical order.

These rules run on all supported platforms except where noted. The rule names are case sensitive.

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

List of Rules

The following rules can only be run when you use the Documaker Bridge. The rules are in alphabetical order, as shown below:

- DPRLoadedXML2Formset on page 184
- DPRLoadFAPImages on page 185
- DPRLoadImportFile on page 186
- DPRLoadXMLAttachment on page 188
- DPRLoadXMLFormset on page 189
- DPRLocateOneRecord on page 190
- DPRLockWip on page 191
- DPRLog on page 193
- DPRLogin on page 194
- DPRLoginUser on page 196
- DPRMail on page 197
- DPRMapRecipData on page 199
- DPRModifyUser on page 201
- DPRModifyWipData on page 205
- DPRPatchLevel on page 207
- DPRParseRecord on page 208
- DPRPrint on page 211
- DPRPrintDpw on page 219
- DPRPrintFormset on page 221
- DPRProcessTemplates on page 224
- DPRRenameVars on page 226
- DPRRetFromUserDict on page 227
- DPRRetrieveDPA on page 230
- DPRRetrieveFormset on page 231
- DPRRotateFormsetPages on page 234
- DPRSearch on page 235
- DPRSearchLDAP on page 237
- DPRSearchWip on page 246
- DPRSendFormsetXML on page 251
- DPRSendMultiFiles on page 252
- DPRSendVersion on page 253
- DPRSet2ImageScope on page 254

NOTE: The Documaker Bridge rules load the FXR and FORM.DAT files once and stores them in cache to speed performance.

The modify date of the FORM.DAT file is checked and the file is reloaded if the modify date change. This means IDS does not have to restart if the FORM.DAT file was changed.

The FXR file caching is done the same way as FAP file caching and it does not check file dates on disk. If you need to disable FXR file caching, disable FAP file caching.

DPRAddBlankPages

Use this rule to add blank or filler pages into a form set. You add these pages to make sure each physical printed page has a front and back. This lets you change a simplex form set or a form set which contains both simplex and duplex forms into a fully duplexed form set.

For instance, you can use this to make it easier to add OMR marks, which are often printed on the back, to simplex forms. Another use is to create PDF files for form sets which contain both simplex and duplex forms but which print as a fully duplexed form set.

Syntax

```
long _DSIAPI DPRAddBlankPages ( 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

This rule assumes that the form set has been loaded by the Documaker Bridge into the DSI variable, DPRFORMSET.

If you are using this rule with a different bridge, you may need to specify a different DSI variable that contains the form set. If you want the system to use a specific FAP file for the filler pages, the name of that FAP file must follow the form set variable name when you specify the rule. Here is an example:

```
function = dprw32->DPRAddBlankPages,DPRFORMSET,FAPFile
```

Omit the FAP file's path and extension.

Here is a table which shows when blank pages will be added, based on the duplex setting of the two current pages and the duplex setting of the next page. *Blank* means a blank page will be added, *As is* means no blank page is needed and the form will be left as is.

	And the next page is					
If the current page is	**Unknown**	**Front**	**Back**	**None**	**Short**	**Rolling**
Unknown	Blank	Blank	As is	Blank	Blank	Blank
None	Blank	Blank	As is	Blank	Blank	Blank
Front	Blank	Blank	As is	Blank	Blank	As is
Short	Blank	Blank	As is	Blank	Blank	As is
Rolling (Front)	Blank	Blank	As is	Blank	Blank	As is
Back	As is	As is	Blank	As is	As is	As is
Rolling (Back)	As is	As is	Blank	As is	As is	As is

NOTE: You can also add blank or filler pages using custom code or a DAL script which includes the AddBlankPages function. See the DAL Reference for more information on the AddBlankPages DAL function.

The API to call from custom code is as follows:

```
DWORD _VMMAPI FAPAddBlankPages(
                VMMHANDLE objectH,      /* form set or form handle */
                 char FAR * imagename)   /* if NULL, "Blank Page" */
```

If the image name is NULL, a blank page is created when a filler page is needed. If the image name is not NULL, the image name is loaded when a filler page is needed. If you include an image name, include only the name of the FAP file—omit the path and file extension.

See also DPRDelBlankPages on page 102

DPRAddLogo

Use this rule to add a logo to a document retrieved from an archive. The logo is not stored with the original document. Instead, it is added when the document is retrieved from archive and only appear in the PDF file that is created from the archive.

NOTE: You can add logos and text. Logos are graphics and may obscure overlapping objects when viewed in Acrobat Reader version 3.x. This is not a problem if you use Acrobat Reader version 4.x. Text displays properly in all versions of Acrobat Reader.

Keep in mind there is no support for transparency in multi-color bitmaps or the z-ordering of FAP objects. For best results, use a mono-color bitmap.

Syntax

```
long _DSIAPI DPRAddLogo ( 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

The DPRAddText and DPRAddLogo rules are located in the DPRW32.DLL and run on MSG_RUNF. Here is an example from the DOCSERV.INI file of the rule list which shows these rules:

```
< ReqType:MTC >
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = mtcw32->MTCLoadFormset
function = dprw32->DPRRotateFormsetPages
function = dprw32->DPRAddLogo
function = dprw32->DPRAddText
function = mtcw32->MTCPrintFormset
```

NOTE: When you use this rule with any bridge other than the Printstream Bridge, you must include the name of the form set, as shown here:

```
function = dprw32->DPRAddLogo,DPRFORMSET
```

If you omit the form set, the system assumes MTCFORMSET is its name. You cannot use this rule with the TIFF Bridge.

INI options

To add a logo, you must add a AddLogo control group to the master resource INI file. This control group will have these options:

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.

NOTE: You can also use DPRAddLogo functionality with the DPRPrint rule. For more information, see Adding Logos when using DPRPrint on page 213.

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

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

DPRAddText

Use this rule to add text to a document retrieved from an archive. The text is not stored with the original document. Instead, it is added when the document is retrieved from archive and only appear in the PDF file that is created from the archive.

NOTE: You can add two types of files: logos and text. Logo are graphics and may obscure overlapping objects when viewed in Acrobat Reader version 3.x. This is not a problem if you use Acrobat Reader version 4.x. Text displays properly in all versions of Acrobat Reader.

Syntax

```
long _DSIAPI DPRAddText ( 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

The DPRAddText and DPRAddLogo rules are located in the DPRW32.DLL and run on MSG_RUNF. Here is an example from the DOCSERV.INI file of the rules list which shows these rules:

```
< ReqType:MTC >
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = mtcw32->MTCLoadFormset
function = dprw32->DPRRotateFormsetPages
function = dprw32->DPRAddLogo
function = dprw32->DPRAddText
function = mtcw32->MTCPrintFormset
```

NOTE: When you use this rule with any bridge other than the PrintStream Bridge, you must include the name of the form set, as shown here:

```
function = dprw32->DPRAddText,DPRFORMSET
```

If you omit the form set, the system assumes MTCFORMSET is its name. You cannot use this rule with the TIFF Bridge.

INI options

To add text, you must add an AddText control group to the INI settings for the master resource INI file. This control group has these options:

Option	Description
Text	The string you want to appear.
FontID	The font ID that identifies the font you want to use. This ID also specifies the point size of the font.
Top	Contains the top coordinate (position) of the text in FAP units (2400 units per inch)
Left	Contains the left coordinate (position) of the text in FAP units (2400 units per inch)
Pages	(Optional) The default is to add the text on all pages. Use this option to set the number of pages on which you want the text to appear. If you set this option to 1, the system adds the text to the first page only.
Angle	(Optional) The default is to display the text at a zero (0) degree angle. You can also enter 90, 180, and 270.
Color	(Optional) Default is to display the text 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.

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

```
< AddText >
    Text   = SAMPLE FORM
    FontID = 11020
    Top    = 12000
    Left   = 12000
    Color = 255
```

DPRAddToUserDict

Use this rule to add words into the user dictionary.

Syntax

```
long _DSIAPI DPRAddToUserDict ( DSIHANDLE hdsi,
                                char * pszParms,
                                unsigned ulMsg,
                                unsigned ulOptions )
```

Parameters

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

Attachment variables

Variable	Description	
AddLine	A line of words you want to add to the user dictionary. Separate the words with commas.	
LanguageOpt	The language selection. The default is US English. You can choose from these languages and dictionaries:	
	Danish	“ssceda.tlx,ssceda2.clx”
	Dutch	“sscedu.tlx,sscedu2.clx”
	Finnish	“sscefi.tlx,sscefi2.clx”
	French	“sscefr.tlx,sscefr2.clx”
	German	“sscege.tlx,sscege2.clx”
	Italian	“ssceit.tlx,ssceit1.clx”
	Norwegian	“sscenb.tlx,sscenb2.clx”
	Portuguese_Brazil	“sscepb.tlx,sscepb2.clx”
	Portuguese	“sscepo.tlx,sscepo2.clx”
	Spanish	“sscesp.tlx,sscesp2.clx”
	Swedish	“sscesw.tlx,sscesw2.clx”
	UK English	“sscebr.tlx,sscebr2.clx”
	US English	“ssceam.tlx,ssceam2.clx”
UserDict	The name of the user dictionary. The default is user.tlx.	

Attachment outputs

None

INI options

You can use these INI options with this rule:

```
< Spell >
    LanguageOpt    =
    UserDict       =
    UserDictPath   =
```

Option	Description
LanguageOpt	Enter the language option. The default is US English. You can choose from these languages and dictionaries: Danish “ssceda.tlx,ssceda2.clx” Dutch “sscedu.tlx,sscedu2.clx” Finnish “sscefi.tlx,sscefi2.clx” French “sscefr.tlx,sscefr2.clx” German “sscege.tlx,sscege2.clx” Italian “ssceit.tlx,ssceit1.clx” Norwegian “sscenb.tlx,sscenb2.clx” Portuguese_Brazil “sscepb.tlx,sscepb2.clx” Portuguese “sscepo.tlx,sscepo2.clx” Spanish “sscesp.tlx,sscesp2.clx” Swedish “sscesw.tlx,sscesw2.clx” UK English “sscebr.tlx,sscebr2.clx” US English “ssceam.tlx,ssceam2.clx”
UserDict	Enter the name of the user dictionary. The default is user.tlx.
UserDictPath	Enter the path to the user dictionary. The default is the current working directory.

Errors

Error	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRDelFromUserDict on page 107

DPRRetFromUserDict on page 227

DPRSpellCheck on page 257

DPRAddWipRecord

Use this rule to take an existing form set and save it to a WIP record. It is equivalent to the IPPAddWIP rule. This rule automatically sets the CreateTime and ModifyTime.

Syntax

```
long _DSIAPI DPRAddWipRecord ( DSIHANDLE hdsi,
                               char * pszParms,
                               unsigned long  ulMsg,
                               unsigned long  ulOptions )
```

Parameters

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

Attachment variables

This rule expects this IDS attachment variable:

Variable	Description
DPRFormset	Passes the form set handle.

This rule expects these attachment variables:

Variable	Description
Unique	If Yes, the rule checks to see if the record exists. If it does not exist, it adds it. If No, it adds it without checking.
UserID	If the input fields do not include CurrUserID and OrigUserID, UserID is used.
(field names)	The fields are defined in the DFD file. To match a record, Key1, Key2, KeyID and RecType are required (DOC_TAG).

Attachment outputs

This rule provides these output attachment variables:

Variable	Description
RecordID	The record ID.
RECNUM or/ and UNIQUE_ID	The record ID as defined in the WIP.DFD file.

INI options

You can use these INI options:

```
< WIPData >
    File =
    Path =
```

Option	Description
File	Enter the name of the WIP file.
Path	Enter the path to the WIP file.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0017	Cannot locate DSI variable #VARIABLE,#.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0036	The DSI variable #VARIABLE,# does not contain a valid FAP form set.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDelMultiWipRecords on page 109

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRUpdateWipRecords on page 287

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRWipTableParms on page 298

DPRApproveWipRecords

Use this rule to approve or reject all records in the WIP file which have a status of WIP.

Syntax

```
long _DSIAPI DPRApproveWipRecords ( DSIHANDLE hdsi,
                                    char * pszParms,
                                    unsigned long  ulMsg,
                                    unsigned long  ulOptions )
```

Parameters

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

Attachment variables

This rule expects these attachment variables:

Variable	Description
UserID	The ID of the queue name for the user
Status	The new status (Approve or Reject)

INI options

You can use these INI options with this rule:

Option	Control group	Description
File	WIPData	Specifies the name of the WIP file.
Path	WIPData	Specifies the path to the WIP file.
File	UserInfo	Specifies the name of the userinfo file.
Path	UserInfo	Specifies the path to the userinfo file. If omitted, the system adds USERID in the user list.
WIP	Status_CD	Specifies the WIP status code.

Here is an example:

```
< WIPData >
   File     = WIP
   Path     = mstrres\sampco\wip
   MaxWIPRecords = 200
< UserInfo >
   File     = userinfo
   Path     = mstrres
< Status_CD >
   WIP      = W
   Approve = AP
   Reject   = RJ
```

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0006	The virtual memory management API #APINAME,# failed.
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also DPRCheckWipRecords on page 88

DPRGetWipList on page 150

DPRGetWipFormset on page 153

DPRGetWipRecipients on page 156

DPRSearchWip on page 246

DPRUpdateWipRecords on page 287

DPRArchiveFormset

Use this rule to send a form set to Documaker archive.

Syntax

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

Parameters

Parameter	Description
DSIHANDLE hdsi	pointer to the rules data
char * pszParms	pointer to rule parameter string which may or may not contain the DSI variable name FormsetName that stores form set handle.
ULONG ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
ULONG ulOptions	options

This rule finds the form set by locating the DSI variable pFormName, where pFormName is from the first input in the input parameter string. The default is DPRFORMSET.

This rule needs input attachments (fields=values) to create the archive record for the archived. Keep in mind that the fields must be the same as those defined in the APPIDX.DFD file.

This rule unloads the form set into temporary files, such as the POL file, NA file, and PACKAG file, along with attached files in the package. After the form set is archived, the temporary files are removed, unless you set the DeleteFiles option to No.

You tell the system whether you want the system to archive to a file or database using the ArchiveMem option. The system creates a semaphore file to block access attempts until the archival is complete.

NOTE: This rule lets you map fields from a WIP record to the Archive index record using the AFEWIP2ArchiveRecord control group. Please refer to the Documaker Server System Reference for information on how you can use the AFEWIP2ArchiveRecord control group.

Attachment variables

This rule expects these attachment variables:

Variable	Description
FormsetName	The DSI variable name from pszParms. The default is DPRFORMSET.
FieldNames	Enter the value of the field to provide information for the form set that is to be archived. The field names should be the same as those in APPIDX.DFD.

INI options

You can use these INI options with this rule:

```
< ArcRet >
    Appidx  = mstrres\sampco\arc\appidx
    ArcPath = mstrres\sampco\arc\
    CarFile = mstrres\sampco\arc\archive
    Catalog = mstrres\sampco\arc\catalog
    CarPath = mstrres\sampco\arc\
< Status_CD >
    Archive = AR
< Debug >
    DeleteFiles = Yes
< Archival >
    ArchiveMem = Yes
```

Control group	Option	Description
ArcRet	Appidx	Enter the path for the application index file, such as mstrres\sampco\arc\appidx.
	ArcPath	Enter the path for the archive, such as mstrres\sampco\arc\
	CARFile	Enter the name and path for the CAR file, such as mstrres\sampco\arc\archive
	Catalog	Enter the name and path for the catalog file, such as mstrres\sampco\arc\catalog
	CARPath	Enter the path for the CAR file, such asmstrres\sampco\arc\
Status_CD	Archive	The default is AR.
Debug	DeleteFiles	Enter Yes if you want the system to remove the POL, NA, and PKG files. Enter No to retain these files. The default is Yes.
Archival	ArchiveMem	Enter Yes to archive to a database. Enter No to archive to a file. The default is No.

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0017	Cannot locate DSI variable #VARIABLE,#.
DPR0022	Cannot add the variable #VARIABLE,# to the attachment list.
DPR0036	The DSI variable #VARIABLE,# does not contain a valid FAP form set.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.
DPR0042	Failed to unload File #FILE,# in #LOCATION,#.
DPR0043	Failed to DBQueryFormatInfo from #FILE,#.

Message	Description
DPR0044	Failed to DBInitializeFile #FILE,#.
DPR0045	Failed to DBOpen #FILE,#.
DPR0046	Failed to UTLLockARC #FILE,#.
DPR0047	Failed to ArcInit #FILE,#.
DPR0048	Failed to ArcArchiveDataFile #FILE,#.

DPRAssignWipRecord

Use this rule to assign a new user ID to a record. It is equivalent to the IPPAssignWIP rule.

Syntax

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

Parameters

Parameter	Description
DSIHANDLE hdsi	pointer to the rules data
char * pszParms	pointer to rule parameter string which may or may not contain the DSI variable name FormsetName that stores form set handle.
ULONG ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
ULONG ulOptions	options

You can assign a WIP record to someone using...

- Record IDs. In this case, the input attachment variable RECORD is required. If it does not exist, the system searches RECNUM for code base or the UNIQUEID for an SQL database. If no ID is found, the system goes to the next record. If none are found, it search for fields.
- Fields. The system searches for the fields defined in DOC_TAG to match a record. For instance: Key1+Key2+KeyId+RecType.

The system automatically adds FromUserID, CurrUserID, and FromTime to the record for update.

Attachment variables

This rule expects these attachment variables:

Variable	Description
AssignUserID	Enter the user ID you want to assign the record to.
RecordID	Enter the record ID. You can define it as the RECNUM or UNIQUEID in your DFD definition.
AssignDesc	Optional. Enter the description to add or replace. (IPPWIP users can no longer use the attachment variable Desc because Desc may be a field as defined in the WIPDFD file.
(field names)	Enter the appropriate value to match a record, Key1, Key2, KeyID, and RecType are required. See the definition of DOC_TAG.in the WIPDFD file.

INI options

You can use these INI options:

```
< WIPData >
    File =
    Path =
```

Option	Description
File	Enter the name of the WIP file.
Path	Enter the path to the WIP file.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRAddWipRecord on page 71

DPRDeleteWipRecord on page 105

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRUpdateWipRecords on page 287

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRWipTableParms on page 298

DPRBatchArchive

You can use this rule to archive one or more transactions. This rule performs a function similar to that of the GenArc program. The primary use for this rule is to archive data created on a platform that does not support archive, such as when IDS runs Documaker on a platform where Documanage archive does not run.

You can set up IDS as a client to another IDS on a platform where Documanage archive is supported. In this case the NEWTRN.DAT, NAFILE.DAT, and POLFILE.DAT files are sent as attachments to the second IDS and this rule archives the data.

Syntax

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

Parameters

Parameter	Description
DSIHANDLE hdsi	pointer to the rules data
char * pszParms	pointer to rule parameter string which may or may not contain the DSI variable name FormsetName that stores form set handle.
ULONG ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
ULONG ulOptions	options

NOTE: You can use this rule with the ATCReceiveFile rule which lets IDS act as an IDS client and send a request to a second IDS.

Attachment variables

This rule expects these attachment variables:

Variable	Description
NEWTRN	The name of the input NEWTRN.DAT file, with one record for each transaction that needs to be archived.
NAFILE	The name of the input NAFILE.DAT file.
POLFILE	The name of the input POLFILE.DAT file.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate attachment variable #VARIABLE#
DPR0045	Failed to DBOpen file #FILE#.
DPR0060	Cannot open file #FILE#.

DPRBuildGroupList

Use this rule to build a rowset of matching Group1/Group2 groups for the form set specified by the CONFIG attachment variable. This is useful when you are creating drop down options for Key1/Key2 for a configuration.

Syntax

```
long _DSIAPI DPRBuildGroupList ( DSIHANDLE hdsi,
                                 char * pszParms,
                                 unsigned long  ulMsg,
                                 unsigned long  ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
CONFIG	The configuration value in the DAP.INI file from which you want the rule to return a group list rowset.

Attachment outputs

Variable	Description
GROUPS	An XML rowset containing the Group1/Group2 combinations for the form set.
RESULTS	Success or failure

Example

Here is an example:

```
<ROWSET NAME="GROUPS">
<ROW NUM="1">
<VAR NAME="GROUP1">AUTO</VAR>
<VAR NAME="GROUP2">LOB</VAR>
</ROW>
<ROW NUM="2">
<VAR NAME="GROUP1">AUTO</VAR>
<VAR NAME="GROUP2">APPLICATION</VAR>
</ROW>
<ROW NUM="3">
<VAR NAME="GROUP1">AUTO</VAR>
<VAR NAME="GROUP2">POLICY</VAR>
</ROW>
<ROW NUM="4">
<VAR NAME="GROUP1">AUTO</VAR>
<VAR NAME="GROUP2">CORRESPONDENCE</VAR>
</ROW>
<ROW NUM="5">
```

```
<VAR NAME="GROUP1">GENERAL LIABILITY</VAR>
<VAR NAME="GROUP2">LOB</VAR>
</ROW>
<ROW NUM="6">
<VAR NAME="GROUP1">GENERAL LIABILITY</VAR>
<VAR NAME="GROUP2">APPLICATION</VAR>
</ROW>
<ROW NUM="7">
<VAR NAME="GROUP1">GENERAL LIABILITY</VAR>
<VAR NAME="GROUP2">POLICY</VAR>
</ROW>
<ROW NUM="8">
<VAR NAME="GROUP1">GENERAL LIABILITY</VAR>
<VAR NAME="GROUP2">CORRESPONDENCE</VAR>
</ROW>
<ROW NUM="9">
<VAR NAME="GROUP1">PROPERTY</VAR>
<VAR NAME="GROUP2">LOB</VAR>
</ROW>
<ROW NUM="10">
<VAR NAME="GROUP1">PROPERTY</VAR>
<VAR NAME="GROUP2">APPLICATION</VAR>
</ROW>
<ROW NUM="11">
<VAR NAME="GROUP1">PROPERTY</VAR>
<VAR NAME="GROUP2">POLICY</VAR>
</ROW>
<ROW NUM="12">
<VAR NAME="GROUP1">PROPERTY</VAR>
<VAR NAME="GROUP2">CORRESPONDENCE</VAR>
</ROW>
<ROW NUM="13">
<VAR NAME="GROUP1">INDIVIDUAL</VAR>
<VAR NAME="GROUP2">POLICY</VAR>
</ROW>
</ROWSET>
```

DPRCheck

Use this rule to check for the existence of WIP and archived records and return the total number of records found in both WIP and archive.

Syntax

```
long _DSIAPI DPRCheck ( DSIHANDLE hdsi,
                        char * pszParms,
                        ULONG  ulMsg,
                        ULONG long  ulOptions )
```

Parameters

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

This rule expects these input attachments:

Variable	Description
LOGINRESULT	(Optional) SUCCESS to continue or FAILURE to stop. Used when the rule follows others.
USERID	The ID of the queue name for user.
PARTIALMATCH	(Optional) If Yes, the rule conducts a partial match for the search values provided for this variable. The default is No.
CASESENSITIVE	(Optional) If Yes, the rule conducts a case sensitive search, otherwise using uppercase values. This variable overwrites the CaseSensitiveKeys INI option. The default is No.
TABLEINIGROUP	(Optional) The name of the INI control group to get the application index table name from. The default is ArcRet.
TABLEINIOPTION	(Optional) The name of the INI option to get the application index table name from. The default is AppIdx.
FIELDNAME	One or more fields as defined in WIP DFD and archive DFD. Search values are used to match records. At least one field is required.
CHECKINARC	(Optional) If Yes, the rule searches archived records. This variable overwrites the CheckInArc INI option. The default is No.

Returns

This rule returns these output attachments:

Variable	Description
RECORDS	Total found records from WIP and ARC.
RESULTS	SUCCESS or FAILIURE

INI options

Use these INI options with this rule:

```
< Control >
    CaseSensitiveKeys  = No
    CheckInArc         = No
< WIPData >
    MaxWIPRecords      = 200
    File               =
    Path               =
< ArcRet >
    MaxRecords         = 200
    AppIdx             = mstrres/formmaker/arc/appidx
    ArcPath            = mstrres/formmakerformmaker/arc/
    CARFile            = mstrres/formmaker/arc/archive
    Catalog            = mstrres/formmaker/arc/catalog
    CARPath            = mstrres/formmaker/arc/
    AppIdxDFD          = mstrres/formmaker/deflib/appidx.dfd
< MasterResource >
    DefLib             = mstrres/formmaker/deflib
```

Option	Description
Control control group	
CaseSensitiveKeys	Enter Yes if the keys are case sensitive. When keys are not case sensitive, the system expects the fields to be uppercase in the database index. If you use case sensitive keys, you have to enter the data on the Archive/ Retrieval window just as it appears in the archive file. The default is No.
CheckInArc	Enter Yes to search archived records The default is No.
WIPData control group	
MaxWIPRecords	Enter the maximum number of WIP records to return. The default is 200.
File	Enter the name of the WIP file.
Path	Enter the complete path to the WIP file.
ArcRet control group	
MaxRecords	Enter the maximum number of archive records to return. The default is 200.
AppIdx	Enter the name and path for the AppIdx file. Here is an example: `mstrres/formmaker/arc/appidx`
ArcPath	Enter the path to the archive files. Here is an example: `mstrres/formmaker/arc/`
CARFile	Enter the name of the archive file. Here is an example: `mstrres/formmaker/arc/archive`

Option	Description
Catalog	Enter the name and path for the catalog file. Here is an example: `mstrres/formmaker/arc/catalog`
CARPath	Enter the path for the archive file. Here is an example: `mstrres/formmaker/arc/`
AppIdxDFD	Enter the name and path for the AppIdxDFD file. Here is an example: `mstrres/formmaker/deflib/appidx.dfd`
MasterResource control group	
DefLib	Enter the path to the DefLib directory. Here is an example: `mstrres/formmaker/deflib`

Returns Success or failure

Errors

Error	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0006	The virtual memory management API #APINAME,# failed.
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0008	The database API #APINAME,# failed accessing the table #TABLENAME,#.
DPR0012	The database API #APINAME,# cannot locate the table #TABLENAME,#.
DPR0036	The DSI variable #VARIABLE,# does not contain a valid FAP form set.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.

Example Here is an example:

```
INPUT
CONFIG formmaker
USERID FORMAKER
KEY1 INSURANCE PACKAGE
KEY2 COMMERCIAL
CHECKINARC YES

OUTPUT
RECORDS 4
RESULTS SUCCESS
SERVERTIMESPENT 0.150
TOTALTIMESPENT 1.072
```

DPRCheckLogin

Use this rule to create a hash password from REALPASSWORD and compare it with the hash password passed in as PASSWORD. The password is case sensitive. If you do not want to make the password case sensitive in the client application, uppercase the password before it is submitted to IDS.

NOTE: The IDS authentication rules include DPRDecryptLogin, DPRDefaultLogin, DPRLoginUser, DPRCheckLogin, and DPRGenerateSeedValue. These rules replace the DPRLogin rule under the Docupresentment authentication model. For more information, see Authenticating Users in the Internet Document Server Guide.

Syntax

```
Function = dprw32->DPRCheckLogin
```

Attachment variables

Variable	Description
LOGINRESULT	If this variable exists and its value is anything other than SUCCESS, the rule does nothing.
USERID	The user ID of the requestor.
PASSWORD	The password of requestor. It is a hash value.
REALUSERID	The user ID from the userinfo database.
REALPASSWORD	The password from the userinfo database.

Attachment outputs

Variable	Description
LOGINRESULT	If there is an error, this variable is created with the value FAILURE.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable VARIABLE in the attachment list.
DPR0053	Unable to get random seed value in #LOCATION,#.
DPR0054	Invalid login.

See also

DPRDecryptLogin on page 97

DPRDefaultLogin on page 100

DPRLoginUser on page 196

DPRGenerateSeedValue on page 128

DPRCheckWipRecords

Use this rule to create a WIP list using the KEYNAME attachment variable to search. This rule does not allow partial matches unless the PARTIALMATCH attachment variable is present.

The search is not case sensitive unless the CASESENSITIVE attachment variable is present or the following INI option is set to Yes:

```
< Control >
   CaseSensitiveKeys = Yes
```

The rule first checks the CaseSensitiveKeys option and then checks the CASESENSITIVE attachment variable. The attachment variable overrides the INI option.

You can specify the starting record and the maximum records number to return. The array of the fields is defined in the WIP DFD file or in DBFFields if the WIP DFD file is missing.

Syntax

```
long _DSIAPI DPRCheckWipRecords ( DSIHANDLE hdsi,
                        char * pszParms,
                        unsigned long  ulMsg,
                        unsigned long  ulOptions )
```

Parameters

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

Attachment variables

This rule expects these attachment variables:

Variable	Description
KEYNAME	The name of one of the fields in WIP DFD.
USERID	The ID of the required name for the user.
STARTRECORD	The starting record number (default is 1).
MAXRECORDS	The maximum number of records to be retrieved (default is 20).
STATUS	A status code specified by the WIP, Approve, and Reject INI options (W, AP, and RJ)
FIELDNAME	The value of the field as defined in the WIP DFD file or default fields, such as Key1, Key2, KeyID, RecType, and so on. You must include all fields even if some do not have values.
PARTIALMATCH	If present, the rule includes partial matches.

Variable	Description
CASESENSITIVE	If present, the rule considers case when building the WIP list.
CURRUSER	(Optional) If you specify this input attachment variable: `CURRUSER=~UNKNOWN~` the rule 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 checks the input attachment variable USEREPORTTOLIST as before and it has no effect if you specify *CURRUSER=~UNKNOWN~*.

Attachment outputs

The output attachment variables include:

Variable	Description
WIP	The status generated from WIP option in the Status_CD control group.
Approve	The status generated from the Approve option in the Status_CD control group.
Reject	The status generated from the Reject option in the Status_CD control group.
Records	The number of selected records.
RECORDSX. FieldName	The field name for selected single or multiple records, where the affix X (WIPSX.FieldName) is the number of selected WIP records, counting from 1 to RECORDS; FieldName is the field name as defined WIPDFD file. If the DFD file is missing, default field names are used, such as. Key1, Key2, KeyID, RecType, and so on.

Request types

ReqType = WFD

Here is an example request type:

```
< ReqType:WFD >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRCheckWIPRecords
```

INI options

You can use these INI options with this rule:

Option	Control group	Description
FormLib	MasterResource	Specifies the path to the forms.
ImageExt	Control	Specifies the type of image file.
LogoExt	Control	Specifies the type of logo image.

Option	Control group	Description
CaseSensitiveKeys	Control	Enter Yes if you want the rule to consider case. The default is No. The CASESENSITIVE attachment variable overrides this option.
File	WIPData	Specifies name of the WIP file.
Path	WIPData	Specifies the path to the WIP file.
MaxWIPRecords	WIPData	Specifies the maximum records to read into the processQ. Prevents it from slowing down because of the volume of records.
File	UserInfo	Specifies name of the userinfo file.
Path	UserInfo	Specifies the path to the userinfo file. If this file is missing, USERID is added in the user list.
WIP	Status_CD	Specifies the WIP status code.
Approve	Status_CD	Specifies the approve status code.
Reject	Status_CD	Specifies the reject status code.

Here is an example:

```
< MasterResource >
   FormLib  = mstrres\sampco\forms\
< Control >
   ImageEXT   = .fap
   LogoExt    = .log
   CaseSensitiveKeys = Yes
< WIPData >
   File    = WIP
   Path    = mstrres\sampco\wip\
   MaxWIPRecords = 200
< UserInfo >
   File    = userinfo
   Path    = mstrres\
< Status_CD >
   WIP     = W
   Approve = AP
   Reject  = RJ
```

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.

Message	Description
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0025	Cannot add variable #VARIABLE,# to the attachment record #RECORD,#
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRAddWipRecord on page 71

DPRApproveWipRecords on page 73

DPRAssignWipRecord on page 78

DPRCheckWipRecords on page 88

DPRDeleteWipRecord on page 105

DPRDelMultiWipRecords on page 109

DPRFindWipRecordsByUser on page 125

DPRGetWipList on page 150

DPRGetWipFormset on page 153

DPRGetWipRecipients on page 156

DPRModifyWipData on page 205

DPRSearchWip on page 246

DPRUpdateWipRecords on page 287

DPRWipTableParms on page 298

DPRCompareXMLFiles

Use this rule to compare XML files.

Syntax

```
long _DSIAPI DPRCompareXMLFiles ( 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

To set up this rule, add the ReqType control group in the DOCSERV.INI file as shown here:

```
< ReqType:XML >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = dprw32->DPRCompareXMLFiles
    function = atcw32->ATCUnloadAttachment
```

Set up the ATTACH.MSG file as shown here:

```
USERID=USER
DOCTYPE=DAP
REQTYPE=PRT
CONFIG=UTILITY
ARCKEY1=00345A0D5600000002
ARCKEY2=00345A0D5600000004
Company=1166666
Lob=Lee
PolicyNum=Roswell,Ga 30015
RunDate=020698
PRINTPATH=html\
recipient=CUSTOMER
PRTTYPE=XML
BIGVARIABLE=123456
```

For debugging purposes, set the Attachments control group as shown here:

```
< Attachments >
    Debug = Yes
```

When this rule is called, it opens the ATTACH.MSG file to locate the ArcKey1 and ArcKey2 variable values. If both ArcKey1, ArcKey2, and the corresponding form sets exist, the rule retrieves and converts the form sets.

DIFCompareXMLFiles generates an XML document which contains the results of the comparison. The XML document handle is stored under DPRXMLFORMSET in the variable list. You can locate it by calling:

```
DSILocateValue( hdsi, "DPRXMLFORMSET", &hDocument,
sizeof(hDocument);
```

where, on success, hDocument returns the new XML document handle for the user's application.

Errors

There are three scenarios an application may run into:

- If ArcKey1 exists but provides a wrong value for the form set retrieval, regardless of whether ArcKey2 exists, there will be no DIFCompareXMLFiles. The system returns a NULL XML document handle and an error condition (DPR0019).
- If ArcKey1 exists and ArcKey2 does not an XML document handle for the first form set is returned without DIFCompareXMLFiles. There is no error condition for this case.
- If both ArcKey1 and ArcKey2 exist, but ArcKey2 provides an incorrect variable value, the second form set is not retrieved. The system will generate an error condition (DPR0019) as a warning. The rule returns an XML document handle for the first form set without DIFCompareXMLFiles.

There may be other errors, including these:

Message	Description
DPR0001	Can not locate ArcKey variable
DPR0011	ArcKey does not contain valid data
DPR0024	Can not create DPRXMLFORMSET variable, and the platform error for DPRCompareXMLFiles resulted from DIFCompareXMLFiles failure.

In any error condition, be sure to check ArcKey and any related problems.

DPRConvertGUID

Use this rule to convert attachment variable containing GUID in the form of a string from a short representation to a long representation and back.

Syntax

```
long _DSIAPI DPRConvertGUID ( 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

On MSG_RUNF the GUID in the input attachment is converted to a long form, on MSG_RUNR the GUID in the output attachment is converted back to a short form.

The short form is when each three bytes of binary data are converted into four bytes of text, the long form is when each of the binary bytes is converted into two bytes of text which is hex representation of the byte.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot find the variable VARIABLE in the attachment list.
DPR0038	The rule parameters PARAMETERS for rule RULE are not correct or empty.

DPRCreateEMailAttachment

Use this rule to create HTML file from XML stored internally at XMLDOCVAR. Run this rule after you run the DPRParseRecord rule to set up XMLDOCVAR.

NOTE: This rule is only available on Windows 32-bit platforms.

Syntax

```
long _DSIAPI DPRCreateEMailAttachment ( 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

After the file is created, it can be used by the DPRMail rule.

See also DPRMail on page 197

DPRDebug

Use this rule as a memory debugging rule for the Documaker Bridge. This rule does a printf of the number of memory allocations, frees, and the difference on every message.

NOTE: Interpreting the information this rule provides should only be done by qualified personnel.

Syntax

```
long _DSIAPI DPRDebug ( 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

DPRDecryptLogin

Use this rule to decrypt the USERID and PASSWORD values on the initial login on the RUNF message. If these values are not encrypted they are left alone. On RUNR, this rule encrypts a value of the encrypted USERID and hash PASSWORD and places this value into the USERID attachment variable.

The password is case sensitive. If you do not want to make the password case sensitive in the client application, uppercase the password before it is submitted to IDS.

NOTE: The IDS authentication rules include DPRDecryptLogin, DPRDefaultLogin, DPRLoginUser, DPRCheckLogin, and DPRGenerateSeedValue. These rules replace the DPRLogin rule under the Docupresentment authentication model. For more information, see Authenticating Users in the Internet Document Server Guide.

Syntax

```
Function = dprw32->DPRDecryptLogin
```

Attachment variables

You have these input and output attachments on RUNF:

Variable	Description
LOGINRESULT	If this variable exists and its value is anything other than SUCCESS, the rule does nothing. In case of error it is created with the value FAILURE.
USERID	The user ID of the requestor.
PASSWORD	The password of the requestor. A hash value is sent to output attachment.
PASSWORDENCRYPTED	A flag. If USERID and PASSWORD are encrypted values, set to Yes. The default is Yes.

You have these output attachments on RUNR:

Variable	Description
USERID	The user ID of the requestor. It is an encrypted value of the encrypted USERID and hash PASSWORD.
LOGINRESULT	If an error occurs, it is created with the value FAILURE.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate the variable VARIABLE in the attachment list.
DPR0004	The user ID USERID is invalid.
DPR0005	The password specified for user USERID is incorrect.

Message	Description
DPR0053	Unable to get a random seed value in #LOCATION,#.

See also

DPRCheckLogin on page 87

DPRDefaultLogin on page 100

DPRLoginUser on page 196

DPRGenerateSeedValue on page 128

DPRDecryptValue

Use this rule to decrypt the key information. Rule parameters are the comma-delimited names of the attachment variables which are to be decrypted.

Syntax

```
long _DSIAPI DPRDecryptValue ( 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

On MSG_RUNF these values are decrypted in the input attachment and put back into the input attachment, on the MSG_RUNR these values are encrypted again from output attachment and put back into output attachment.

This rule should be the first rule in the rule list for a particular request type after the ATCLoadAttachment and ATCUnloadAttachment rules have been called. If one of the variables is not found in the attachment, error message is generated and processing continues.

INI options

Use the Debug option with this rule:

```
< DPRDecryptValue >
    Debug = No
```

This option defaults to No. If you set this option to Yes, the values before and after encryption and decryption are written to the DPRTRC.LOG file.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot find the variable VARIABLE in the attachment list.
DPR0037	The attachment value VARIABLE with the value VALUE is not a valid encrypted string.
DPR0038	The rule parameters PARAMETERS for rule RULE are not correct or empty.

DPRDefaultLogin

Use this rule to get the USERID value from input attachment and locate a matching record in the user table. By default, the rule uses Documaker's USERINFO table. In RUNF message, this rule creates the REALUSERID and REALPASSWORD values from userinfo database based on the USERID value passed in.

The password is case sensitive. If you do not want to make the password case sensitive in the client application, uppercase the password before it is submitted to IDS.

NOTE: The IDS authentication rules include DPRDecryptLogin, DPRDefaultLogin, DPRLoginUser, DPRCheckLogin, and DPRGenerateSeedValue. These rules replace the DPRLogin rule under the Docupresentment authentication model. For more information, see Authenticating Users in the Internet Document Server Guide.

Syntax

```
Function = dprw32->DPRDefaultLogin
```

Attachment variables

Variable	Description
LOGINRESULT	If this variable exists and its value is anything other than SUCCESS, the rule does nothing.
USERID	The user ID of the requestor.

Attachment outputs

Variable	Description
LOGINRESULT	In case of error, this variable is created with the value FAILURE.
REALUSERID	The matched user ID from the userinfo database.
REALPASSWORD	The password for the matched user ID.
RIGHTS, REPORTTO, SECURITY, and USRMESSAGE	These values come from the corresponding columns in the Documaker user table.

INI options

You can use this INI option:

```
< UserInfo >
    UserInfo =
```

Option	Description
UserInfo	Enter the name of the userinfo database file.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate the variable #VARIABLE,# in the attachment list.
DPR0003	The user information database FILENAME could not be opened.
DPR0004	The user ID USERID is invalid.

See also

DPRCheckLogin on page 87

DPRDecryptLogin on page 97

DPRLoginUser on page 196

DPRGenerateSeedValue on page 128

DPRDelBlankPages

Use this rule to remove blank or filler pages from a form set. For instance, you can use this rule to remove blank pages reserved for OMR marks.

NOTE: When you use the DPRDelBlankPages or DPRRotateFormsetPages rules with form sets created from Metacode or AFP print streams, the rules work fine. If, however, you use these rules with form sets created from Documaker archives or from import files, the rule appear to work incorrectly because not all of the static form data is loaded when these rules execute. The result is that text may not be rotated or pages with content may be deleted.

Use the DPRLoadFAPImages rule to correct this problem. Insert this rule after the rule that creates the form set, such as DPRRetrieveFormset or DPRLoadImportFile, and before the rule that prints the form set, such as DPRPrintFormset or DPRPrint.

Syntax

```
long _DSIAPI DPRDelBlankPages ( 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

This rule assumes that the form set has been loaded by the Documaker Bridge into the DSI variable, DPRFORMSET.

If you are using this rule with a different bridge, you may need to specify a different DSI variable that contains the form set. Here is an example,

```
function = dprw32->DPRDelBlankPages,MTCFORMSET
```

NOTE: You can also remove blank or filler pages using custom code or a DAL script which includes the DelBlankPages function. See the DAL Reference for more information on the DelBlankPages function.

The API to call from custom code is as follows:

```
DWORD _VMMAPI FAPDelBlankPages(
               VMMHANDLE objectH,      /* form set or form handle */
```

See also

DPRAddBlankPages on page 63

DPRLoadFAPImages on page 185

DPRDeleteFiles

Use this rule to delete the following file types from an IDS Documanage cache: XML, TXT, HTM, PDF, TIF, JPG, DPA, AFP, GIF, MET, DOC, BMP, and RTF.

Syntax

```
long _DSIAPI DPRDeleteFiles ( DSIHANDLE hdsi,
                              char * pszParms,
                              unsigned long  ulMsg,
                              unsigned long  ulOptions )
```

Parameters

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

This rule is useful for deleting files cached by an IDS Documanage request when you are running performance benchmark tests. This rule runs in RUNR message.

The DPRDeleteFiles rule only removes the files associated with the file name value for the GEN_TARGETFILENAME attachment variable generated by a Documanage request which generates the aforementioned output attachment variable. This rule only looks for files to remove in the default cache directory of the current IDS server.

Attachment variables

Variable	Description
GEN_TARGETFILENA ME	Contains the output file name of a file requested in a Documanage request. This variable is generated by Documanage request types that retrieve a file from Documanage, such as the BIA request type. This information is used to remove all files associated with the file requested.

Attachment outputs

Variable	Description
RESULTS	Success or failure

DPRDeleteWipRecord

Use this rule to delete a record and remove the NAFILE.DAT and POLFILE.DAT files. It is equivalent to the IPPDeleteWIP rule.

Syntax

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

Parameters

Parameter	Description
DSIHANDLE hdsi	pointer to the rules data
char * pszParms	pointer to rule parameter string which may or may not contain the DSI variable name FormsetName that stores form set handle.
ULONG ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
ULONG ulOptions	options

The system identifies the record to delete by first looking for the attachment RECORDID (or RECNUM and UNIQUE_ID). If the RECORDID is not found, it searches for the fields defined in DOC_TAG.

Attachment variables

This rule expects these attachment variables:

Variable	Description
RecordID	Enter the record ID. You can define it as the RECNUM or UNIQUE_ID in your DFD definition. UNIQUE_ID is typically used in SQL databases.
(*field names*)	Enter the appropriate value to match a record. Key1, Key2, KeyID, and RecType are required. See the definition of DOC_TAG.in the WIP.DFD file.

INI options

You can use these INI options:

```
< WIPData >
   File =
   Path =
```

Option	Description
File	Enter the name of the WIP file.
Path	Enter the path to the WIP file.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRAddWipRecord on page 71

DPRAssignWipRecord on page 78

DPRDelMultiWipRecords on page 109

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRUpdateWipRecords on page 287

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRWipTableParms on page 298

DPRDelFromUserDict

Use this rule to delete words from the user dictionary.

Syntax

```
long _DSIAPI DPRDelFromUserDict ( DSIHANDLE hdsi,
                         char * pszParms,
                         unsigned ulMsg,
                         unsigned ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
DelLine	A line of words you want deleted from the user dictionary. Separate the words with commas.
LanguageOpt	The language selection. The default is US English. You can choose from these languages and dictionaries: Danish “ssceda.tlx,ssceda2.clx” Dutch “sscedu.tlx,sscedu2.clx” Finnish “sscefi.tlx,sscefi2.clx” French “sscefr.tlx,sscefr2.clx” German “sscege.tlx,sscege2.clx” Italian “ssceit.tlx,ssceit1.clx” Norwegian “sscenb.tlx,sscenb2.clx” Portuguese_Brazil “sscepb.tlx,sscepb2.clx” Portuguese “sscepo.tlx,sscepo2.clx” Spanish “sscesp.tlx,sscesp2.clx” Swedish “sscesw.tlx,sscesw2.clx” UK English “sscebr.tlx,sscebr2.clx” US English “ssceam.tlx,ssceam2.clx”
UserDict	The name of the user dictionary. The default is user.tlx.

Attachment outputs

None

INI options

You can use these INI options with this rule:

```
< Spell >
    LanguageOpt     =
    UserDict        =
    UserDictPath    =
```

Option	Description
LanguageOpt	Enter the language option. The default is US English. You can choose from these languages and dictionaries:
	Danish “ssceda.tlx,ssceda2.clx”
	Dutch “sscedu.tlx,sscedu2.clx”
	Finnish “sscefi.tlx,sscefi2.clx”
	French “sscefr.tlx,sscefr2.clx”
	German “sscege.tlx,sscege2.clx”
	Italian “ssceit.tlx,ssceit1.clx”
	Norwegian “sscenb.tlx,sscenb2.clx”
	Portuguese_Brazil “sscepb.tlx,sscepb2.clx”
	Portuguese “sscepo.tlx,sscepo2.clx”
	Spanish “sscesp.tlx,sscesp2.clx”
	Swedish “sscesw.tlx,sscesw2.clx”
	UK English “sscebr.tlx,sscebr2.clx”
	US English “ssceam.tlx,ssceam2.clx”
UserDict	Enter the name of the user dictionary. The default is user.tlx.
UserDictPath	Enter the path to the user dictionary. The default is the current working directory.

Errors

Error	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRAddToUserDict on page 69

DPRRetFromUserDict on page 227

DPRSpellCheck on page 257

DPRDelMultiWipRecords

Use this rule to delete records and remove NAFILE.DAT and POFILE.DAT files.

Syntax

```
long _DSIAPI DPRDelMultiWipRecords ( DSIHANDLE hdsi,
                        char * pszParms,
                        unsigned long  ulMsg,
                        unsigned long  ulOptions )
```

Parameters

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

Attachment variables

This rule expects one of these attachment variables:

Variable	Description
RECORDID	Multiple record numbers separated by commas.
RECNUM	Multiple record numbers separated by commas.
UNIQUE_ID	Multiple record numbers separated by commas for SQL databases.

To identify records, it first looks for the RECORDID attachment variable. If that variable is not found, it looks for RECNUM, then UNIQUE_ID. Specify the multiple records using ID numbers separated by commas.

Here is an example:

```
RECORDID = 5,4,3,2,1
```

INI options

You can use these INI options:

```
< WIPData >
    File =
    Path =
```

Option	Description
File	Enter the name of the WIP file.
Path	Enter the path to the WIP file.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.
DPR0061	Unable to WIPFind the record #RECORDID,#.
DPR0062	Unable to WIPDelete the record #RECORDID,#.

See also

DPRAddWipRecord on page 71

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRUpdateWipRecords on page 287

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRWipTableParms on page 298

DPRDepagination

Use this rule to depaginate a form set you will export to an XML tree.

Syntax

```
long _DSIAPI DPRDepagination ( DSIHANDLE hdsi,
                               char * pszParms,
                               unsigned ulMsg,
                               unsigned ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
DPRFORMSET	This DSI variable should contain the name of the DAP form set to export. This form set is created by some other rule, such as the DPRLoadImportFile rule.

Attachment outputs

None

Example

Here is an example of the request type:

```
< ReqType:PGN >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRLoadImportFile
    function = dprw32->DPRDepagination
    function = dprw32->DPRUnloadExportFile
```

Errors

Error	Description
DPR0017	Cannot locate DSI variable #VARIABLE,#.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0024	Cannot create DSI variable #VARIABLE,#.
DPR0036	The DSI variable #VARIABLE,# does not contain a valid FAP form set.

DPRDpw2Wip

Use this rule to save the DPW file contents in the WIP record. This rule expects the DPW file to have already been created with the ATCReceiveFile rule.

Syntax

```
long _DSIAPI DPRDpw2Wip ( 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

Attachment variables

This rule expects this input attachment variable:

Variable	Description
RECNUM or UNIQUE_ID	Lets the rule find the correct WIP record.

Attachment outputs

The WIP record is stored in this attachment variable:

Variable	Description
RF_POSTFILE	The file name of DPW file.

Errors

This rule can return these messages:

Message	Description
DPR0001	The variable RF_POSTFILE is not present.
DPR0031	Could not open the DPW file
DPR0007	Could not find <WipData> <File>
DPR0039	The API failed.

See also

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRFile2Dpw on page 120

DPRGetOneWipRecord on page 143

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPREditUserDict

Use this rule to edit a user dictionary.

Syntax

```
long _DSIAPI DPREditUserDict ( DSIHANDLE hdsi,
                               char * pszParms,
                               unsigned ulMsg,
                               unsigned ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
EditFile	The name of the input XML file you want to use to edit the user dictionary.
LanguageOpt	The language selection. The default is US English. You can choose from these languages and dictionaries: Danish “ssceda.tlx,ssceda2.clx” Dutch “sscedu.tlx,sscedu2.clx” Finnish “sscefi.tlx,sscefi2.clx” French “sscefr.tlx,sscefr2.clx” German “sscege.tlx,sscege2.clx” Italian “ssceit.tlx,ssceit1.clx” Norwegian “sscenb.tlx,sscenb2.clx” Portuguese_Brazil “sscepb.tlx,sscepb2.clx” Portuguese “sscepo.tlx,sscepo2.clx” Spanish “sscesp.tlx,sscesp2.clx” Swedish “sscesw.tlx,sscesw2.clx” UK English “sscebr.tlx,sscebr2.clx” US English “ssceam.tlx,ssceam2.clx”
UserDict	The name of the user dictionary. The default is user.tlx.

Attachment outputs

None

INI options

You can use these INI options with this rule:

```
< Spell >
    LanguageOpt     =
    UserDict        =
    UserDictPath    =
```

Option	Description	
LanguageOpt	Enter the language option. The default is US English. You can choose from these languages and dictionaries:	
	Danish	"ssceda.tlx,ssceda2.clx"
	Dutch	"sscedu.tlx,sscedu2.clx"
	Finnish	"sscefi.tlx,sscefi2.clx"
	French	"sscefr.tlx,sscefr2.clx"
	German	"sscege.tlx,sscege2.clx"
	Italian	"ssceit.tlx,ssceit1.clx"
	Norwegian	"sscenb.tlx,sscenb2.clx"
	Portuguese_Brazil	"sscepb.tlx,sscepb2.clx"
	Portuguese	"sscepo.tlx,sscepo2.clx"
	Spanish	"sscesp.tlx,sscesp2.clx"
	Swedish	"sscesw.tlx,sscesw2.clx"
	UK English	"sscebr.tlx,sscebr2.clx"
	US English	"ssceam.tlx,ssceam2.clx"
UserDict	Enter the name of the user dictionary. The default is user.tlx.	
UserDictPath	Enter the path to the user dictionary. The default is the current working directory	

Errors

This rule can return these messages:

Error	Message
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

Edit file layout

Here is an example of the edit file layout:

```
<SPELLER TYPE="IENTRY" VERSION="3.1">
<FIELDH ACTION="DELETE">speling</FIELDH>
<FIELDH>spellin</FIELDH>
<FIELDH ACTION="ADD">spellng</FIELDH>
</SPELLER>
```

DPRExecuteDAL

Use this rule to execute a DAL script. The parameters to DPRExecuteDAL specify where the DAL script is located and on what DSI message to execute this script. Values for the rule parameter include the name of the DAL script and one of these strings:

- INIT
- RUNF
- RUNR
- TERM

Syntax

```
long _DSIAPI DPRExecuteDAL ( 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

This rule returns an error code if DAL had errors.

Attachment variables

None.

Example

Here is an example:

```
function = DPRW32->DPRExecuteDAL,myownscript.dal,INIT
```

This will execute myownscript.dal when this rule receives message INIT. By default, the script is executed on message DSI_MSGRUNF.

Errors

Message	Description
DPR0064	Failed to execute DAL script #NAME

DPRFap2Html

Use this rule to produce HTML output for one or more FAP files. This rule can produce standard HTML output through the HTML Print Driver or an HTML representation of a TerSub paragraph.

This rule can process images from a form set in memory, a comma delimited list of images, or a form set retrieved for a GROUP1/GROUP2 combination. It can write the HTML output to a PRINTPATH or to the current IDS directory.

This rule can also send the HTML output as file attachments in the output message. This lets you decide whether to print the files to a remote location or send them as part of the output message.

In addition, this rule can generate unique names for each file or it can use the names of the images as the names of the output files. It can cache the output files, when appropriate. This rule removes the files if the Send option is set to Yes and the Debug option is omitted. You can also send debugging information to the DPR trace log if the debug option is set.

Syntax

```
long _DSIAPI DPRFap2Html ( 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

Attachment variables

This rule expects these input attachment variables:

Variable	Description
RESULTS	(Optional) If present and the value is not SUCCESS, the rule exits. You can use this rule to make sure other rules running before this rule but in the same request run successfully.
SOURCE	(Optional) If omitted, the system checks for the DPRFORMSET DSI global variable to retrieve the form set from memory that it will use to retrieve image information to produce HTML output. If present, it overrides any form set in memory and you can provide one of these values: • A comma-delimited list of images to process to output HTML. The list can consist of one or more images. In this case, there is no need for a form set to reside in memory as each image will be loaded and processed. • An asterisk (*) tells the system to process all images for a GROUP1/GROUP2 combination. You must provide the GROUP1 and GROUP2 input attachment variables. The rule uses them to retrieve the form set it will use to get image information for producing the HTML output.

Variable	Description
GROUP1	(Optional) Only include this variable when the value for the SOURCE input attachment variable is an asterisk (*). This variable is used to retrieve a form set.
GROUP2	(Optional) Only include this variable when the value for the SOURCE input attachment variable is an asterisk (*). This variable is used to retrieve a form set.
TERSUB	(Optional) If you set this variable to Yes, the rule produces an HTML representation of a TerSub paragraph for the images provided in the SOURCE input attachment variable or in the form set in memory.
SEND	(Optional) If you set this variable to Yes, the rule sends the HTML output as file attachments in the output message.
UNIQUE	(Optional) If you set this variable to Yes, the rule generates a unique name for each output file. If you omit this variable, the image name is used as the name portion of the output files.
DEBUG	(Optional) If you set this variable to Yes, the rule sends debugging information to the DPR trace log.
CACHE	(Optional) If you set this variable to Yes, the rule caches the HTML output files on disk.
PRINTPATH	(Optional) If you include this variable, the rule uses the path provided as the location for the HTML output files it will write to disk.

Returns

This rule outputs these attachment variables:

Variable	Description
RESULTS	Success or failure.

Errors

Error	Description
DPR0001	Missing input attachment variable.
DPR0006	Virtual memory API failed. Returns the name of the API that failed.
DPR0010	API failed. Returns the name of the calling API and the name of the API that failed. Used to indicate one of the external APIs not used exclusively by this rule failed.
DPR0016	Failed to unload the HTML template. Returns the name and location of the template that failed to unload.
DPR0017	Failed to locate the DSI variable DPRformset.
DPR0027	Failed to load the form set.
DPR0028	Failed to load the image.

Error	Description
DPR0039	API failed. Returns the name of the calling API and the name of the API that failed. Used to indicate one of the internal APIs used exclusively by this rule failed.

Example

Here are example request types:

```
[ReqType:TEST_DPRFap2Html_W_Source]
function = atcw32->ATCLoadAttachment
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSetConfig
function = dprw32->DPRInitLby
function = dprw32->DPRFap2Html

[ReqType:TEST_DPRFap2Html_W_formsetInMemory]
function = atcw32->ATCLoadAttachment
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSetConfig
function = dprw32->DPRInitLby
function = dprw32->DPRLoadImportFile
function = dprw32->DPRFap2Html
```

DPRFile2Dpw

Use this rule to insert files into the DPW file. You can also use it to download files such as DFD, INI, or any other file accessible by IDS.

Syntax

```
long _DSIAPI DPRFile2Dpw ( 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

INI options

Be sure to include these INI options:

```
< File2DPW >
    INIToken    = d:\docserv\sfcdwn.ini
    DFD         = d:\sfc\wip\wip.dfd
    XRFToken    = safeco.fxr
```

Attachment variables

This rule expects this attachment variables:

Variable	Description
RF_POSTFILE	The path to the DPW file.

Errors

This rule can return these messages:

Message	Description
DPR0039	The API failed.
DPR0001	Failed to get the attachment variable.
DPR0056	File not present.

See also

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDpw2Wip on page 112

DPRIni2XML on page 159

DPRGetOneWipRecord on page 143

DPRLockWip on page 191

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRUnlockWip on page 281

DPRFilterFormsetForms

Use this rule to search for forms.

Syntax

```
long _DSIAPI DPRFilterFormsetForms ( 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

You can use these attachment variables to search for forms:

- DPRFORMNAME
- DPRFORMDESCRIPTION
- DPRKEY1
- DPRKEY2

The conditions that can be used are: equals, contains, and starts with. To specify the value and the condition the user will have to provide the attachment variable in the following format:

```
DPRFORMNAME.CONTAINS.
```

Conditions are checked in the following order: equals, starts with, contains. If more than one is provided the first one found will be used.

See also

DPRLoadXMLAttachment on page 188

DPRLoadedXML2Formset on page 184

DPRSendFormsetXML on page 251

DPRUpdateFromMRL on page 282

DPRSortFormsetForms on page 260

DPRGetFormList on page 137

DPRGetHTMLForms on page 141

DPRFindTemplate

Use this rule to find the correct template using transaction type. The REQTYPE attachment variable is matched with an option in the XML2ATTACH or XML2BODY control groups.

Either of these INI options should contain a path to the template for the transaction. The file name of the template is added as an attachment variable (XMLTEMPLATTACH) if the REQTYPE is found under XML2ATTACH.

The file name is added as the XMLTEMPLBODY variable if REQTYPE is found under XML2BODY.

NOTE: This rule is only available on Windows 32-bit platforms.

Syntax

```
long _DSIAPI DPRFindTemplate ( 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

The system expects the REQTYPE attachment variable, which should have matching entry in the INI file.

Errors

This rule can return these messages:

Message	Description
DPR0001	cannot find attachment variable
DPR0007	INI option not defined
DPR0010	API failed, reason unknown

DPRFindWipRecords

NOTE: The DPRFindWipRecords rule was replaced by the DPRSearchWip rule with the release of Shared Objects version 11.1. Any calls to DPRFindWipRecords execute DPRSearchWip instead and there is no difference in the result. The DPRFindWipRecords name was kept for legacy support. For more information, see DPRSearchWip on page 246.

DPRFindWipRecordsByUser

Use this rule to search for one or more records based on provided fields and user IDs. This rule returns a list by adding every field of each record into the attachment in the user's queue.

Syntax

```
long _DSIAPI DPRFindWipRecordsByUser ( DSIHANDLE hdsi,
                        char * pszParms,
                        unsigned long  ulMsg,
                        unsigned long  ulOptions )
```

Parameters

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

Attachment variables

The system expects these attachment variables:

Variable	Description
USERID	The ID of the queue name for the user.
STARTRECORD	The starting record number. The default is one (1).
MAXRECORDS	The maximum number of records to be retrieved. The default is 20.
STATUSCODE	One of statuses specified by WIP, Approve, and Reject (W, AP, and RJ).
CURRUSER	While this rule does not support USEREPORTTOLIST and paging, if you specify: `CURRUSER=~UNKNOWN~` the system generates the same unknown user WIP list as does the DPRFindWipRecords/DPRSearchWip rule.
FIELDNAME= Value	The value of the field as defined in the WIPDFD file or in the default fields, such as Key1, Key2, KeyID, and RecType. You must list all fields even if some fields do not have values.

Attachment outputs

The system creates these output attachment variables:

Variable	Description
WIP	The WIP status generated from WIP option in the Status_CD control group.
APPROVE	The approve status generated from Approve option in the Status_CD control group.
REJECT	The reject status generated from Reject option in the Status_CD control group.
RECORDS	The number of selected records.
RECORDSX .FieldName	The field name for selected single or multiple records, where the affix X (WIPSX.FieldName) is the number of selected WIP records, counting from one to RECORDS and FieldName is the field name as defined WIPDFD file. If the DFD file is missing, the default field names are used, such as Key1, Key2, KeyID, and RecType.

Request types

ReqType = WFD

The requested type is required in the DOCSERV.INI file. Here is an example:

```
< ReqType:WFD >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRFindWipRecordsByUser
```

INI options

You can use these INI options:

```
< MasterResource >
    FormLib        = mstrres\sampco\forms
< Control >
    ImageExt       = .fap
    LogoExt        = .log
< WIPData >
    File           = WIP
    Path           = mstrres\sampco\wip
    MaxWIPRecords  = 200
< UserInfo >
    File           = userinfo
    Path           = mstrres
< Status_CD >
    WIP            = W
    Approve        = AP
    Reject         = RJ
```

Option	Description
MasterResource control group	
FormLib	Specifies the path to the forms.
Control control group	

Option	Description
ImageExt	Specifies the type of image file.
LogoExt	Specifies the type of logo image.
WIPData control group	
File	Specifies the WIP file name.
Path	Specifies the path to the WIP file
MaxWIPRecords	Specifies the maximum records to read in the processQ. Use this to prevent it from slowing due to volume records.
UserInfo control group	
File	Specifies the USERINFO file name.
Path	Specifies the path to the USERINFO file. If the USERINFO file is missing, USERID is added to the user list.
Status_CD control group	
WIP	Specifies the WIP status code.
Approve	Specifies the approve status code.
Reject	Specifies the reject status code.

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0001	Cannot locate the variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0022	Cannot add the variable #BARIABLE,# to the attachment list.
DPR0025	Cannot add the variable #VARIABLE,# to the attachment record #RECORD,#
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also DPRSearchWip on page 246

DPRGenerateSeedValue

Use this rule to generate a random seed value of two bytes for encrypting a text string by crypt(). It checks to see if a seed value exists and if not found, creates one. The rule can create a new random seed on a timer if you use it with the timer setup.

NOTE: The IDS authentication rules include DPRDecryptLogin, DPRDefaultLogin, DPRLoginUser, DPRCheckLogin, and DPRGenerateSeedValue. These rules replace the DPRLogin rule under the Docupresentment authentication model. For more information, see the Internet Document Server Guide.

Syntax

```
Function = dprw32->DPRGenerateSeedValue
```

There are no attachments. This rule runs on the RUNF message. You should execute this rule at least once a day. Here is an example:

```
< ReqType:WLG >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRDecryptLogin
    function = dprw32->DPRDefaultLogin
    function = dprw32->DPRLoginUser
    function = dprw32->DPRGetWipList
< ReqType:WLT >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRDecryptLogin
    function = dprw32->DPRDefaultLogin
    function = dprw32->DPRCheckLogin
    function = dprw32->DPRGetWipList
```

INI options

Use these INI options to reset the seed value every day at 3:00 AM.

```
< Timer >
    ResetSeed   = 3:00:00 AM
< ReqType:RESETSEED >
    function    = dprw32->DPRGenerateSeedValue
```

Errors

This rule can return these messages:

Message	Description
DPR0055	Unable to DSIGlobalDataCreate in #LOCATION,#. The global data setup is probably incorrect.

See also

DPRCheckLogin on page 87

DPRDecryptLogin on page 97

DPRDefaultLogin on page 100

DPRLoginUser on page 196

DPRGetConfigList

Use this rule to get a list 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

Attachment variables

None

Attachment outputs

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 the 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
```

DPRGetDFDInfo

Use this rule to retrieve an XML document with DFD field information for WIP or archive.

Syntax

```
long _DSIAPI DPRGetDFDInfo ( DSIHANDLE hdsi,
                        char * pszParms,
                        unsigned long  ulMsg,
                        unsigned long  ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
CONFIG	The configuration for which to get DFD information.
DFD	Enter one of these values: • WIP tells the system to return DFD information for WIP • ARCHIVE tells the system to return DFD information for archive. These values are not case sensitive.
DEBUG	(Optional) Yes tells the system to output the full path and file name of the DFD file it loaded. No tells the system to omit this debugging information.
PRINTPATH	(Optional) A path for the generation of the XML document with the DFD information. If this variable is omitted and the PRINTFILE input attachment variable is not provided, the system generates a unique file name and writes the XML document to the current IDS directory.
PRINTFILE	(Optional) A path and file name for the final output file. If this variable is present, it overrides any values provided for PRINTPATH.

Attachment outputs

Variable	Description
RESULTS	A value of success or failure.
DFDINFO	The path and file name of the XML document that contains the DFD information requested for a configuration.

Errors

Error	Description
DPR0001	Unable to find the input attachment variable.

Error	Description
DPR0006	Virtual memory API error.
DPR0039	API error.
DPR0043	Failed to DBQueryFormatInfo for DFD file.
DPR0049	Failed to create the XML document.
DPR0071	The value for attachment variable is not supported.
DPR0073	Failed to retrieve field information for DFD.

Keep in mind...

The XML document will contain a root node of name of WIPKEYS or ARCHIVEKEYS, depending on which DFD was requested by the DFD input attachment variable. The root node will contain a list of nodes with names that correspond to each of the base DFD field names. Each of those nodes will contain the following attributes:

Attribute	Description

NAME	The actual name in the user-defined DFD. This determination is made by reading the base name and looking for a mapping in the ArcRet or WIPData control group. If no mapping is found, the system assumes the name is the same as that of the base name. Here is an example of an entry in one of those groups:
	`< WIPData >`
	`    Key1 = Company`

KEY	A value of Yes or No indicating whether or not the field is defined as a key in the DFD. This determination is made by reading the Key = Yes/No setting for each field in the DFD.

DISPLAY	A value of Yes or No indicating whether or not the field is a display field. The value is derived by looking for Field entries in the AFEWipDisplay or AFEArchiveDisplay control groups. If the fields are not defined in these groups, the system sets the value equal to Yes for all fields defined as keys (see the Key attribute). Here is an example of an entry:
	`< AFEWIPDisplay >`
	`    Field = Key1,%-30.30s,Company`
	where Key1 is the field name used to set the DISPLAY attribute for the corresponding field in the DFD and Company is the description used to set the DOCSETHEADINGS attribute.

Attribute	Description

DOCSET HEADINGS	A description or text label for a display field. The value is derived by looking in the AFEWipDisplay or AFEArchiveDisplay control groups for Field entries. If the fields are not defined in these groups, the system sets the value equal to that of the field name for all fields defined as keys (see Key attribute). Here is an example of an entry:
	`< AFEWIPDisplay >`
	`    FIELD = KEY1,%-30.30s,Company`
	where KEY1 is the field name used to set the DISPLAY attribute for the corresponding field in the DFD and Company is the description used to set the DOCSETHEADINGS attribute.

STATUSCODE	The STATUSCODE field contains children derived in the following manner:
	The system looks for the STATUS_CD control group and adds each of the options listed as a child where the name of the node and the value of the name attribute are defined by the name of the option in the control group and the text for the node equals the value for the option in the control group.
	If the STATUS_CD control group is omitted, these defaults are used:
	`ARCHIVE = AR`
	`ASSIGN = A`
	`BATCHPRINT = B`
	`COMBINE = CO`
	`DUPLICATE = DU`
	`IN_PROGRESS = I`
	`PRINTED = P`
	`QUOTE = Q`
	`TRANSMIT = T`
	`WIP = W`

TRANCODE	The TRANCODE field contains children derived in this manner:
	The system looks for the TRANS_CD control group and adds each of the options listed as a child where the name of the node and the value of the name attribute are derived from the name of the option in the control group and the text for the node equals the value of the option in the control group.
	If the TRANS_CD control group is omitted, these defaults are used:
	`NEW = NB`
	`ENDORSE = EN`
	`REINSTATE = EI`
	`RENEWAL = RN`
	`CANCEL = CN`

In addition, the root element contains a child named CUSTOMKEYS with children corresponding to all user-defined DFD fields that are not part of the standard DFD field names. This determination is made by analyzing the user-defined DFD field names and looking for mappings in the ArcRet and WIPData control groups. If an entry is not found, the system looks for a field in the base DFD file that matches the name in the user-defined DFD file. If a match is not found, the field is deemed as a custom field and added as a KEY child to the CUSTOMKEYS node. Each KEY child contains these attributes:

Attribute	Description
NAME	The actual name in the user-defined DFD.
KEY	A value of Yes indicates the field is defined as a key in the DFD. This determination is made by reading the Key = Yes/No setting for each field in the DFD.

DISPLAY	A value of Yes indicates the field is a display field. This determination is made by looking for Field entries in the AFEWipDisplay or AFEArchiveDisplay control groups. If the fields are not defined in these groups, the system sets the value equal to Yes for all fields defined as keys (see the Key attribute). Here is an example of an entry:
	`< AFEWIPDisplay >`
	`    Field = KEY1,%-30.30s,Company`
	where KEY1 is the field name used to set the DISPLAY attribute for the corresponding field in the DFD file and Company is the description used to set the DOCSETHEADINGS attribute.

DOCSET HEADINGS	A description or text label for a display field. The value is derived by looking in the AFEWIPDisplay or AFEArchiveDisplay control groups for Field entries. If the fields are not defined in these groups, the system sets the value equal to that of the field name for all fields defined as keys (see the Key attribute). Here is an example of an entry:
	`< AFEWIPDisplay >`
	`    Field = KEY1,%-30.30s,Company`
	where Key1 is the field name used to set the DISPLAY attribute for the corresponding field in the DFD file and Company is the description used to set the DOCSETHEADINGS attribute.

Here is an example of an output file for WIP:

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT TYPE="RPWIP" VERSION="11.1">
<WIPKEYS>
   <KEY1 NAME="KEY1" KEY="YES" DISPLAY="YES" DOCSETHEADINGS="KEY1"/>
   <KEY2 NAME="KEY2" KEY="YES" DISPLAY="YES" DOCSETHEADINGS="KEY2"/>
    <KEYID NAME="KEYID" KEY="YES" DISPLAY="YES"
DOCSETHEADINGS="KEYID"/>
   <RECTYPE NAME="RECTYPE" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    <CREATETIME NAME="CREATETIME" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS=""/>
    <ORIGUSER NAME="ORIGUSER" KEY="YES" DISPLAY="YES"
DOCSETHEADINGS="ORIGUSER"/>
    <CURRUSER NAME="CURRUSER" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS=""/>
    <MODIFYTIME NAME="MODIFYTIME" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS=""/>
    <FORMSETID NAME="FORMSETID" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS=""/>
    <TRANCODE NAME="TRANCODE" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS="">
        <NEW NAME="NEW">NB</NEW>
        <ENDORSE NAME="ENDORSE">EN</ENDORSE>
        <CANCEL NAME="CANCEL">CN</CANCEL>
        <REINSTATE NAME="REINSTATE">EI</REINSTATE>
        <RENEWAL NAME="RENEWAL">RN</RENEWAL>
    </TRANCODE>
    <STATUSCODE NAME="STATUSCODE" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS="">
        <ARCHIVE NAME="ARCHIVE">A</ARCHIVE>
        <ASSIGN NAME="ASSIGN">A</ASSIGN>
        <BATCHPRINT NAME="BATCHPRINT">B</BATCHPRINT>
        <COMBINE NAME="COMBINE">CO</COMBINE>
        <DUPLICATE NAME="DUPLICATE">DU</DUPLICATE>
        <PRINTED NAME="PRINTED">P</PRINTED>
        <QUOTE NAME="QUOTE">Q</QUOTE>
        <TRANSMIT NAME="TRANSMIT">T</TRANSMIT>
        <WIP NAME="WIP">W</WIP>
    </STATUSCODE>
    <FROMUSER NAME="FROMUSER" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS=""/>
    <FROMTIME NAME="FROMTIME" KEY="NO" DISPLAY="NO"
DOCSETHEADINGS=""/>
    <TOUSER NAME="TOUSER" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    <TOTIME NAME="TOTIME" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    <DESC NAME="DESC" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    <INUSE NAME="INUSE" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    <ARCKEY NAME="ARCKEY" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
   <APPDATA NAME="APPDATA" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    <RECNUM NAME="RECNUM" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    <CUSTOMKEYS>
        <KEY NAME="PRODUCERNO" KEY="YES" DISPLAY="YES"
DOCSETHEADINGS="PRODUCERNO"/>
        <KEY NAME="CLAIMNO" KEY="YES" DISPLAY="YES"
DOCSETHEADINGS="CLAIMNO"/>
        <KEY NAME="CLAIMANT" KEY="YES" DISPLAY="YES"
DOCSETHEADINGS="CLAIMANT"/>
        <KEY NAME="INSUREDNM" KEY="YES" DISPLAY="YES"
DOCSETHEADINGS="INSUREDNM"/>
        <KEY NAME="DATE_TIME" KEY="YES" DISPLAY="YES"
DOCSETHEADINGS="DATE_TIME"/>
       <KEY NAME="ARCDATE" KEY="NO" DISPLAY="NO" DOCSETHEADINGS=""/>
    </CUSTOMKEYS>
</WIPKEYS>
</DOCUMENT>
```

DPRGetFormList

Use this rule to work with the IDS MRL and to get the group list, form list, and image list. This rule is a replacement for the following rules and exists only to make it more convenient to define the request type.

- DPRLoadXMLAttachment
- DPRLoadedXML2Formset
- DPRSendFormsetXML
- DPRUpdateFromMRL
- DPRFilterFormsetForms
- DPRSortFormsetForms
- DPRGetHTMLForms

When no customizations or changes to the parameters for these rules are needed, all of these rules, in this order, can be replaced by the DPRGetFormList rule, so the same request type can have these rules:

```
function = atcw32->ATCLoadAttachment
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSetConfig
function = dprw32->DPRGetFormList
```

Syntax

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

Parameters

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

See also

See the documentation for each rule for details on attachment variables, parameters, and return values:

DPRLoadXMLAttachment on page 188

DPRLoadedXML2Formset on page 184

DPRSendFormsetXML on page 251

DPRUpdateFromMRL on page 282

DPRFilterFormsetForms on page 122

DPRSortFormsetForms on page 260

DPRGetHTMLForms on page 141

DPRGetFormsetRecips

Use this rule to return a list recipients for the form set.

Syntax

```
long _DSIAPI DPRGetFormsetRecips ( DSIHANDLE hdsi,
                                   char * pszParms,
                                   ULONG ulMsg,
                                   ULONG ulOptions )
```

Parameters

Parameter	Description
DSIHANDLE hdsi	pointer to the rules data
char * pszParms	pointer to the rule parameter string
ULONG ulMsg	DSI_ message, such as DSI_MSGRUNF
ULONG ulOptions	options

Attachment variables

Variable	Description
DPRFORMSET	This DSI variable supplies the name of the form set to print, which has been created by some other rule, such as DPRLoadImportFile or DPRRetrieveDPA. You can overwrite the name DPRFORMSET using a parameter to this rule stored in the IDS configuration file.

Attachment outputs

This rule creates an attachment record called RECORDS with these values:

Variable	Description
RECIPIENT	The name of the recipient from the POL file.
DESCRIPTION	The recipient description, if specified in the Recip_Names control group, or if it is the same as the recipient name in the POL file. The application should use DESCRIPTION for displaying the recipient list.

The rule creates an attachment variable called RESULTS which runs on the DSI_MSGRUNF message.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0010	System encountered an internal error of unknown type. The call by LOCATION to APINAME failed.
DPR0022	Cannot add variable VARIABLE to the attachment list.
DPR0025	Cannot add variable VARIABLE to the attachment record RECORD

See also DPRGetRecipients on page 145

DPRGetHTMLForms

Use this rule to return HTML representation of FAP files (images). This rule is specified in the form set located in the DPRFORMSET DSI variable. Any images designated as print only or hidden are skipped.

Syntax

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

Parameters

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

The HTML files produced are sent back via the attachment. The delimiter name for the SOAP attachment is the image name or imagename_pagenum for the second and later pages of a multi-page image.

This rule runs on RUNF message.

See also

DPRLoadXMLAttachment on page 188

DPRLoadedXML2Formset on page 184

DPRSendFormsetXML on page 251

DPRUpdateFromMRL on page 282

DPRFilterFormsetForms on page 122

DPRSortFormsetForms on page 260

DPRGetInitValue

Use this rule to look up an INI value and add it as an attachment variable to the input and output queues. This rule is useful when you are running Java rules in IDS version 1.8 which need INI values from the DAP.INI or other INI file.

Syntax

```
long _DSIAPI DPRGetInitValue ( DSIHANDLE hdsi,
                               char * pszParms,
                               unsigned long  ulMsg,
                               unsigned long  ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
INIGROUP	The name of the INI control group to retrieve the value from.
INIOPTION	The name of the INI option to retrieve the value from.
INIVALUE	The name of the attachment variable generated in the input/output queues which will hold the INI value.

Attachment outputs

Variable	Description
RESULTS	Success or failure

The DPRGetInitValue rule can also take arguments instead of the attachment variables specified above. The arguments override the input attachment variables.

Here is an example of a request type that passes the arguments to the rule:

```
[ReqType:TEST8]
function = atcw32->ATCLogTransaction
function = dsijrule->JavaRunRule,;com/docucorp/ids/rules/
IDSTransactionRule;;static;reportTimes;INCLUDEMS
function = atcw32->ATCLoadAttachment
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSetConfig
function = dprw32->DPRGetInitValue,SQLPROCEDURES,FILE,PROCFILE
function = dprw32-
>DPRGetInitValue,SQLPROCEDURES,GLOBALPATH,SQLPROCEDURES_GLOBALPATH
function = dsijrule->JavaRunRule,;com/docucorp/ids/rules/
SQLDBRule;Obj8;transaction;SQLDecryptProc;
```

DPRGetOneWipRecord

Use this rule to return all of the WIP index fields as attachment variables. This rule is very similar to the DPRGetWipList rule except this rule returns the WIP index for a specific record set by the RECNUM or UNIQUE_ID. The WIP index fields are returned as attachment variables.

You can use this rule with the WIP Edit plug-in when a WIP record is locked. This rule lets you view the index information for the record before taking any action to unlock the record or postpone changes.

Syntax

```
long _DSIAPI DPRGetOneWipRecord ( 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

Attachment variables

This rule expects these attachment variables:

Attachment	Description
RECNUM or UNIQUE_ID	Lets the rule find the correct WIP record.

Errors

This rule can return these messages:

Message	Description
DPR0001	The attachment variable was not found.
DPR0039	The API failed.
DPR0022	The system could not add the attachment variable.

See also

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRGetWipList on page 150

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRGetRecipients

Use this rule to return a list of recipients for the form set. This rule runs on the DSI_MSGRUNF message. This rule uses the DAP.INI file.

Syntax

```
long _DSIAPI DPRGetRecipients ( 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

Attachment variables

This rule expects these attachment variables:

Variable	Description
USERID	ID of the requester
ARCKEY	Archive key value used to retrieve the data

This rule creates an attachment record called RECORDS with these values:

Record	Description
RECIPIENT	The name of the recipient from POL file
DESCRIPTION	The recipient description, if specified in the Recip_Names control group or same as recipient name in POL file. The application should use DESCRIPTION for displaying the recipient list.

This rule also creates an attachment variable called RESULTS, which copies the input attachment into the output attachment.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate the variable named *VARIABLE* in the attachment list
DPR0010	The system encountered an internal error of unknown type. The call by LOCATION to APINAME failed
DPR0011	The attachment field *VARIABLE* does not contain valid data

Message	Description
DPR0019	Cannot retrieve data into the FILE file. ARCRetrieveDoc API failed. CATALOGKEY = CATALOGKEY CARID = CARID.
DPR0020	DSLoadFormList API failed on the file named *FILE*
DPR0022	Cannot add the variable named *VARIABLE* to the attachment list.
DPR0025	Cannot add the variable named *VARIABLE* to the attachment record RECORD

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

Attachment variables

Variable	Description
CONFIG	Configuration

Attachment outputs

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.

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

Variable	Description
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

This rule can return these messages:

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
```

See also DPRModifyUser on page 201

DPRGetWipList

Use this rule to retrieve a list of WIP records for a specified user ID. It returns the list by adding every field of each record into the attachment in your queue. You can specify the starting record and the maximum records number to return.

The array of the fields is defined in the WIP DFD file or in DBFFields if the WIP DFD file is missing.

Syntax

```
long _DSIAPI DPRGetWipList ( DSIHANDLE hdsi,
                             char * pszParms,
                             unsigned long  ulMsg,
                             unsigned long  ulOptions )
```

Parameters

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

Attachment variables

This rule expects these attachment variables:

Variable	Description
USERID	The ID of the queue name for user.
STARTRECORD	The starting record number (default is 1).
MAXRECORDS	The maximum number of records to be retrieved (default is 20).
STATUS	The current status for sorting the WIP list.
CURRUSER	If you specify this input attachment variable: `CURRUSER=~UNKNOWN~` the rule lists the records that do not belong to users found in the valid user list.

Attachment outputs

The output attachment variables include:

Variable	Description
WIP	The WIP status generated from WIP option in the Status_CD control group
Approve	The status generated from Approve option in the Status_CD control group
Reject	The status generated from Reject option in the Status_CD control group
Records	The number of selected records.

Variable	Description
RECORDSX. FieldName	The field name for selected single or multiple records, where the affix X (WIPSX.FieldName) is the number of selected WIP records, counting from 1 to RECORDS; FieldName is the field name as defined WIP DFD file. If the DFD file is missing, default field names are used, such as Key1, Key2, KeyID, RecType, and so on.

Request types

ReqType = WLT

The requested type is required in the DOCSERV.INI file. Here is an example:

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

INI options

You can use these INI options with this rule:

Option	Control group	Description
File	WIPData	Specifies the name of the WIP file.
Path	WIPData	Specifies the path to the WIP file.
MaxWIPRecords	WIPData	Specifies the maximum records to be read into the processQ. Prevents it from slowing down because of the volume of records.
File	UserInfo	Specifies the name of the userinfo file.
Path	UserInfo	Specifies the path to the userinfo file. If the userinfo file is missing, USERID is added in the user list.
WIP	Status_CD	Specifies the WIP status code.
Approve	Status_CD	Specifies the approve status code.
Reject	Status_CD	Specifies the reject status code.

Here is an example:

```
< WIPData >
    File    = WIP
    Path    = mstrres\sampco\wip\
    MaxWIPRecords = 200
< UserInfo >
    File    = userinfo
    Path    = mstrres\
< Status_CD >
    WIP     = W
    Approve = AP
    Reject  = RJ
```

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #APINAME,# failed.
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0025	Cannot add variable #VARIABLE,# to the attachment record #RECORD,#.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRApproveWipRecords on page 73

DPRCheckWipRecords on page 88

DPRFindWipRecordsByUser on page 125

DPRGetWipFormset on page 153

DPRGetWipRecipients on page 156

DPRSearchWip on page 246

DPRUpdateWipRecords on page 287

DPRGetWipFormset

Use this rule to retrieve a form set from the WIP record. If the record exists, it loads the WIP form set by loading POL and NA files. The form set handle is added into the attachment for other processes, such as printing out as PDF, HTML, or XML.

Syntax

```
long _DSIAPI DPRGetWipFormset ( DSIHANDLE hdsi,
                                char * pszParms,
                                unsigned long  ulMsg,
                                unsigned long  ulOptions )
```

Parameters

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

This rule expects these attachment variables:

Variable	Description
FieldName	The value of the field as defined in the WIP DFD file or default fields, such as Key1, Key2, KeyID, RecType, and so on. You must list all fields even if some do not have values.

The output attachment variables include:

Variable	Description
DPRFormset	The form set handle used to extract the form set for printing.

Request types

ReqType = WFS

The requested type is required in the docserv.ini file. Here is an example:

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

INI options

You can use these INI options with this rule:

Option	Control group	Description
FormLib	MasterResource	Specifies the path to the forms.
ImageExt	Control	Specifies the type of image file.
LogoExt	Control	Specifies the type of logo image.
File	WIPData	Specifies WIP file name
Path	WIPData	Specifies the path to the WIP file
MaxWIPRecords	WIPData	Specifies the maximum records to be read into the processQ. Prevents it from slowing down because of the volume of records.

Here is an example:

```
< MasterResource >
   FormLib  = mstrres\sampco\forms\
< Control >
   ImageEXT   = .fap
   LogoExt    = .log
< WIPData >
   File    = WIP
   Path     = mstrres\sampco\wip\
```

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0020	DSLoadFormList API failed on file #FILE,#.
DPR0021	DSLoadNAFormset API failed on file #FILE,#.
DPR0022	Cannot add variable #BARIABLE,# to the attachment list.
DPR0024	Cannot create DSI variable #VARIABLE,#.
DPR0025	Cannot add variable #VARIABLE,# to the attachment record #RECORD,#
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also	DPRApproveWipRecords on page 73
	DPRCheckWipRecords on page 88
	DPRGetWipList on page 150
	DPRGetWipRecipients on page 156
	DPRSearchWip on page 246
	DPRUpdateWipRecords on page 287

DPRGetWipRecipients

Use this rule to retrieve a list of recipients from the POL file for the selected WIP record.

Syntax

```
long _DSIAPI DPRGetWipRecipients ( DSIHANDLE hdsi,
                                   char * pszParms,
                                   unsigned long  ulMsg,
                                   unsigned long  ulOptions )
```

Parameters

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

Attachment variables

This rule expects these input attachment variables:

Variable	Description
Fields	An array of the fields in the selected record as defined in the WIP.DFD file or in DBFFields if there is no WIP.DFD file.

Attachment outputs

The output attachment variables include:

Variable	Description
RECORDS	The number of recipients in the recipient list.
RECORDSX. RECIPIENT	The name of the recipient from the POL file.
RECORDSX. DESCRIPTION	The recipient description specified in the Recip_Names control group. If omitted, it defaults to the recipient name where the affix X (in RECORDSX.RECIPIENT and RECORDSX.DESCRIPTION) is the index number of the recipients counting from one (1) to RECORDS.

Request types

```
ReqType = WRC
```

The requested type is required in the docserv.ini file. Here is an example:

```
< ReqType:WRC >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRGetWipRecipients
```

INI options

Use these INI options with this rule:

```
< WIPData >
   File       = WIP
   Path       = mstrres\sampco\wip
< Recip_Names >
   AGENT      = 001,Agent Copy
   HOME OFFICE=002,Home Office Copy
   INSURED    =003,Insured Copy
```

Option	Description
WIPData control group	
File	Specifies the WIP file name.
Path	Specifies the path to the WIP file.
Recip_Names control group	
(*recipients*)	Include the recipient name on the left and the description on the right of the equals sign.

Returns

Success or failure

Errors

Error	Description
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0007	The INI option #INIOPTION,# could not be located in the control group #INIGROUP,#.
DPR0020	DSLoadFormList API failed on file #FILE,#.
DPR0022	Cannot add the variable #BARIABLE,# to the attachment list.
DPR0025	Cannot add the variable #VARIABLE,# to the attachment record #RECORD,#
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also	DPRAddWipRecord on page 71
	DPRApproveWipRecords on page 73
	DPRAssignWipRecord on page 78
	DPRCheckWipRecords on page 88
	DPRDeleteWipRecord on page 105
	DPRDelMultiWipRecords on page 109
	DPRGetWipList on page 150
	DPRGetWipFormset on page 153
	DPRSearchWip on page 246
	DPRUpdateWipRecords on page 287

DPRIni2XML

Use this rule to add items from the INI file to the XML tree found in the WIPXMLVAR variable. The WIPXMLVAR variable is created by the DPRWipIndex2XML rule. The DPRIni2XML rule must be run after the DPRIndex2XML rule

Syntax

```
long _DSIAPI DPRIni2XML ( 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

IDS can use the DPRIni2Xml rule to pass an encrypted password to the WIP Edit plug-in to provide authentication when saving data back to IDS.

```
< INI2XML >
    HTTPUserID     = encrypteduserID
    HTTPPassword   = encryptedpassword
```

You can also use the cryruw32 program to create an encrypted value that can be understood by the WIP Edit plug-in. This lets you avoid putting passwords in the INI file where they can easily be read. For instance, if you enter this from the command line:

```
cryruw32.exe password
```

you will see the output similar to the following:

```
Encrypted string (2XAUnkxUYlx7i5AnQ4m4E1m00)
```

INI options

Include this INI option:

```
< INI2XML >
    Name of node in XML = Value of Node
```

Attachment variables

This rule expects no specific attachment variables, however, you can include the value of an attachment variable in the XML tree if you precede the option name with an octothorp (#).

Here is an XML example:

```
< INI2XML >
    PutURL  = localhost
```

Errors

This rule can return these messages:

Message	Description
DPR0001	Failed to get attachment variable.

See also

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRGetOneWipRecord on page 143

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRInit

Use this rule to initialize the Documaker subsystem and start virtual memory management and file caching. This rule initializes VMM, FAP, DB, and loads the DAP.INI file. The rule also initializes FAP file cache based on rule parameters. If you omit the rule parameter, the rule sets the number of cached FAP files to 1000.

This rule also sets the Documaker trace file name, based on the TraceFile option in the Data control group. The default trace file name is TRACE.

This rule runs on the DSI_INIT and DSI_TERM messages. On termination, all of the above is terminated.

Syntax

```
long _DSIAPI DPRInit ( 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

This rule loads the DAP.INI file. You can specify the name and location of the DAP.INI file you want to use as shown here:

```
< ReqType:INI >
    function = DPRW32->DPRInit,500,d:\docserv\dap.ini
```

Separate parameters with commas.

The first parameter specifies the file cache. The default FAP file cache is 1000. The second parameter specifies where to find the INI file. *DAP.INI* is the default file name.

NOTE: This approach does not work with the DPRCoLogin rule. Use the DPRLogin rule instead.

Returns

Success or failure

DPRInitLby

Use this rule to initialize the Library Manager. The rule runs on DSI_INIT and DSI_TERM messages. On termination, this rule terminates the Library Manager.

You do not have to use this rule if your Documaker environment does not use the Library Manager to store resources. Place this rule after the DPRInit rule in the rule list.

NOTE: Keep in mind that, with Shared Objects 11.0, Patch 22 and higher, it is no longer necessary to specify this rule. The DPRSetConfig rule will automatically do what the DPRInitLby rule used to do.

This rule uses the DAP.INI file.

Syntax

```
long _DSIAPI DPRInitLby ( 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

Returns Success or failure

See also DPRInit on page 161

DPRSetConfig on page 255

DPRLbyCopy

Use this rule to copy a resource from one location to another, such as from one library to another. Keep in mind...

- The resource and destination file names *must match*.
- The config value for the resource *must differ* from the config value for the destination.

If the resource you are copying does not exist in the destination library, it will be added as a new resource with a version and revision of *00001*. If the resource being copied exists in the destination, it will be added as a new version and revision; this is true regardless of what version and revision was specified for the resource or destination file names. The DRPLbyCopy rule supports this WebDav command:

Use this command	To
copy [source] [destination]	Copies a resource from one location to another.

Syntax

```
long _DSIAPI DPRLbyCopy ( 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

Attachment variables

Variable	Description
LBYFILE	The resource you want to use for the copy operation. A full path and file name generated by DPRLbyGet rule, which should be run before this rule in the WEBDAVCOPY request type.

DESTINATIONURI	A URI that contains the destination of the resource you want to copy. Here are some examples of destination URIs:
	`/jdoe/dms1/ddt/master.ddt`
	`/jdoe/DMS1/DDT/`
	`MASTER_0000100001_20030707.DDT`

OVERWRITE	(Optional) An overwrite flag indicator. A *T* means to overwrite the destination if it exists. An *F* indicates the rule will fail if the destination exists. Reserved for future use.

Variable	Description
USERID	(Optional) The user ID you want to use for the copy operation. If this attachment variable exists, it overrides the user ID provided in the destination URI. If the user ID is omitted from the attachment variable and the destination URI, the rule will fail.
ARCEFFECTIVEDATE	(Optional) An archive effective date. Here is an example of the format you should use: `MM/DD/YYYY` If this variable exists, its value is used as the archive effective date for the copy operation. Otherwise, the rule uses the current date for the archive effective date.

Attachment outputs

Variable	Description
RESULTS	SUCCESS or ERROR.
WEBDAVERRORCODE	This attachment variable only exists if RESULTS equals ERROR. It can contain one of these values: 403 (Webdav 'forbidden' error code) - The source and destination URIs are the same. 409 (Webdav 'conflict' error code) - The resource cannot be created at the destination. 412 (Webdav 'precondition failed' error code) - The overwrite header is F and the state of the destination resource is non-null. 420 (Webdav 'method failure' error code) - An internal error or memory error occurred. 423 (Webdav 'locked' error code) - The destination resource was locked.

DPRLbyDelete

Use this rule to remove a resource or collection from Library Manager. This rule can remove a resource file by version and revision or by name, in which case the rule removes the latest version and revision for the resource file you specified.

If the resource you specify is a collection (file type), all resources for the collection will be removed, provided none are locked. This rule supports this WebDav command:

Use this command	To
delete [path] file	Delete a resource.

Syntax

```
long _DSIAPI DPRLbyDelete ( 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

Attachment variables

Variable	Description
RESOURCEURI	The resource URI of the resource you want to delete from Library Manager. Here is an example of the format you should use: `/userid/config/filetype/resource` Here are some examples: `/jdoe/dms1/ddt/master.ddt` `/jdoe/DMS1/DDT/MASTER_0000100001_20030707.DDT` If the resource file name in RESOURCEURI does not contain version, revision, and archive effective date information, the DPRLbyDelete rule tries to delete the last version and revision of the file resource you specified.
RESULTS	(Optional) This variable is only generated by the DPRLby rules running prior to this rule in the same request type, such as the DPRLbyGet and DPRLbyCopy rules running in the WEBDAVMOVE request type. If this variable exists and is set to Error — indicating either the DPRLbyGet or DPRLbyCopy rule failed — this rule will not execute.

Variable	Description
WEBDAVERRORCODE	(Optional) This variable is only generated by DPRLby rules running prior to this rule in the same request type, such as the DPRLbyGet and DPRLbyCopy rules running in the WEBDAVMOVE request type. If this variable exists — indicating that either the DPRLbyGet or DPRLbyCopy rule failed — this rule will not execute.

Attachment outputs

Variable	Description
RESULTS	Success or Error.
WEBDAVERRORCODE	This attachment variable is only present if RESULTS equals Error. It can contain one of these values: 404 - (WebDav ’not found’ error code) - The RESOURCEURI cannot be found. 409 - (WebDav ’conflict’ error code) - The RESOURCEURI specified is invalid. 420 - (WebDav ’method error’ error code) - An internal API error or memory error occurred. 423 - (WebDav ’locked’ error code) - The resource is locked.

DPRLbyGet

Use this rule to get or check out a resource file from Library Manager. This rule can retrieve a resource file by version and revision or by name, in which case it retrieves the latest version and revision for the resource specified. This rule supports these WebDav commands:

Use this command	To
get [path] file	Get a resource.
head [path] file	Get header info for a resource. (currently works same as get)

Syntax

```
long _DSIAPI DPRLbyGet ( 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

Arguments

Parameter	Description
CHECKOUT	If you include this rule argument and set its value to Yes, the rule tries to check out (get and lock) the resource specified. This is useful for configuring this rule for a check-out or get request type.

Attachment variables

Variable	Description
RESOURCEURI	The resource URI of the resource you want to retrieve from Library Manager. Here is an example of the format for the resource URI: `/userid/config/filetype/resource` Here are some examples: `/jdoe/dms1/ddt/master.ddt` `/jdoe/DMS1/DDT/MASTER_0000100001_20030707.DDT` If the resource file name does not contain version, revision, and archive effective date information, the DPRLbyGet rule retrieves the last version and revision for the resource specified. Use the DPRLbyGet rule to get or check out a resource from Library Manager.

Variable	Description
USERID	(Optional) The user ID you want to use for the get operation. If you include this attachment variable, it overrides the user ID provided as part of the resource URI. If the user ID is missing as an attachment variable and in the resource URI, the rule will fail.

Attachment outputs

Variable	Description

PROPERTIES	A rowset with a row for the resource specified in RESOURCEURI. The row contains the following properties for a file resource:
	supportedlock - If locking is allowed, this XML string appears:
	`property:  <lockentry>`
	`    <lockscope>`
	`        <exclusive/>`
	`    </lockscope>`
	`    <locktype>`
	`        <write/>`
	`    </locktype>`
	`    </lockentry>`
	getContentLanguage - currently returns *en_US*.
	resourcetype - blank if the resource is a file, otherwise *collection* if the resource is a file type/directory.
	displayname - the display name of the resource.
	HREF - the resource URL for this resource
	getlastmodified - a date and time indicating when the resource was last modified. This is a long value that contains the number of milliseconds since January 1, 1970.
	getContentLength - currently zero (0) because there is no support for retrieving the file size of a document stored in Library Manager.
	If a resource is locked these additional properties are returned:
	LOCKOWNER - The user ID that set the lock.
	LOCKSCOPE - The scope of the lock (exclusive).
	LOCKSUBJECT - The name of the resource locked.
	LOCKDEPTH - The depth of the resource locked (0).
	LOCKTYPE - The type of lock (write).
	LOCKTIMEOUT - The time-out value after which the lock will expire (infinity).
	LOCKTOKEN - A unique ID that identifies the resource locked.
	This rowset is only present if RESULTS contains SUCCESS.

RESULTS	SUCCESS or ERROR

Variable	Description
WEBDAVERRORCODE	This attachment variable is only present if RESULTS equals ERROR. It can contain one of these values: 404 - (WebDav 'not found' error code) - The RESOURCEURI cannot be found. 409 - (WebDav 'conflict' error code) - The RESOURCEURI specified is invalid. 420 - (WebDav 'method error' error code) - An internal API error or memory error occurred. 423 - (WebDav 'locked' error code) - The resource is locked and the system attempted a check out operation.

DPRLbyLock

Use this rule to lock a resource in Library Manager. This rule supports the following WebDav command:

Use this command	To
lock [path] file	Locks a resource.

Syntax

```
long _DSIAPI DPRLbyLock ( 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

Attachment variables

Variable	Description

RESOURCEURI	The resource URI of the resource you want to lock in Library Manager. Here is an example of the format for a resource URI:
	`/userid/config/filetype/resource`
	Here are some examples:
	`/cjr/rpex1/ddt/master.ddt`
	`/jdoe/RPEX1/DDT/MASTER_0000100001_20030707.DDT`
	If the resource file name in RESOURCEURI does not contain version, revision, and archive effective date information, the DPRLbyLock rule tries to lock the last version and revision of the file resource you specified.

USERID	(Optional) The user ID you want to use for the lock operation. If this attachment variable is present, it overrides the user ID provided as part of the resource URI. If the user ID is omitted from the attachment variable and from the resource URI, the rule will fail.

Attachment outputs

Variable	Description
LOCKOWNER	The user ID that owns the lock.
LOCKSCOPE	The scope of the lock (exclusive).
LOCKSUBJECT	The name of the resource locked.
LOCKDEPTH	The depth of the resource locked (0).

Variable	Description
LOCKTYPE	The type of lock (write).
LOCKTIMEOUT	The time-out value after which the lock will expire (infinity).
LOCKTOKEN	A unique ID that identifies the resource locked.
RESULTS	SUCCESS or ERROR.
WEBDAVERRORCODE	This attachment variable only exists if RESULTS equals ERROR. It can contain one of these values: 404 - (WebDav 'not found' error code) - The RESOURCEURI cannot be found. 409 - (WebDav 'conflict' error code) - The RESOURCEURI specified is invalid. 420 - (WebDav 'method error' error code) - An internal API error or memory error occurred. 423 - (WebDav 'locked' error code) - The resource is already locked.

DPRLbyMKCol

Use this rule to create a collection in Library Manager. This rule supports this WebDav command:

Use this command	To
mkcol	Not supported by Library Manager.

Keep in mind the mkcol command is not supported by Library Manager. You cannot make new collections (file types) in Library Manager without first adding a resource of that type.

This rule always returns RESULTS set to *ERROR* and WEBDAVERRORCODE set to *unsupported media type*.

Syntax

```
long _DSIAPI DPRLbyMKCol ( 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

Attachment variables

None

Attachment outputs

Variable	Description
RESULTS	ERROR.
WEBDAVERRORCODE	This attachment variable only exists if RESULTS equals ERROR, which in this case is always true. It contains the following value: 415 - (WebDav 'unsupported media type' error code) - The server does not support or understand the mkcol request type.

DPRLbyOptions

Use this rule to display the WebDav commands supported by Library Manager. This rule supports this WebDav command:

Use this command	To
options [path / url]	Display the options available for a path or URL.

This rule displays the following WebDav commands that are supported by Library Manager:

options	get	head
propfind	propgetall	lock
unlock	delete	copy
move	proppatch	mkcol

Syntax

```
long _DSIAPI DPRLbyOptions ( 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

Attachment variables

None

Attachment outputs

Variable	Description
OPTIONS	A comma-delimited string of WebDav commands supported by Library Manager.
RESULTS	SUCCESS.

DPRLbyPropFind

Use this rule to return:

- The properties for a file if the resource you specify is a file
- A list of files and their properties if the resource you specify is a collection or file type (FAP, LOG, DDT, DAL, FOR, GRP, BDF)
- A list of collections or file types if the resource you specify is root (/).

This rule supports these WebDav commands by querying Library Manager for the configuration specified:

Use this command	To
ls [path]	List the contents of a collection.
cd [path]	Change directories.
propget [path] [property]	Get a property.
propfind [path] [property]	Find a property.
propgetall [path]	List all properties for a resource.

Syntax

```
long _DSIAPI DPRLbyPropFind ( 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

Attachment variables

Variable	Description

RESOURCEURI	A resource URI specifying a user ID, config, file type, and resource. Here are some examples of resource URIs:
	`/userid/config/filetype/resource/`
	`/userid/config/filetype/`
	`/userid/config/`
	`/userid/`

DEPTH	Enter a depth of 0ne (1) for collections or file types in Library Manager. Enter a depth of zero (0) for file resources.

Attachment outputs

Variable	Description

PROPERTIES	A rowset of rows that match each of the file resources available for a particular collection/file type. If DEPTH is one (1) and RESOURCEURI specifies a collection or file type in Library Manager, the PROPERTIES rowset returns a row for each resource available in the collection/file type.
	If DEPTH is zero (0) and RESOURCEURI specifies a file resource, the PROPERTIES rowset returns a single row with the properties for the resource you specified.
	Each row in the PROPERTIES rowset contains the following properties for a file resource:
	supportedlock - If locking is allowed, this XML string appears:
	`property:  <lockentry>`
	`    <lockscope>`
	`        <exclusive/>`
	`    </lockscope>`
	`    <locktype>`
	`        <write/>`
	`    </locktype>`
	`    </lockentry>`
	getContentLanguage - currently returns *en_US*.
	resourcetype - blank if the resource is a file, otherwise *collection* if the resource is a file type/directory.
	displayname - the display name of the resource.
	HREF - the resource URL for this resource
	getlastmodified - the date and time indicating when the resource was last modified. This is a long value that contains the number of milliseconds since January 1, 1970.
	getContentLength - currently zero (0) because there is no support for retrieving the file size of a document stored in Library Manager (reserved for future use).
	If a resource is locked these additional properties are returned:
	LOCKOWNER - The user ID that set the lock.
	LOCKSCOPE - The scope of the lock (exclusive).
	LOCKSUBJECT - The name of the resource locked.
	LOCKDEPTH - The depth of the resource locked (0).
	LOCKTYPE - The type of lock (write).
	LOCKTIMEOUT - The time-out value after which the lock will expire (infinity).
	LOCKTOKEN - A unique ID that identifies the resource locked.
	This rowset is only present if RESULTS contains SUCCESS.

RESULTS	SUCCESS or ERROR

Variable	Description
WEBDAVERRORCODE	This attachment variable is only present if RESULTS equals ERROR. It can contain one of these values: 404 - (WebDav 'not found' error code) - The RESOURCEURI cannot be found. 409 - (WebDav 'conflict' error code) - The RESOURCEURI specified is invalid. 420 - (WebDav 'method error' error code) - An internal API error or memory error occurred.

INI options

Use these options in the DAP.INI file to see a listing of the configurations that support Library Manager.

```
< LbyConfigs >
   Config  = RPEX1
   Config  = RPEX2
```

DPRLbyPropPatch

Use this rule to set or remove properties defined on the resource identified by the RESOURCEURI. This rule supports this WebDav command:

Use this command	To
proppatch	Not supported by Library Manager.

The proppatch command is not supported by Library Manager. You cannot modify the properties for records in Library Manager. This rule always returns RESULTS set to *ERROR* and WEBDAVERRORCODE set to *method not allowed*.

Syntax

```
long _DSIAPI DPRLbyPropPatch ( 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

Attachment variables

None

Attachment outputs

Variable	Description
RESULTS	ERROR.
WEBDAVERRORCODE	This attachment variable only exists if RESULTS contains ERROR, which in this case is always true. It will contain this value: 405 - (WebDav 'method not allowed' error code) - The server does not allow or support this method.

DPRLbyPut

Use this rule to add a new resource or to check in (unlock and put) an existing resource into Library Manager. You can add a new resource or put an existing resource into Library Manager.

If the resource is new, its version and revision will be 00001. If the resource is an existing one and it is locked by the same user ID performing the put operation, the resource will be put into Library Manager with a new version and revision.

This rule supports this WebDav command:

Use this command	To
put [path]	Put a file into Library Manager.

Keep in mind that if a put operation is attempted on an existing resource and the version and revision specified is not the latest one, the put operation will fail. The system only supports put operations for new documents or for the last existing version and revision which must be locked prior to the put call.

Syntax

```
long _DSIAPI DPRLbyPut ( 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

Attachment variables

Variable	Description
RESOURCEURI	A resource URI specifying the resource you want to place into Library Manager. Here is an example of the format of the URI: `/userid/config/filetype/resource/` Here are some examples: `/cjr/rpex1/ddt/master.ddt` `/jdoe/RPEX1/DDT/MASTER_0000100001_20030707.DDT` Keep in mind that if the resource file name in RESOURCEURI does not contain version, revision, and archive effective date information, the DPRLbyPut rule tries to put the last version and revision of the file resource you specified.

Variable	Description
USERID	(Optional) The user ID you want to use for the put operation. If this attachment variable is present, it overrides the user ID provided in the resource URI. If the user ID is missing from the attachment variable and from the resource URI, the rule will fail. For put operations with an existing resource, the user ID must match that of the locked record or the put operation will fail.
ARCEFFECTIVEDATE	(Optional) An archive effective date. Here is the format for this attachment variable: `MM/DD/YYYY` If this variable is present, its value is used as the archive effective date for the put operation. If it is missing, the rule uses the current date as the archive effective date.

Attachment outputs

Variable	Description
RESULTS	SUCCESS or ERROR.
WEBDAVERRORCODE	This attachment variable only exists if RESULTS equals ERROR. It can contain one of these values: 404 - (WebDav ’not found’ error code) - The RESOURCEURI cannot be found. 409 - (WebDav ’conflict’ error code) - The RESOURCEURI specified is invalid. 420 - (WebDav ’method error’ error code) - An internal API error or memory error occurred. 423 - (WebDav ’locked’ error code) - The resource is locked under a different user ID.

DPRLbyUnlock

Use this rule to unlock a resource file in a library maintained by Library Manager. This rule supports this WebDav command:

Use this command	To
unlock [path] file	Unlock a resource.

Syntax

```
long _DSIAPI DPRLbyUnlock ( 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

Attachment variables

Variable	Description
RESOURCEURI	The resource URI of the resource you want to unlock in Library Manager. Here is an example of the format for a resource URI: `/userid/config/filetype/resource` Here are some examples: `/cjr/rpex1/ddt/master.ddt` `/jdoe/RPEX1/DDT/MASTER_0000100001_20030707.DDT` If the resource file name in RESOURCEURI does not contain version, revision, and archive effective date information, the DPRLbyUnlock rule tries to unlock the last version and revision of the file resource specified.
USERID	(Optional) The user ID you want to use for the unlock operation. If this attachment variable is present, it overrides the user ID provided in the resource URI. If the user ID is omitted from the attachment variable and from the resource URI, the rule fails. If the user ID does not match the one for the locked record, the rule fails.

Attachment outputs

Variable	Description
RESULTS	SUCCESS or ERROR.

Variable	Description
WEBDAVERRORCODE	This attachment variable only exists if RESULTS equals ERROR. It can contain one of these values: 404 - (WebDav 'not found' error code) - The RESOURCEURI cannot be found. 409 - (WebDav 'conflict' error code) - The RESOURCEURI specified is invalid. 420 - (WebDav 'method error' error code) - An internal API error or memory error occurred. 423 - (WebDav 'locked' error code) - The resource is locked by another user.

DPRLoadDPA

Use this rule to create an internal form set from a DPA file stored in Documanage. The system expects the DPRRetrieveFormset and DPRPrint rules to follow this rule.

This rule splits the functionality of the DPRRetrieveDPA rule so you can insert the DPRInitLby rule in the rule list. Unlike the DPRRetrieveDPA rule, you must call the DPRRetrieveFormset rule.

Syntax

```
long _DSIAPI DPRLoadDPA ( DSIHANDLE hdsi,
                          char * pszParms,
                          ULONG  ulMsg,
                          ULONG ulOptions )
```

Parameters

Parameter	Description
DSIHANDLE hdsi	Pointer to the rules data
char * pszParms	Pointer to rule parameter string
ULONG ulMsg	DSI_ message
ULONG ulOptions	Options

Attachment variables

This rule expects these attachment variables:

Variable	Description
DMSARCFILE	The path to the DPA file that has been retrieved.

Attachment outputs

This rule creates these attachment variables:

Variable	Description
OLDCONFIG	CONFIG is set to the value in the DPA file during the run forward. The run reverse step returns it to its original value.

The CONFIG value is changed to the value stored in the DPA file when the rule is run forward. When run in reverse, the system changes the CONFIG value back to its original value.

Here is a sample rule list:

```
[ ReqType:BIA ]
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = briutls->GUTSwapAttachments
    function = pobrs->POWInputSession
    function = pobrs->POWHandleSession
    function = pobrs->POWAccessPage
    function = Tpdw32->TPDCreateFormset
    function = mtcw32->MTCLoadFormset
    function = dprw32->DPRLoadDPA
    function = dprw32->DPRInitLby
    function = dprw32->DPRRetrieveFormset
    function = dprw32->DPRPrint
    function = pobrs->POWPostConversion
    function = briutls->GUTSetUIConfig
    function = pobrs->POWOutputSession
    function = atcw32->ATCUnloadAttachment
```

See also DPRInitLby on page 162

DPRRetrieveDPA on page 230

DPRRetrieveFormset on page 231

DPRLoadedXML2Formset

Use this rule to load an XML tree in memory which is located in the DSI variable DPRXMLFORMSET into a FAP form set and put it into the DSI variable DPRFORMSET. If the DPRXMLFORMSET variable is missing, this rule does nothing and no error message appears.

NOTE: Use this rule with the DPRLoadXMLAttachment rule.

Syntax

```
long _DSIAPI DPRLoadedXML2Formset ( DSIHANDLE hInstance,
                              char * pszParms,
                              unsigned long  ulMsg,
                              unsigned long  ulOptions )
```

Parameters

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

This rule runs on the DSI_RUNF message, destroys the FAP form set, and deletes the DSI variable DPRFORMSET on the DSI_RUNR message.

See also

DPRLoadXMLAttachment on page 188

DPRSendFormsetXML on page 251

DPRUpdateFromMRL on page 282

DPRFilterFormsetForms on page 122

DPRSortFormsetForms on page 260

DPRGetFormList on page 137

DPRGetHTMLForms on page 141

DPRLoadFAPImages

Use this rule to load all FAP files used in a form set. Be sure to first create the form set using a rule such as the DPRRetrieveFormset rule.

This rule is useful when you are using the DPRDelBlankPages or DPRRotateFormsetPages rules with form sets retrieved from Documaker archives or from import files.

Syntax

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

Parameters

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

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0017	DSI variable VARIABLE cannot be located. Usually means the form set was not created correctly because of some other error
DPR0036	The DSI variable #VARIABLE,# does not contain a valid FAP form set

See also

DPRRetrieveFormset on page 231

DPRDelBlankPages on page 102

DPRRotateFormsetPages on page 234

DPRLoadImportFile

Use this rule to load an import file into a form set. The import file must meet the specifications outlined for the Documaker system.

NOTE: See the Documaker Supervisor Guide for more information on import file formats.

Syntax

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

Parameters

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

On the RUNF message, this rule loads the import file into a form set and creates the DSI variable DPRFORMSET with this form set handle. On the RUNR message, this rule destroys the form set and removes DSI variable.

Attachment variables

The system only creates the DPRFORMSET value if the load was successful. This rule expects these attachment variables:

Variable	Description
IMPORTFILE	The name of the import file
PRINTFILE	The name of the output file, if omitted the system will generate it.
FILETYPE	Set to CMBNA to import combined NA/POL files. If this variable is blank or omitted, the system looks into the import file to see how the file begins. • If the file begins with <*?xml*, the system assumes it is an XML file import. • If the file begins with *WIP*=, the system assumes it is a combined NA/POL file import. • If the file begins with something other than <*?xml* or <*?xml*, the system assumes it is a V2 file import. By leaving the variable blank you can use the same request type and the same attachment variables to import all supported import file types into Documaker.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate attachment variable IMPORTFILE
DPR0010	DLL version mismatch, internal API failure
DPR0018	Cannot load font cross reference file
DPR0022	Cannot add variable RESULTS to the attachment
DPR0024	Cannot create DSI variable DPRFORMSET
DPR0026	Cannot load import file
DPR0027	Cannot load the form set definition (FORM.DAT) file
DPR0028	Cannot FAP file for image IMAGENAME.
DPR0029	Loading of the submitted import file resulted in an empty form set.
DPR0030	The combination of group names (GROUPNAME1) and (GROUPNAME2)and the form name (FORMNAME) does not exist in the form definition file. Invalid import data.
DPR0031	Cannot open import file FILE.
DPR0032	No form name provided in the import file.
DPR0033	Cannot parse import file. Line: LINEDATA
DPR0034	Invalid import data. The combination of group names (GROUPNAME1 and GROUPNAME2), the form name (FORMNAME), and the image name (IMAGENAME) does not exist in form definition file.

See also DPRUnloadExportFile on page 278

DPRLoadXMLAttachment

Use this rule to load the XML attachment that is attached to the IDS message XML file and create the DSI variable DPRXMLFORMSET with the handle to this XML document. DPRLoadXMLAttachment is used with the DPRUpdateFromMRL rule.

Syntax

```
long _DSIAPI DPRLoadXMLAttachement ( DSIHANDLE hdsi,
                                     char * pszParms,
                                     unsigned long  ulMsg,
                                     unsigned long  ulOptions )
```

Parameters

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

You can receive the XML file from the IDS message with the delimiter XMLIMPORT or, if you are using a different delimiter to send the XML, you can specify this name as a rule parameter. Here is an example:

```
function = DPRW32->DPRLoadXMLAttachment,MYOWNDELIMETER
```

The delimiter is the value used by the client as the pszAttachName parameter when it executed DSISendFile or DSISendBuffer APIs.

This rule runs on DSI_MSGRUNF.

It destroys the XML tree in memory and deletes the DPRXMLFORMSET DSI variable on DSI_MSGRUNR.

If the attachment to the IDS message is missing this rule does nothing and no error message is produced.

See also

DPRLoadedXML2Formset on page 184

DPRSendFormsetXML on page 251

DPRUpdateFromMRL on page 282

DPRFilterFormsetForms on page 122

DPRSortFormsetForms on page 260

DPRGetFormList on page 137

DPRGetHTMLForms on page 141

DPRLoadXMLFormset

Use this rule to load an XML form set into memory for the DPRPrint rule.

Syntax

```
long _DSIAPI DPRLoadXMLFormset ( DSIHANDLE hdsi,
                                 char * pszParms,
                                 unsigned long  ulMsg,
                                 unsigned long  ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
XMLFORMSET	Specifies the full path and file name of the XML form set.

Attachment outputs

Variable	Description
RESULTS	Success or failure

NOTE: You must pass a CONFIG attachment variable to the DPRSetConfig rule.

See also

DPRPrint on page 211

DPRSetConfig on page 255

DPRUnloadXMLFormset on page 280

DPRLocateOneRecord

Use this rule to locate one record matching the search criteria. If more than one record matches, only the first one is found.

Syntax

```
long _DSIAPI DPRLocateOneRecord ( 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

This rule calls the DPRSearch rule to do the search and then copies the RECORDS1.ARCKEY value in the output attachment into an ARCKEY value in the input attachment, so the DPRRetrieveFormset rule can be used. Parameters to this rule are the FIELDS value for the DPRSearch rule, the default is UNIQUE_ID.

INI options

Use the Debug option with this rule:

```
< DPRLocateOneRecord >
    Debug = No
```

This option defaults to No. If you set this option to Yes, the values before and after encryption and decryption are written to the DPRTRC.LOG file.

Errors

None.

See also

DPRSearch on page 235

DPRRetrieveFormset on page 231

DPRLockWip

Use this rule to lock a WIP record for editing purposes. This prevents one user from overwriting changes made by another user. The lock is by user.

Syntax

```
long _DSIAPI DPRLockWip ( 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

INI options

Use these INI options to determine the response if record is locked. There are three levels: error, warning, and ignore, as this example shows:

```
< WIPLock >
    MatchUserID    = Warning
    UnMatchUserID  = Error
```

Option	Description
MatchUserID	If the record is locked and the user IDs match, present an error, warning, or ignore.
UnMatchUserID	If the record is locked and user IDs do not match, present an error, warning, or ignore.

Attachment variables

Expects these attachment variables.

Variable	Description
OVERRIDELOCK	Lets the rule continue even if it's locked. Used if a warning was returned.
USERID	The user ID you want to lock.
RECNUM or UNIQUE_ID	Lets the rule find the correct WIP record.

Errors

This rule can return these messages:

Message	Description
DPR0041	If the record is locked but should only return a warning.
DPR0042	If the record is locked but should return an error.

See also	DPRAssignWipRecord on page 78
	DPRDeleteWipRecord on page 105
	DPRDpw2Wip on page 112
	DPRFile2Dpw on page 120
	DPRGetOneWipRecord on page 143
	DPRIni2XML on page 159
	DPRUnlockWip on page 281
	DPRWip2Dpw on page 290
	DPRWipIndex2XML on page 296

DPRLog

Use this rule to confirm whether an email was sent by the Internet Document Server.

Syntax

```
long _DSIAPI DPRLog ( 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

This rule stores information in a log file from either the attachment variables or the XML document created by the DPRParseRecord rule. The DPRMail rule puts the RESULTS attachment variable into the output queue.

You can use this information to determine if the email was sent. If no RESULTS variable exists, the DPRMail rule was not executed and no mail was sent.

INI options

Use the DPRLog control group to determine the name of the log file:

```
< DPRLog >
    File = .\mail.log
```

Use the DPRLogVar control group to determine what fields go into the log:

```
< DPRLogVar >
    FieldName =
```

Option	Description
FieldName	DBCOLUMN - If you specify DBCOLUMN, the system uses the XML tree created by the DPRParseRecord rule to get data from the DFD-defined record. ATTACHIN - If you specify ATTACHIN, the system uses the attachment variables from the input queue. ATTACHOUT - If you specify ATTACHOUT, the system uses the attachment variables from the output queue. XPOINTER - If you specify XPOINTER, the system searches the XML tree with XPointer syntax which is created by DPRParseRecord rule. If you enter anything else, the system copies that text into the log file. Enter as many FieldName options as you need.

See also

DPRParseRecord on page 208

DPRMail on page 197

DPRLogin

This is the server login rule—do not run this rule on the client. This rule uses Documaker user information in a database table to verify user IDs and passwords. This rule runs on the DSI_RUNF message.

You can also use the DPRDecryptLogin, DPRDefaultLogin, DPRLoginUser, DPRCheckLogin, and DPRGenerateSeedValue rules to authenticate logins. These rules replace the DPRLogin rule under the Docupresentment authentication model.

This rule uses the DAP.INI file.

NOTE: This rule is only available on Windows 32-bit platforms.

Syntax

```
long _DSIAPI DPRLogin ( 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

Attachment variables

This rule expects these attachment variables:

Variable	Description
USERID	user ID of the requestor
PASSWORD	password of the requestor

This rule creates attachment variables:

Variable	Description
RESULTS	SUCCESS or an error code.
RIGHTS, REPORTTO, SECURITY and USRMESSAGE	values from corresponding columns in the Documaker user table.

If execution is successful, this rule copies the input attachment into the output attachment.

Returns

Success or failure

Errors This rule can return these messages:

Message	Description
DPR0001	Cannot locate the variable *VARIABLE* in the attachment list.
DPR0003	The user information database, *FILENAME*, could not be opened.
DPR0004	The user ID *USERID* is invalid.
DPR0005	The password specified for the user USERID is incorrect.

DPRLoginUser

Use this rule to compare the hash value generated from REALPASSWORD with the hash value of PASSWORD. If the values do not match, an error message is generated.

NOTE: The IDS authentication rules include DPRDecryptLogin, DPRDefaultLogin, DPRLoginUser, DPRCheckLogin, and DPRGenerateSeedValue. These rules replace the DPRLogin rule under the Docupresentment authentication model.

The password is case sensitive. If you do not want to make the password case sensitive in the client application, uppercase the password before it is submitted to IDS.

Syntax

```
Function = dprw32->DPRLoginUser
```

Attachment variables

Variable	Description
LOGINRESULT	If this variable exists and its value is anything other than SUCCESS, the rule does nothing.
USERID	The user ID of the requestor.
PASSWORD	The password of the requestor. It is a hash value.
REALUSERID	The user ID from the userinfo database.
REALPASSWORD	The password from the userinfo database.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate the variable VARIABLE in the attachment list.
DPR0053	Cannot get the random seed value in #LOCATION,#.
DPR0054	Invalid login.

See also

DPRCheckLogin on page 87

DPRDecryptLogin on page 97

DPRDefaultLogin on page 100

DPRGenerateSeedValue on page 128

DPRMail

Use this rule to send email from IDS.

NOTE: This rule is only available on Windows 32-bit platforms.

Syntax

```
long _DSIAPI DPRMail ( 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

Attachment variables

This rule expects:

Variable	Description
Address	email address
Msgbody	body of email
Subject	subject of email
Attachment	file used as attachment no template processing
HTMLBodyFile	file of HTML included in the body
HTMLAttachFile	HTML attachment from template processing

INI options

You can use the following INI options with the email rules. Place all of these options in the DAP.INI file.

```
[ EmailDFD ]
    Path = .\data\attchdfd.dfd
[ Email2IDS ]
    Data = c:\docserv\html
    Message = MsgBody
    Subject = Subject
    Address = Address
[ XML2Body ]
    T1 = C:\DOCSERV\HTML\login.htm
< XML2Attach >
    T2 = C:\DOCSERV\HTML\login.htm
```

Here is an explanation of the various options:

Option	Description
Path	Used by DPRParseRecord to define the attachment record.
Data	Directory to store temporary files.
Message	Maps variables in DFD to the attachment variables expected by Subject and Address.
Subject	The DPRMail and DPRCreateEMailAttachment.
Address	The address.
T1	Sets template for all request type T1.
T2	Sets template for all request type T2.

DPRMapRecipData

Use this rule to map class recipient data into archived documents retrieved using Docupresentment. This rule references the RecipMap2GVM control group (which should correspond to the batch RecipMap2GVM used to create the archive document) and the new Recip2Image control group.

For each occurrence of the form/image (form is optional) specified in RecipMap2GVM, the rule replicates the form set and then propagates the Req and Opt fields to the target image.

You define the target image using the new Image option in the Recip2Image control group. You can specify multiple target images.

Syntax

```
long _DSIAPI DPRMaapRecipData ( 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

Example

Here is an example that uses the following example INI options and data:

```
< RecipMap2GVM >
    Image   = tpinfo
    Opt     = tpinfo1;CORRESPOND_MAILTOADDR01;
    Opt     = tpinfo2;CORRESPOND_MAILTOADDR02;
    Opt     = tpinfo3;CORRESPOND_MAILTOADDR03;
    Opt     = tpinfo4;CORRESPOND_MAILTOADDR04;
    Opt     = tpid;TPID;
< Recip2Image >
    Image   = pvacov1tp
```

NA data segments:

```
...
\NA=pvacov1tp,LN=1,DUP=OFF,SIZE=L,TRAY=U,X=0,Y=0,PA=1,OPT=DLSN\
\FAP\
H,2400,(0,0),(600,400,26400,20400),pvacov1tp
A,H5," ",600,400
F,(6924,3484,7236,13084),(16010,392,352,312),40,CORRESPOND_MAILTOAD
DR01
A,F6," ",0,1,0,0,0," ",0,0,600,0,0,0,0,0,0,0,0,0,0," "
F,(7324,3484,7636,13084),(16010,392,352,312),40,CORRESPOND_MAILTOAD
DR02
A,F6," ",0,1,0,0,0," ",0,0,600,0,0,0,0,0,0,0,0,0,0," "
F,(7735,3482,8047,13082),(16010,392,352,312),40,CORRESPOND_MAILTOAD
DR03
A,F6," ",0,1,0,0,0," ",0,0,600,0,0,0,0,0,0,0,0,0,0," "
```

```
F,(8140,3468,8452,13068),(16010,392,352,312),40,CORRESPOND_MAILTOAD
DR04
A,F6," ",0,1,0,0,0," ",0,0,600,0,0,0,0,0,0,0,0,0,0," "
...
\ENDFAP\
...
\ENDIMAGE\
\NA=tpinfo-lp,LN=1,DUP=OFF,SIZE=0x0,TRAY=U,X=0,Y=0,PA=1,OPT=DSZ\
FCORRESPOND_MAILTOADDR01;34;350;16010;HN;;\US BANK, NA
FCORRESPOND_MAILTOADDR02;34;750;16010;HN;;\P. O. BOX 3427
FCORRESPOND_MAILTOADDR03;34;1161;16010;HN;;\OSH KOSH WI 54903
FTPID;34;1953;16010;HN;;\200053192
\ENDIMAGE\
\NA=tpinfo-lp,LN=1,DUP=OFF,SIZE=0x0,TRAY=U,X=0,Y=0,PA=1,OPT=DSZ\
FCORRESPOND_MAILTOADDR01;34;350;16010;HN;;\FORD MOTOR CREDIT
FCORRESPOND_MAILTOADDR02;34;750;16010;HN;;\P. O. BOX 23834
FCORRESPOND_MAILTOADDR03;34;1161;16010;HN;;\TUSCON AZ 85734
FTPID;34;1953;16010;HN;;\200053193
\ENDIMAGE\
\NA=tpinfo-lp,LN=1,DUP=OFF,SIZE=0x0,TRAY=U,X=0,Y=0,PA=1,OPT=DSZ\
FCORRESPOND_MAILTOADDR01;34;350;16010;HN;;\MOUNTAIN NAT'L. BANK
FCORRESPOND_MAILTOADDR02;34;750;16010;HN;;\320 COLLEGE DRIVE
FCORRESPOND_MAILTOADDR03;34;1161;16010;HN;;\MARTINSVILLE VA 24115
FTPID;34;1953;16010;HN;;\200053194
\ENDFORM\
```

This rule will replicate the form set three times (once for each occurrence of tpinfo-lp). The field data from the first occurrence of tipinfo-lp will be mapped to the correspondingly named fields in first occurrence of pvacov1tp. The field data for second occurrence of tpinfo-lp will be mapped to the first occurrence of pvacov1tp in the second copy of the form set. The process will be repeated for each occurrence of the source image.

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

Attachment variables

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
```

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.

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

Returns

Success or failure

Errors This rule can return these messages:

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.
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.

See also DPRGetUserList on page 147

DPRModifyWipData

Use this rule to modify a WIP record and create new NAFILE.DAT and POLFILE.DAT files. The rule uses RECORDID (or RECNUM, or UNIQUE_ID) or FIELD attachment variables to identify the record. All fields can be updated as defined in WIPDFD except RECORDID (or RECNUM, or UNIQUE_ID) and FORMSETID. The new NAFILE.DAT and POLFILE.DAT files override the existing ones.

Syntax

```
long _DSIAPI DPRModifyWipData ( 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

Attachment variables

This rule expects these attachment variables:

Variable	Description
RecordID	Enter the record ID. You can define it as the RECNUM or UNIQUE_ID in your DFD definition. UNIQUE_ID is typically used in SQL databases.
(*field names*)	Enter the appropriate value to match a record. Key1, Key2, KeyID, and RecType are required. See the definition of DOC_TAG.in the WIP.DFD file.
NEWWIP. FieldName	Used to update the record. Note that RECORDID (or RECNUM, or UNIQUE_ID) and FORMSETID will be ignored.

INI options

You can use these INI options:

```
< WIPData >
    File =
    Path =
```

Option	Description
File	Enter the name of the WIP file.
Path	Enter the path to the WIP file.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

See also

DPRAddWipRecord on page 71

DPRApproveWipRecords on page 73

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDelMultiWipRecords on page 109

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRGetOneWipRecord on page 143

DPRIni2XML on page 159

DPRLockWip on page 191

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRPatchLevel

Use this rule to get a Summary Patch Report for IDS (Docupresentment) and for Documaker.

Syntax

```
long _DSIAPI DPRPatchLevel ( 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

The Summary Patch Report for Documaker is conditional and uses attachment variables to determine if it should be run.

This report contains information about the names of attachment variables, sample output, and so on. You can then display the patch information via HTML.

NOTE: The rule provides a summary patch report. For more detailed information, use the FSIVER utility. See the Docutoolbox Reference for more information on this utility.

DPRParseRecord

Use this rule to assemble the attachment into a record and then convert it to a XML tree. The assembled record must be treated as a DFD internal record. The DFD defined in the Path option of the EmailDFD control group is used to map into the internal record.

NOTE: This rule is only available on Windows 32-bit platforms.

Syntax

```
long _DSIAPI DPRParseRecord ( 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

Use the DPRCreateEMailAttachment rule after this rule to merge the XML tree with a template and place the result into an attachment file. A global variable named XMLDOCVAR contains the handle to the XML tree. This variable is used by the DPRCreateEMailAttachment rule.

See also DPRCreateEMailAttachment on page 95

DPRPostDMProcess

Use this rule when the Documanage post processing rules cannot be used. For example, you can use this rule as a replacement for the post Documanage bridge processing in dual IDS configurations, where one IDS is running on Linux and another on Windows NT. This lets you retrieve data from the Linux client and use the Linux IDS for presentment (production of PDF files).

Syntax

```
long _DSIAPI DPRPostDMProcess ( 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

The PRINTFILE attachment variable is removed from both input and output attachment the PRINTPATH is added to the input attachment on the RUNF message. The PRINTPATH value is added for later use by the DPRPrint rule. The rest of the logic is executed on the RUNR message and does the following:

- The value of REMOTEPRINTFILE is in the output attachment and consists of PRINTPATH and the PRINTFILE values. Here is an example:

  ```
  \\servername\share\directory\tempfilename.pdf
  ```

- The system adds this value to the output attachment, GEN_TEMPFILE.
- The system uses the file name to build the URL with the following INI option This result is added to the output attachment as GEN_DESTINATION. The Documanage bridge client uses GEN_DESTINATION to redirect the browser to a new URL, for example, to display a PDF file. Here are the INI options from the CONFIG.INI file, used by this rule:

  ```
  < Attachments >
      URL =
      PrintPath =
  ```

- The URL should have the terminating slash, such as:

  ```
  https://www.domain.com/doc-html/
  ```

  If the slash is missing it will be appended.
- The file name portion of the REMOTEPRINTFILE is appended so in the example shown here the value of GEN_DESTINATION will be:

  ```
  https://www.domain.com/doc-html/tempfilename.pdf
  ```

Attachment variables

These variables are used as input:

Variable	Description
REMOTEPRINTFILE	This value is created by the DPRPrint rule. This attachment variable is used on RUNR message.

These variables are created by this rule:

Variable	Description
PRINTPATH	Taken from the appropriate PrintPath option in the Attachments control group of the CONFIG.INI file. This is added to the input attachment on the RUNF message.
GEN_TEMPFILE	Used by the client of Documanage bridge. This value is copied from REMOTEPRINTFILE attachment variable and is added to output attachment on RUNR message.
GEN_DESTINATION	Used by the client of Documanage bridge. This value is built from PRINTFILE attachment variable. Added to the output attachment on RUNR message.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot find variable #VARIABLE# in the attachment.

See also

DPRPrint on page 211

DPRPrint

Use this rule on the DSI_MSGRUNF message to return a print output. If you have recipient filtering turned on, this rule uses the Recip_Names control group to translate short recipient names into longer names, if this group exists in the DAP.INI file.

The DPRPrintFormset rule was replaced by two rules: DPRRetrieveFormset and DPRPrint. If the DPRPrintFormset is specified in the INI file, it execute these rules in a row, just as if they were specified in the INI file.

This change lets the custom rule have access to the FAP form set handle prior to print, so additional objects can be added on the fly. Place the DPRPrint rule in the list after the DPRRetrieveFormset rule. DPRRetrieveFormset rule creates DSI variable DPRFORMSET, which contains FAP form set handle.

If recipient filtering is on, this rule uses the Recip_Names control group in the DAP.INI file to translate short recipient names into longer names—if this control group exists in the INI file.

If you set the PRTTYPE to HTM, the form set in memory is converted into an XML tree and the DSI variable named DPRXMLFORMSET is created. This variable is used by DPRProcessTemplates rule.

Syntax

```
long _DSIAPI DPRPrint ( DSIHANDLE hdsi,
                         char * pszParms,
                         ULONG  ulMsg,
                         ULONG  ulOptions )
```

NOTE: The DPRPrint rule also works with the Documanage Bridge. If you include the MTCLoadFormset rule in the rule list, the DPRPrint rule will work with the form set loaded from that rule as well.

Parameters

Parameter	Description
DSIHANDLE hdsi	Pointer to the rules data
char * pszParms	Pointer to rule parameter string
ULONG ulMsg	DSI_ message
ULONG ulOptions	Options

Attachment variables

This rule expects:

Variable	Description
DPRFORMSET	This DSI variable contains the form sets to print and is created by some other rule, such as the DPRLoadImportFile rule.
PRINTFILE	This attachment variable contains the name of the output file. If the name of the output file is missing, this rule will generate a unique name and add it to the attachment with the name PRINTFILE.

Variable	Description
ALLRECIPIENTS	If this attachment variable is present, all recipients copies are printed.
RECIPIENT	This attachment variable contains the names of the recipients to print. If these names are missing, the system will print without recipient filtering. You can select multiple but not all recipients by including a comma-delimited list of the recipients you want to print in the RECIPIENT attachment variable. The rule reads the recipients listed in the attachment variable and prints copies for those recipients. You can set up multiple RECIPIENT attachment variables, but no RECIPIENT attachment variable can exceed 2047 bytes. If the ALLRECIPIENTS variable is present, the system ignores this value.
XMLALLOBJECTS	See XMLALLFIELDS. If the print type is HTML or XML, include this attachment variable to have the system dump the objects to HTML or XML. The system includes empty fields and object attributes. If the print type is XML, the page is loaded into an attachment variable called SENDBACKPAGE. If the print type is HTML, the page is stored in memory.
XMLALLFIELDS	Include this attachment variable to include empty fields as well as fields with data in an extended XML file. Use this attachment variable instead of the XMLALLOBJECTS attachment variable. The latter results in overly large XML files.
DPRPROOFLOGO	Include this attachment variable with a value of Yes to, for instance, create a normal PDF file and create another PDF file for proofing (with a *PROOF* logo). See Adding Logos when using DPRPrint on page 213 for more information.
PRTTYPE	(Optional) This attachment variable indicates the name of the printer in the PrtType control group. The default is PDF. Set to *CMBNA* to create a combined NA/POL export file. Set to *V2* to create a V2 export file.

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0010	Internal API failure, usually indicates the system is not configured correctly
DPR0022	The attachment variable cannot be located

Adding Logos when using DPRPrint

When using the DPRPrint rule, you can include DPRAddLogo functionality without having the DPRAddLogo rule in the request type. This lets you use the same request type, for example, to create a normal PDF file and create another PDF file for proofing (with a *PROOF* logo).

To use this functionality, you must pass the DPRPROOFLOGO attachment variable with value of Yes and you have to have the same setup as the DPRAddLogo rule in the CONFIG.INI file.

No error message is produced if the CONFIG.INI file does not include the AddLogo control group with these options:

```
< AddLogo >
    Logo    =
    Top     =
    Left    =
    Pages   =
    Color   =
```

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.

Adding Transaction Index Information to the XML Export File

The DPRPrint rule can output XML with field information needed by iPPS and iDocumaker. These fields are mapped from a WIP record using the WIPData control group:

```
< WIPData >
    Key1        = Company
    Key2        = Key2
    KeyID       = KeyID
    TranCode    = TranCode
    StatusCode  = StatusCode
    Desc        = Desc
```

The field values in the WIPData control group should be the field names that correspond to those in the WIP DFD file. If the fields are not defined in the WIPData control group for the master resource library (MRL) configuration file, the default names are used. In addition, the CONFIG value will also be added as a LIBRARY element.

Here is an example of the field information:

```
<?xml version="1.0" encoding="UTF-8"?>
 <DOCUMENT TYPE="RPWIP" VERSION="11.1">
 <DOCSET NAME="">
  <LIBRARY CONFIG="amergen_import">amergen_import</LIBRARY>
  <KEY1 NAME="Company">GENERAL LIABILITY</KEY1>
  <KEY2 NAME="KEY2">POLICY</KEY2>
  <KEYID NAME="KEYID">TEST</KEYID>
  <TRANCODE NAME="TRANCODE">RN</TRANCODE>
  <STATUSCODE NAME="STATUSCODE">W</STATUSCODE>
  <DESC NAME="DESC" />
  </DOCSET>
  </DOCUMENT>
```

Generating File Names Based on Transaction Values

You can use the DPRPrint and DPRUnloadExportFile rules to 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 gives you control over output file names and can be used, for example, when you need to interface to a 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.

INI options

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	

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 path. If omitted, the system checks the INI options in the following control groups to determine the file type. • For V2, it checks the ExpFile_CD control group • For CMBNA, it checks the ImpExpCombined control group • For XML, it checks the XML_IMP_EXP control group. If these INI options are omitted, the system creates a unique file name.

INI options

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.

See also

DPRAddLogo on page 65

DPRPrintFormset on page 221

DPRRetrieveFormset on page 231

DPRProcessTemplates on page 224

DPRUnloadExportFile on page 278

DPRPrintDpw

Use this rule to print a DPW file that can be added as a new WIP record or to generate a DPW file from an existing WIP record. The rule creates a temporary INI context and adds the necessary INI options for DPWLIB to generate a DPW file. The code looks up values for the DPW index as follows:

- It first looks up values for the DPW index from rule arguments (see the Rule Arguments section below).
- The code then looks up values in the Ini2Xml group for backwards compatibility (see feature 1208 for version 1.8).
- Finally, it traverses the WIP index fields and looks up the values from input attachment variables matching the field names. In the case were values are found in more than one location, rule arguments take first precedence, then values from the Ini2Xml group, and lastly, values from input attachment variables.

Syntax

```
long _DSIAPI DPRPrintDpw ( DSIHANDLE hdsi,
                           char * pszParms,
                           ULONG  ulMsg,
                           ULONG  ulOptions )
```

Parameters

Parameter	Description
DEBUG	(Optional) If this rule argument is present, the rule will set the debug flag for DPWLIB.
ATTACHVARNAME or ATTACHVARNAME = VALUE	(Optional) Where ATTACHVARNAME is the name of an input attachment variable you want to use to update the DPW index. Multiple ATTACHVARNAME names can be provided. If a value is provided, the rule uses it instead of looking up a value in the input attachment.

Attachment variables

None

Attachment outputs

Attachment	Description
RESULTS	Returns Success or failure.

Example

Here is an example request type:

```
<section name="ReqType:DPR_IWIPEDIT">
    <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
    <entry name="function">atcw32->
;ATCSendFile,RF_POSTFILE,PRINTFILE,Binary</entry>
    <entry name="function">dprw32->;DPRSetConfig</entry>
    <entry name="function">dprw32->;DPRInitLby</entry>
    <entry name="function">dprw32->;DPRDecryptLogin</entry>
    <entry name="function">dprw32->;DPRDefaultLogin</entry>
    <entry name="function">dprw32->;DPRCheckLogin</entry>
    <entry name="function">dprw32->;DPRGetWipFormset</entry>
    <entry name="function">dprw32-
>;DPRPrintDpw,ENCRYPTEDLOGIN,DEBUG,KEYID</entry>
```

```
    </section>
```

NOTE: The DPRPrintDpw rule uses DPWLIB to generate the DPW file. For more information on generating DPW files, see the Internet Document Server Guide.

DPRPrintFormset

Use this rule to return printed output. This rule retrieves data from a Documaker archive, loads the NA and POL files, and creates a print spool file in PDF format. This rule also registers the PDF file with the server cache for removal in two hours.

Syntax

```
long _DSIAPI DPRPrintFormset ( 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

Attachment variables

This rule expects these attachment variables:

Variable	Description
USERID	ID of the requester.
ARCKEY	Documaker archive key value used to retrieve the data.
PRTTYPE	(Optional) Defaults to PDF, the name of printer in the PrtType control group.
PRINTPATH	(Optional) Location of the output print file, if this value is missing, the system uses PrintPath option in the Attachments control group.
RECIPIENT	The name of the Documaker recipient from POL file.

This rule creates these attachment variables:

Variable	Description
REMOTEPRINTFILE	The name of the output file, it consists of the PRINTPATH and the generated output file name. For instance, if PRINTPATH was html\, the REMOTEPRINTFILE will be something like html\00001AB0.pdf
RESULTS	SUCCESS or DPRXXXX error code

If the execution was successful, this rule copies input attachment into the output attachment.

INI options

This rule uses these INI options:

Control Group	Option	Description
Attachments	Debug	If set to Yes, the temporary NA and POL files are not removed. This is useful for debugging purposes. The default is No.
	PrintPath	Location for the output PDF file, this option is ignored if attachment variable PRINTPATH exists.
MasterResource	DefLib	Location of the Documaker resources DefLib. Defaults to current directory.
	XRFFile	Name of the FXR file, no default. If you omit this option, an error occurs.
Control	XRFExt	Extension of the FXR file. Defaults to *FXR*
	FormLib	Location of Documaker resources. Defaults to the current directory.
	ImageExt	Extension of Documaker image files. Defaults to *FAP*
	LogoExt	Extension of Documaker logo files. Defaults to *LOG*
PDFFileCache	TimeOut	Specifies the number of seconds to keep the PDF file before deleting it. The default is 7200 seconds or 2 hours. You can add this control group and option to the DAP.INI file or in the each of the configuration INI files.
Recip_Names		(Optional) Use this INI control group to translate short recipient names from POL file into long names.
PrtType:PDF		See the chapter on using the PDF Converter in the Internet Document Server Guide for more information.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot find variable VARIABLE in the attachment list.
DPR0010	The system encountered an internal error of unknown type. The call by LOCATION to APINAME failed. Usually indicates improper configuration or DLL version mismatch.
DPR0011	The attachment field VARIABLE does not contain valid data. Means a required variable, such as ARCKEY, is blank.
DPR0014	Documaker platform error.
DPR0018	Cannot load font cross reference file.

Message	Description
DPR0019	Cannot retrieve data into the file. ARCRetrieveDoc API failed. Means the archive is corrupt or Documaker internal error.
DPR0020	DSLoadFormList API failed on file FILE. The POLFILE.DAT file cannot be loaded.
DPR0021	DSLoadNAFormset API failed on file FILE. The NAFILE.DAT file cannot be loaded.
DPR0022	Cannot add variable VARIABLE to the attachment list.

DPRProcessTemplates

Use this rule to take information from an XML tree and place it onto an HTML template. Use this rule with the DPRPrint rule and place it in the rule list after the DPRPrint rule.

When you use the DPRProcessTemplates rule, the system runs template processing against the XML tree in memory located in the DPRXMLFORMSET DSI variable. You create this tree using the DPRPrintFormset rule.

You can specify the name of this variable as a parameter to the rule. If the system cannot find the variable, no error is generated and the rule simply returns.

The names of the templates are determined by INI control groups. The main page is specified in the Template option of the EBPP control group. The templates for the other pages are specified in the EBPPTemplates control group. Here is an example:

```
< EBPP >
    Template = mstrres\ebpp\tmpl\bill.htm
    DebugXML = Yes
< EBPPTemplates >
    History = mstrres\ebpp\tmpl\history.htm
    Details = mstrres\ebpp\tmpl\details.htm
< Attachments >
    PrintPath = mstrres\ebpp\html
```

Option	Description
Template	Use this option to specify the template to use for the main page.
DebugXML	Set this option to Yes if you want the system to unload the XML tree into a file named EBPPTEST.XML. You can review this file for debugging purposes. This option defaults to No.
History Details	These options serve as examples of how you specify the templates to use for the remaining pages.
PrintPath	Use this option to tell the system where to place the output files it creates.

The following settings add the following XML elements to the XML tree as children of the <DOCSET> element and will produce three output files. The extension of the file output names are the same as the extensions of the input files, as specified in the INI file.

```
<TEMPLATES>
<MAINPAGE>
    7C311063A8F2F811D3F0B6C600A028CC56DF6578.htm
</MAINPAGE>
<Details>
    B6313FD4CDF2F711D322B6C600A048CC56DF659A.htm
</Details>
<History>
    B6313FD6CFF2F711D326B6C600A050CC56DF659B.htm
</History>
</TEMPLATES>
```

Syntax

```
long _DSIAPI DPRProccessTemplate ( DSIHANDLE hdsi,
                                   char * pszParms,
                                   unsigned long  ulMsg,
                                   unsigned long  ulOptions )
```

Parameters

Parameter	Description
DSIHANDLE hdsi	Pointer to the rules data
char * pszParms	Pointer to rule parameter string
unsigned long ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
unsigned long ulOptions	Options

Attachment variables

This rule expects these attachment variables:

Variable	Description
PrintFile	If provided, the output of a main template is written to this file. Otherwise, the output the system generates a unique file name and writes the output to that file.
PrintPath	The location for the output files. If you omit this value, the system uses the PrintPath option in the Attachments control group for this information.
PRTType	The default extension for the output file names, if the specification of the templates in the INI file did not include an extension.

This rule creates these attachment variables:

Variable	Description
RemotePrintFile	The name of the output file. This name consists of the print path and the generated output file name. For instance if the print path was *html\*, the result will be something like *html\\00001AB0.pdf*.
Results	Success or a DPRXXXX error code

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0022	Cannot add variable VARIABLE to the attachment list.
DPR0037	Cannot find the template file #FILE,#.

See also

DPRPrint on page 211

DPRPrintFormset on page 221

DPRRenameVars

Use this rule to rename attachment variables. The rule parameters specify a *name1=name2* pair. On the MSG_RUNF the *name1* attachment value in the input attachment is renamed to *name2*, on MSG_RUNR the *name2* attachment variable in the output attachment is renamed to *name1*. Multiple pairs of comma-delimited *name1=name2* pairs can be specified for the same rule.

Syntax

```
long _DSIAPI DPRRenameVars ( 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

You can use this rule to glue together two rules, one of which creates the attachment variable with one name, but another expects this value in the different attachment variable.

This rule should be the very first rule in the rule list for a particular request type after the ATCLoadAttachment and ATCUnloadAttachment rules. If the variable is missing in the attachment, error is generated and processing continues.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot find the variable VARIABLE in the attachment list.

DPRRetFromUserDict

Use this rule to retrieve words from a user dictionary.

Syntax

```
long _DSIAPI DPRRetFromUserDict ( DSIHANDLE hdsi,
                                  char * pszParms,
                                  unsigned ulMsg,
                                  unsigned ulOptions )
```

Parameters

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

INI options

You can use these INI options with this rule:

```
< Spell >
    LanguageOpt    =
    UserDict       =
    UserDictPath   =
```

Option	Description	
LanguageOpt	Enter the language option. The default is US English. You can choose from these languages and dictionaries:	
	Danish	“ssceda.tlx,ssceda2.clx”
	Dutch	“sscedu.tlx,sscedu2.clx”
	Finnish	“sscefi.tlx,sscefi2.clx”
	French	“sscefr.tlx,sscefr2.clx”
	German	“sscege.tlx,sscege2.clx”
	Italian	“ssceit.tlx,ssceit1.clx”
	Norwegian	“sscenb.tlx,sscenb2.clx”
	Portuguese_Brazil	“sscepb.tlx,sscepb2.clx”
	Portuguese	“sscepo.tlx,sscepo2.clx”
	Spanish	“sscesp.tlx,sscesp2.clx”
	Swedish	“sscesw.tlx,sscesw2.clx”
	UK English	“sscebr.tlx,sscebr2.clx”
	US English	“ssceam.tlx,ssceam2.clx”
UserDict	Enter the name of the user dictionary. The default is user.tlx.	
UserDictPath	Enter the path to the user dictionary. The default is the current working directory.	

Attachment variables

Variable	Description
RETFILE	The output XML file name with full path for the new XML document tree. This file will include all words retrieved from the user dictionary. If you omit the name, the system generates a unique name for you. If you omit the path, the system checks the UserDictPath INI option to get the default path. If no path is specified there, the system exports the file to the current working directory.
LanguageOpt	The language selection. The default is US English. You can choose from these languages and dictionaries: Danish “ssceda.tlx,ssceda2.clx” Dutch “sscedu.tlx,sscedu2.clx” Finnish “sscefi.tlx,sscefi2.clx” French “sscefr.tlx,sscefr2.clx” German “sscege.tlx,sscege2.clx” Italian “ssceit.tlx,ssceit1.clx” Norwegian “sscenb.tlx,sscenb2.clx” Portuguese_Brazil “sscepb.tlx,sscepb2.clx” Portuguese “sscepo.tlx,sscepo2.clx” Spanish “sscesp.tlx,sscesp2.clx” Swedish “sscesw.tlx,sscesw2.clx” UK English “sscebr.tlx,sscebr2.clx” US English “ssceam.tlx,ssceam2.clx”
UserDict	The name of the user dictionary. The default is user.tlx.

Attachment outputs

Variable	Description
RETFILE	The name of the retrieved XML tree file with a full path. See the discussion of this variable in the input attachment table.

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
RPD0002	Cannot create #TAGNAME,# at #LOCATION,#.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.
DPR0051	Failed to unload XML file #FILE,# in #LOCATION,#.

Here is an example of the retrieved file layout:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SPELLER TYPE="IENTRY" VERSION="3.1">
<FIELDH>speling</FIELDH>
<FIELDH>spellin</FIELDH>
<FIELDH>spellng</FIELDH>
</SPELLER>
```

DPRRetrieveDPA

Use this rule to read a DPA file and create in memory a form set.

Before you run the DPRRetrieveDPA rule, the DPA file must be placed on disk by some other rule or set of rules. For instance, if you are using Documanage, you could use Documanage Bridge rules to put the DPA file on disk.

Once this rule creates the form set from the DPA file, you can use other Documaker Bridge rules, such as DPRPrint, to further process the form set.

Syntax

```
Function = dprw32->DPRRetrieveDPA
```

Attachment variables

Variable	Description
DMSARCFILE	This tells the DPRRetrieveDPA rule the path to the cached DPA file. If DMSARCFILE does not appear on the input attachment list, the DPRRetrieveDPA rule does nothing.

Attachment outputs

Variable	Description
OLDCONFIG	The DPRRetrieveDPA rule sets CONFIG to the value in the DPA file during the run forward step. During the run reverse step, the DPRRetrieveDPA rule restores CONFIG to its original value unless that value differs from what it was set to for the DPA conversion.

Be sure to set up the proper INI file options and resources before using this rule.

The DPRRetrieveDPA rule automatically calls the DPRRetrieveFormset and DPRSetConfig rules. There is no need to place them on the rules list.

See also

DPRPrint on page 211

DPRRetrieveFormset

Use this rule to retrieve a form set from a Documaker archive. This rule retrieves data from Documaker archive, loads the NA and POL files, and creates the DSI variable DPRFORMSET.

Syntax

```
long _DSIAPI DPRRetrieveFormset ( DSIHANDLE hdsi,
                                  char * pszParms,
                                  ULONG  ulMsg,
                                  ULONG ulOptions )
```

Parameters

Parameter	Description
DSIHANDLE hdsi	Pointer to the rules data
char * pszParms	Pointer to rule parameter string
ULONG ulMsg	DSI_ message
ULONG ulOptions	Options

The DPRPrintFormset rule was replaced by two rules: DPRRetrieveFormset and DPRPrint. If the DPRPrintFormset is specified in the INI file, it execute these rules in a row, just as if they were specified in the INI file.

This change lets the custom rule have access to the FAP form set handle prior to printing, so additional objects can be added. Place the DPRPrint rule after the DPRRetrieveFormset rule. DPRRetrieveFormset rule creates DSI variable DPRFORMSET, which contains FAP form set handle.

INI options

This rule uses these INI options:

```
< Attachments >
   Debug = No
< MasterResource >
   DefLib = /DefLib
   FormLib = /FormLib
< Control >
   ImageExt =
   LogoExt =
< Attachments >
   PrintPath =
< Recip_Names >
   xxx = xxx
< PrtType:PDF >
   xxx = xxx
```

The Recip_Names control group is used to translate short recipient names from POL file into long names, this group is optional. The entire PrtType:PDF control group is used. See the Using the PDF Converter in the Internet Document Server Guide for more information.

Option	Description
Debug	If set to Yes, the temporary NAFILE.DAT and POLFILE.DAT files are not removed, useful for debugging. Defaults to No.
DefLib	The location of the Documaker resources DefLib. Defaults to the current directory.
FormLib	The location of Documaker resources. Defaults to the current directory.
ImageExt	The extension of Documaker image files. Defaults to *FAP*.
LogoExt	The extension of Documaker logo files. Defaults to *LOG*.
PrintPath	The location for the output PDF file, this option is ignored if the PRINTPATH attachment variable exists.

Attachment variables

This rule expects these attachment variables:

Variable	Description
UserID	The ID of the requester.
ARCKEY	The Documaker archive key value used to retrieve the data.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot find variable VARIABLE in the attachment list.
DPR0010	Internal API failure, usually indicates something is configured incorrectly or there is a DLL version mismatch.
DPR0011	The attachment field VARIABLE does not contain valid data. Means the required variable, such as ARCKEY, is blank.
DPR0014	Documaker platform error.
DPR0018	Cannot load the font cross-reference file (FXR).
DPR0019	Cannot retrieve data into the file. ARCRetrieveDoc API failed. Means the archive is corrupt or there was a Documaker internal error.
DPR0020	DSLoadFormList API failed on the file named FILE. The POLFILE.DAT file cannot be loaded.
DPR0021	DSLoadNAFormset API failed on the file named FILE. The NAFILE.DAT file cannot be loaded.
DPR0022	Cannot add variable VARIABLE to the attachment list.

See also

DPRPrint on page 211

DPRPrintFormset on page 221

DPRRotateFormsetPages

Use this rule with the Printstream Bridge to Documerge to rotate text from Metacode pages. This rule rotates the pages if most of the text and other objects are rotated so the page will look correct when viewed with the PDF viewer. This rule does not expect any attachment variables.

NOTE: When you use the DPRDelBlankPages or DPRRotateFormsetPages rules with form sets created from Metacode or AFP print streams, the rules work fine. If, however, you use these rules with form sets created from Documaker archives or from import files, the rule appear to work incorrectly because not all of the static form data is loaded when these rules execute. The result is that text may not be rotated or pages with content may be deleted.

Use the DPRLoadFAPImages rule to correct this problem. Insert this rule after the rule that creates the form set, such as DPRRetrieveFormset or DPRLoadImportFile, and before the rule that prints the form set, such as DPRPrintFormset or DPRPrint.

Syntax

```
long _DSIAPI DPRRotateFormsetPages ( 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

The pszParms parameter is the name of the variable in the form set. The default value, if no rule parameter is specified in the INI file, is MTCFORMSET. It is registered on the MTC request in between the MTCLoadFormset and MTCPrintFormset rules.

This DSI variable should contain a valid Documaker form set handle. This rule runs on DSI_RUNF message.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0017	DSI variable VARIABLE cannot be located

DPRSearch

Use this rule to return a list of matching archive records.

Syntax

```
long _DSIAPI DPRSearch ( 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

Attachment variables

This rule expects these attachment variables:

Variable	Description
USERID	The user ID of the requestor
FIELDS	The comma-delimited list of columns in application index file, to be used in the query
MAXRECORDS	(Optional) Enter the maximum number of records to return, if this value is missing, the system uses the value entered for the MaxRecords option in the ArcRet control group, if none is specified, the system uses 20 as the default.
CASESENSITIVE	If this attachment variable is present, the search is case sensitive.
RESTART	The condition for setting the start position before the search
PARTIALMATCH	If this value is present, the search condition uses partial match, so the value A matches the column value ABC.
TABLEINIGROUP	(Optional) The name of the INI control group to get the application index table name from, the default is the ArcRet control group
TABLEINIOPTION	(Optional) The name of the INI option to get the application index table name from, the default is AppIdx.

If the TABLEINIGROUP and TABLEINIOPTION variables are missing, the system uses the value for the AppIdx option in the ArcRet control group as a default.

All of the columns specified in the FIELDS attachment variable should be in the attachment as well. For example, if...

```
FIELDS = Key1,Key2,KeyD
```

...then Key1, Key2, and KeyID are required attachment variables.

This rule creates these attachment variables:

Variable	Description
MORERECORDS	if there are more matches than was returned, this variable is set to Yes.
FIRSTRECORD	the number of the first record in the returned record set, 1 when this is the first search
LASTRECORD	the number of the last record in the returned record set
RECORDS	the attachment record, stem variable with every column from the application index table.

This rule creates an attachment variable RESULTS with the value SUCCESS.

On successful execution, this rule copies the input attachment into output.

Returns Success or failure

DPRSearchLDAP

Use this rule to search a Directory Information Tree (DIT) in an LDAP server to determine a user ID group or role membership. This rule looks for all configuration options in rule arguments, a properties file, INI options, and input attachment variables, in that order. Option values found in more than one source override the previous value.

Syntax

```
long _DSIAPI DPRSearchLDAP ( 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

Attachment variables

Option	Description
RUNMSG	(Optional) An integer value between 1 and 4 indicating in which message the rule should run: INIT(1), TERM(2), RUNF(3), RUNR(4). This option is only searched as a rule argument or input message variable. The default is 3.
LDAP.PROPERTIES	(Optional) The name of a properties file that should be used to look up the options for the rule. The default file name is *ldap.properties*, which is looked up in the current directory for IDS. This option is only searched as a rule argument or input message variable.
LDAP.HOST	(Optional) The host name or IP address of the LDAP server. The default is localhost.
LDAP.PORT	(Optional) The port in which the LDAP server is listening on. The default is 389 when SSL is not used, 636 otherwise (see LDAP.USE.SSL option).
LDAP.URL	(Optional) The URL the LDAP server is listening on. If a value is specified for this property, it overrides the values specified for LDAP.HOST and LDAP.PORT.
LDAP.UID	(Optional) The user ID for logging into the LDAP server. If this value is provided and LDAP.USER is not provided, the user ID is derived from this value and the value provided for LDAP.DOMAIN option, such as *administrator@pd.com*.

Option	Description
LDAP.USER	(Optional) An explicit value to use for the user ID for the purpose of login into the LDAP server. Define this option to override the behavior used to determine the user ID when LDAP.UID and LDAP.DOMAIN are defined - see LDAP.DOMAIN.
LDAP.AUTHENTICATION. MODE	(Optional) The method of authentication used to login into the LDAP server. You can choose from: (simple) - clear-text password authentication (none) - anonymous authentication The default is (simple).
LDAP.PWD	(Optional) The password used to login into the LDAP server.
LDAP.TIMEOUT	(Optional) The amount of time (in milliseconds) after which a connection attempt or query should expire. The default is 10000 (10 seconds).
LDAP.SEARCH.BASE	(Optional) The base of the search in the DIT. This is the starting point (node location) of a search in the DIT. If a value is not provided, the system looks for the LDAP.DOMAIN option and builds a search base from it.
LDAP.DOMAIN	(Optional) This is the domain of the LDAP server. It is used to build the user ID for login into the LDAP server by appending the at symbol (@) plus the value of this option to the LDAP.UID value. The value of LDAP.DOMAIN is further parsed into domain components which are used as the default value for LDAP.SEARCH.BASE, if not already defined.
LDAP.OBJECTS	(Optional) A semicolon-delimited filter list of object classes to search in the LDAP server. If defined, it overrides the default filter list of object classes to search: group and groupOfNames.

LDAP.OBJECTS.SEARCH. STRING	(Optional) An explicit string value to be used as the filter of object classes to search. If defined, it overrides any value provided for LDAP.OBJECTS option.
	The value provided for this option must be specified in the appropriate LDAP protocol filter format. Also, if the search filter contains a question mark (?), the system replaces it with the user ID passed in as an argument to this function. Here are some examples:
	`(|(objectClass=group)(objectClass=group OfNames)).`
	`Cn=?`

LDAP.OBJECT. ATTRIBUTES	(Optional) The name of the attributes to retrieve for each object class, which contain a value that will be used to determine a match for USERID specified. The default values are *member* and *cn* (*cn* is always included).

Option	Description
LDAP.USER	(Optional) An explicit value to use for the user ID for the purpose of login into the LDAP server. Define this option to override the behavior used to determine the user ID when LDAP.UID and LDAP.DOMAIN are defined - see LDAP.DOMAIN.
LDAP.AUTHENTICATION. MODE	(Optional) The method of authentication used to login into the LDAP server. You can choose from: (simple) - clear-text password authentication (none) - anonymous authentication The default is (simple).
LDAP.PWD	(Optional) The password used to login into the LDAP server.
LDAP.TIMEOUT	(Optional) The amount of time (in milliseconds) after which a connection attempt or query should expire. The default is 10000 (10 seconds).
LDAP.SEARCH.BASE	(Optional) The base of the search in the DIT. This is the starting point (node location) of a search in the DIT. If a value is not provided, the system looks for the LDAP.DOMAIN option and builds a search base from it.
LDAP.DOMAIN	(Optional) This is the domain of the LDAP server. It is used to build the user ID for login into the LDAP server by appending the at symbol (@) plus the value of this option to the LDAP.UID value. The value of LDAP.DOMAIN is further parsed into domain components which are used as the default value for LDAP.SEARCH.BASE, if not already defined.
LDAP.OBJECTS	(Optional) A semicolon-delimited filter list of object classes to search in the LDAP server. If defined, it overrides the default filter list of object classes to search: group and groupOfNames.

LDAP.OBJECTS.SEARCH. STRING	(Optional) An explicit string value to be used as the filter of object classes to search. If defined, it overrides any value provided for LDAP.OBJECTS option.
	The value provided for this option must be specified in the appropriate LDAP protocol filter format. Also, if the search filter contains a question mark (?), the system replaces it with the user ID passed in as an argument to this function. Here are some examples:
	`(|(objectClass=group)(objectClass=group OfNames)).`
	`Cn=?`

LDAP.OBJECT. ATTRIBUTES	(Optional) The name of the attributes to retrieve for each object class, which contain a value that will be used to determine a match for USERID specified. The default values are *member* and *cn* (*cn* is always included).

Option	Description
LDAP.USER	(Optional) An explicit value to use for the user ID for the purpose of login into the LDAP server. Define this option to override the behavior used to determine the user ID when LDAP.UID and LDAP.DOMAIN are defined - see LDAP.DOMAIN.
LDAP.AUTHENTICATION. MODE	(Optional) The method of authentication used to login into the LDAP server. You can choose from: (simple) - clear-text password authentication (none) - anonymous authentication The default is (simple).
LDAP.PWD	(Optional) The password used to login into the LDAP server.
LDAP.TIMEOUT	(Optional) The amount of time (in milliseconds) after which a connection attempt or query should expire. The default is 10000 (10 seconds).
LDAP.SEARCH.BASE	(Optional) The base of the search in the DIT. This is the starting point (node location) of a search in the DIT. If a value is not provided, the system looks for the LDAP.DOMAIN option and builds a search base from it.
LDAP.DOMAIN	(Optional) This is the domain of the LDAP server. It is used to build the user ID for login into the LDAP server by appending the at symbol (@) plus the value of this option to the LDAP.UID value. The value of LDAP.DOMAIN is further parsed into domain components which are used as the default value for LDAP.SEARCH.BASE, if not already defined.
LDAP.OBJECTS	(Optional) A semicolon-delimited filter list of object classes to search in the LDAP server. If defined, it overrides the default filter list of object classes to search: group and groupOfNames.

LDAP.OBJECTS.SEARCH. STRING	(Optional) An explicit string value to be used as the filter of object classes to search. If defined, it overrides any value provided for LDAP.OBJECTS option.
	The value provided for this option must be specified in the appropriate LDAP protocol filter format. Also, if the search filter contains a question mark (?), the system replaces it with the user ID passed in as an argument to this function. Here are some examples:
	`(|(objectClass=group)(objectClass=group OfNames)).`
	`Cn=?`

LDAP.OBJECT. ATTRIBUTES	(Optional) The name of the attributes to retrieve for each object class, which contain a value that will be used to determine a match for USERID specified. The default values are *member* and *cn* (*cn* is always included).

Option	Description
LDAP.MATCH. ATTRIBUTES	(Optional) The name of one or more attributes that are contained within the value returned by a search for the LDAP.OBJECT.ATTRIBUTES option. This is the name of an attribute whose value will be used to compare vs. the USERID specified to determine a match. For example, if LDAP.OBJECTS contains a value of 'groupOfUniqueNames' and LDAP.OBJECT.ATTRIBUTES contains a value of 'uniqueMember' and value returned for the 'uniqueMember' attribute of 'groupOfUniqueNames' object class is 'uid=admin,ou=people, dc=mycompany,dc=com' and you want to match the USERID value with the value for 'uid', you would supply a value of 'uid' for this option. The default is *cn*.
LDAP.SEARCH. SCOPE	(Optional) The scope of the search. You can choose from: (base) - search only the named context (one) - search one level below the named context but not the named context (sub) - search the entire subtree, including the named context. The default is (sub).
LDAP.DEREF.LINK	(Optional) Enter No if you do not want the system to reference links to other nodes during a search. The default is Yes.
LDAP.VERSION	(Optional) An integer value indicating the LDAP protocol version to use. You can choose from: (2) - Version 2 (3) - Version 3 The default is (3).
LDAP.SEARCH.LEVEL	(Optional) This property specifies the search level. You can choose from 1 (USER) 2 (GROUPS) The system executes different logic to search group type objects or user type objects based on the search level specified. The default is 1 (USER).

Option	Description

LDAP.DN.IDENTIFIER	(Optional) The value for this property is used in the following ways:
	
	- In cases were LDAP.SEARCH.LEVEL is equal to 1 (USER) and there is no LDAP.OBJECTS.SEARCH.STRING value specified, the system generates a default search filter of the format identifier=userID, where *identifier* is the value of this property and *userID* is the user ID passed in as an argument to this function.
	
	- In cases were LDAP.SEARCH.LEVEL is equal to 2 (GROUPS) and there is no LDAP.OBJECTS.SEARCH.STRING value specified, the system generates a default search filter from LDAP.OBJECTS and LDAP.OBJECT.ATTRIBUTES, where each attribute value in the search filter is an asterisk (*). This tells the system to match any value for the attributes specified. If LDAP.RDNDS property is also provided, the asterisk (*) is replaced with identifer=userID, followed by a comma and the LDAP.RDNS value to fine tune the search, where *identifier* is the value for this property and *userID* is the user ID passed in as an argument to this function. Here is an example of a default search filter:
	
	`(&((objectClass=groupOfNames)(member=*)))`
	
	In a case where a value of
	
	`'CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM'`
	
	is specified for LDAP.RDNS and this property contains a value of *CN*, the search filter generated would look like this:
	
	`(&((objectClass=groupOfNames)(member=CN =Administrator, CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM))).`
	
	The default is *CN*.

Option	Description

LDAP.RDNS	(Optional) This property is only used when LDAP.SEARCH.LEVEL is equal to 2 (GROUPS) and when the LDAP.OBJECTS.SEARCH.STRING option is not specified. In such a case, the system builds a default search filter from LDAP.OBJECTS and LDAP.OBJECT.ATTRIBUTES and attribute values specified in the default search filter will contain an asterisk (*). This tells the system to match any value for the attributes specified.
	When this property is specified, the system uses the value along with the value for LDAP.DN.IDENTIFIER to replace the asterisk (*) and narrow the search, thereby speeding the process. Here is an example of a default search filter:
	`(&((objectClass=groupOfNames)(member=*)))`
	In a case where a value of
	`'CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM'`
	is specified for this property and LDAP.DN.IDENTIFIER contains a value of *CN*, the search filter generated would look like this:
	`(&((objectClass=groupOfNames)(member=CN=Administrator, CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM))).`

LDAP.USE.SSL	(Optional) A value of Yes enables encrypted communication through an SSL channel. For SSL connections to work, the LDAP server must be configured for SSL with a certificate from a trusted certification authority. This configuration is vendor specific, consult your vendor documentation for more information.
LDAP.DEBUG	(Optional) A value of Yes enables logging of debugging information to a file named *trace*.

Attachment outputs

Variable	Description
RESULTS	Success or failure. No matches means failure.</td></tr>
LDAPERROR	A standard LDAP error is returned as a rowset in case of failure. In such a case, the LDAPERROR will also be added as an entry in the ERRORS rowset.
LDAP.ENTRIES	Matches for the search criteria specified will be returned in the form of an LDAP.ENTRIES rowset, with each element named as an ENTRY in the rowset.

Example

Here is an example of a properties file:

```
ldap.uid=Administrator
ldap.pwd=~Encrypted 2XAUnkxUYlx7i5AnQ4m4E1m00
ldap.host=PDDC.pd.com
ldap.port=389
ldap.authentication.mode=simple
ldap.domain=PDDC.pd.com
ldap.objects.search.string=(&(objectClass=group)(cn=Administrators)
)
ldap.object.attributes=member
ldap.match.attributes=cn
ldap.debug=yes
```

Here is another example of a properties file:

```
ldap.user=uid=admin,ou=people,dc=mycompany,dc=com
ldap.pwd=~Encrypted 2XAUnkxUYlx7i5AnQ4m4E1m00
ldap.host=localhost
ldap.port=636
ldap.authentication.mode=simple
ldap.search.base=ou=roles,dc=mycompany,dc=com
ldap.objects=group;groupOfNames;groupOfUniqueNames
ldap.object.attributes=uniqueMember;member
ldap.match.attributes=uid;cn
ldap.debug=yes
ldap.version=3
ldap.search.scope=sub
ldap.deref.link=true
ldap.use.ssl=Y
ldap.ssl.protocol=SSLv3
ldap.ssl.socketFactory.class=com.docucorp.util.LDAPSSLSocketFactory
ldap.ssl.key.store=c:/docserv/keystore/javakeystore
ldap.ssl.key.store.pwd=~Encrypted 2yQgqaRIZkRJd6m8L7WWD1000
ldap.ssl.key.store.type=JKS
ldap.ssl.key.store.manager.type=SunX509
ldap.ssl.trust.store=c:/docserv/keystore/javakeystore
ldap.ssl.trust.store.pwd=~Encrypted 2yQgqaRIZkRJd6m8L7WWD1000
ldap.ssl.trust.store.type=JKS
ldap.ssl.trust.store.manager.type=SunX509
```

Here is another example of a properties file:

```
ldap.host=localhost
ldap.port=389
ldap.authentication.mode=none
ldap.search.base=ou=roles,dc=mycompany,dc=com
ldap.objects=group;groupOfNames;groupOfUniqueNames
ldap.object.attributes=uniqueMember;member
ldap.match.attributes=uid;cn
ldap.debug=yes
ldap.version=3
ldap.search.scope=sub
ldap.deref.link=true
```

Here is an example request type:

```
<section name="ReqType:TEST_LDAP_Search_2">
    <entry name="function">atcw32-&gt;ATCLoadAttachment</entry>
    <entry name="function">atcw32-&gt;ATCUnloadAttachment</entry>
    <entry name="function">dprw32-&gt;DPRSetConfig</entry>
    <entry name="function">dprw32-&gt;DPRSearchLDAP,
        RUNMSG=4</entry>
</section>
```

Keep in mind...

- Encrypted option values should be preceded by this keyword:

  ```
  ~Encrypted
  ```

  followed by a space (see the ldap.pwd value in the examples of a properties file).

- The options in an INI file for a configuration available to IDS should be placed in a control group named LDAP. You must also provide a CONFIG input message variable or rule argument so IDS can find the LDAP control group in the appropriate INI file. Here is an example:

  The DAP.INI file configuration:

  ```
  < Config:Example >
      INIFile = example.ini
  ```

  The EXAMPLE.INI file configuration:

  ```
  < LDAP >
      ldap.host = localhost
      ldap.port = 389
      ldap.timeout = 10000
      ldap.uid = userID@PDDC.pd.com
      ldap.pwd = 123456xxx
      ldap.objects.search.string = cn=?
      ldap.authentication.mode = simple
      ldap.domain = PDDC.pd.com
      ldap.dn.identifier = cn
  ```

  The input message variable that is part of the request:

  ```
  CONFIG=Example
  ```

  The request type:

  ```
  <section name="ReqType:SearchLDAP">
  <entry name="function">atcw32-&gt;ATCLoadAttachment</entry>
  <entry name="function">atcw32-&gt;ATCUnloadAttachment</entry>
  <entry name="function">dprw32-&gt;DPRSetConfig</entry>
  <entry name="function">dprw32-&gt;DPRSearchLDAP>/entry>
  </section>
  ```

- Configuring this rule with SSL involves installing the certificate submitted by the LDAP server into the trusted certification authorities store of the box where IDS is running. If the client program (IDS) is also to submit a certificate during the SSL hand-shake, then that certificate also needs to be installed into the trusted certification authorities store of the LDAP server.

DPRSearchWip

Use this function to return a list of records from a WIP database that matches the search fields specified.

Syntax

```
long _DSIAPI DPRSearchWip ( 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

Search fields can include date field values which should be specified in one of these formats:

Format	Description
YYYYMMDD	default; search any values equal to the given date.
<.YYYYMMDD	search any values less than the given date.
<=.YYYYMMDD	search any values less than or equal to the given date.
=.YYYYMMDD	search any values equal to the given date.
>.YYYYMMDD	search any values greater than the given date.
>=.YYYYMMDD	search any values greater or equal than the given date.
!=.YYYYMMDD	search any values not equal to the given date.

Using the STATUS, STATUSCODE, and KEYNAME input attachment variables, the DPRSearchWIP rule can also filter records by status code and return a sorted list of records based on input status code and sort key values. Information on input attachment variables can be found in the table below.

You can also filter the records returned by a report-to list for the user ID specified in the USERID input attachment variable if the USEREPORTTOLIST input attachment variable is present. The rule builds the user report-to list for filtering records in the following manner:

- If the user ID is found in the userinfo database, all user IDs reporting to the user ID provided, including that user ID, will be returned. For example: If user ID USER1 reports to user ID FORMAKER which reports to user ID DOCUCORP, and user ID DOCUCORP is provided in the USERID input attachment variable, the user list returned will contain user IDs DOCUCORP, FORMAKER, and USER1.

- If the user ID provided can not be found in the userinfo database, the system returns a list with one user entry that corresponds to the user ID provided.
- If the input attachment variable USERLIST is provided, the system does not use the userinfo database to build the report-to list; instead, it uses the user IDs provided in the attachment variable.

Attachment variables

Variable	Description
RESULTS	(Optional) If present, the rule checks that its value is SUCCESS. If the value is other than SUCCESS the rule will exit.
LOGINRESULT	(Optional) If present, the rule checks that its value is SUCCESS. If the value is other than SUCCESS the rule will exit.
USERID	If not found, the rule issues an error and exits. Use this variable to build a list of users that report to the user ID specified to filter the records returned from the WIP database. See also the USERLIST and USEREPORTTOLIST input attachment variables. While USERID is a required attachment variable, it is not checked against CURRUSER if the report to functionality is turned off.
PAGE	(Optional) The page number to display. The default is one (1).
PAGESIZE	(Optional) The number of records to display per page. The default is 20.
STARTRECORD STARTON	(Optional) The starting record number to display for the records matching the search criteria provided. The system first looks for STARTRECORD. If not found, it looks for STARTON. If a value is found, the start record defined overrides any values provided for the PAGE and PAGESIZE variables, which are otherwise used to compute the start record number. PAGE, PAGESIZE, NEXTPAGE, and PREVPAGE should not be used when using the STARTRECORD, STARTON, and MAXRECORDS input attachment variables.
MAXRECORDS	(Optional) Only used if the STARTRECORD or STARTON input attachment variables are present. If MAXRECORDS is present it will override any value provided for PAGESIZE. The default is 20.
KEYNAME	(Optional) The name of one of the fields in WIP DFD used to sort the records returned.
PARTIALMATCH	(Optional) If present, the rule conducts a partial match for the search values provided. The value provided for this variable is irrelevant; as long as the variable is present the option is enabled.
CASESENSITIVE	(Optional) If present, the rule will conduct a case sensitive search; otherwise, the rule will conduct a search using uppercase values. The value provided for this variable is irrelevant; as long as the variable is present the option will be enabled.

Variable	Description
FIELDNAME	(Optional) Any field name in the WIP DFD with a search value to be used for filtering the records returned, where FIELDNAME is the name of one of the fields in the DFD and the value provided for the input attachment variable is the search value. Multiple search fields are supported. Add a FIELDNAME/value entry for each search field that should be used to filter the records returned.
STATUS	(Optional) The WIP status used for filtering the list of STATUSCODE records returned. If STATUS is present, it will be used to look up an option with the same name under the STATUS_CD control group. If that option is not found, the value used for STATUS to look up the INI option is used. If STATUS cannot be found, the STATUSCODE input attachment variable is used and it follows the same logic applied for STATUS. If neither STATUS nor STATUSCODE exist, the value from the WIP option under the STATUS_CD control group is used and it will follow the same logic applied for STATUS. The default is WIP. If the value is an asterisk (*), the status code is not used as a filter for the records returned.
STATUSCODELIST	(Optional) - A comma-delimited list of status codes to search. If present, this variable overrides the behavior described for STATUS and STATUSCODE input message variables.
USERLIST	(Optional) A comma-delimited string of user IDs that should be used to build the report-to list for filtering records. If present, the code will use the user IDs provided to build the list instead of looking in the userinfo database.
USEREPORTTOLIST	(Optional) If present, the rule filters the records returned using a report-to list for the user ID provided in the USERID input attachment variable. Only the records which contain a current user that matches one of the users in the report-to list are returned.
CURRUSER	(Optional) If you specify this input attachment variable: `CURRUSER=~UNKNOWN~` the rule 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 checks the input attachment variable USEREPORTTOLIST as before and it has no effect if you specify *CURRUSER=~UNKNOWN~*.
RECORDID RECNUM UNIQUE_ID	(Optional) If a value is present for one of these input attachment variables, the system returns a single record matching the value provided without applying the search and filter logic.

Attachment outputs

Variable	Description
WIP	The WIP status generated from the WIP option in the STATUS_CD control group. The default is WIP.

Variable	Description
APPROVE	The Approve status generated from the Approve option in the STATUS_CD control group. The default is AP.
REJECT	The Reject status generated from the Reject option in the STATUS_CD control group. The default is RJ.
STATUS	The status definition used for the search. This is the same value determined by the logic defined under the STATUS and STATUSCODE input attachment variables. The default is WIP.
MORERECORDS	Indicates there are more records matching the search criteria. This variable is only present if there are more records.
NEXTPAGE	The next page number. This is only present if there are more records matching the search criteria.
PREVPAGE	The previous page number. This is only present if the current page is not the first page.
RESULTS	This attachment variable contains a value of SUCCESS if the rule ran successfully; otherwise, FAILURE.

INI options

You can use these INI options with this rule:

```
< WIPSearchFormatKeys >
    FieldName = Format
```

Where FieldName is one of the date fields in the DFD and Format is one of the formats supported:

- DX = Hex
- DT = ODBC date field
- D4 = A date value already in YYYYMMDD format

Here is an example of the different format specifiers:

```
< WIPSearchFormatKeys >
    CreateTime  = DX
    ModifyTime  = DT
    FromTime    = D4
```

If the date fields are not defined in the WIPSearchFormatKeys control group, the rule only checks these default date fields and assumes they are defined in hex format:

```
CREATETIME
FROMTIME
MODIFYTIME
TOTIME


< STATUS_CD >
    Approve = Definition of value for approve.
    Reject  = Definition of value for reject.
    WIP     = Definition of value for WIP.
    Status  = Definition of value for status.
```

For a definition of the APPROVE, REJECT or other options, you can refer to variables with the same name in the output attachment variables table.

Errors

These errors may be returned:

Error	Description
DPR0001	Attachment variable not found.
DPR0006	Virtual memory API error.
DPR0009	No matches found for the search criteria specified.
DPR0022	Error adding output attachment variable.
DPR0039	API error - describes the caller name and name of the API that failed.
DPR0068	INI group not configured correctly.
DPR0069	No search criteria specified.
DPR0070	DFD variable not configured correctly.
DPR0071	Invalid data specified for input attachment variable.

See also

DPRApproveWipRecords on page 73

DPRCheckWipRecords on page 88

DPRGetWipList on page 150

DPRGetWipFormset on page 153

DPRGetWipRecipients on page 156

DPRSearchWip on page 246

DPRUpdateWipRecords on page 287

DPRSendFormsetXML

Use this rule to convert the form set specified in the DSI variable DPRFORMSET into an XML file in memory and then send this XML file as an attachment to the IDS client.

Syntax

```
long _DSIAPI DPRSendFormsetXML ( 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

The delimiter name for this attachment can be specified as this rule's parameter. If not specified it defaults to DOCUMENTSTREAM. The default is used if no rule parameter is provided.

If the DPRFORMSET DSI variable does not exist this rule does nothing and no error message is produced.

This rule runs on DSI_MSGRUNR.

See also

DPRLoadXMLAttachment on page 188

DPRLoadedXML2Formset on page 184

DPRUpdateFromMRL on page 282

DPRFilterFormsetForms on page 122

DPRSortFormsetForms on page 260

DPRGetFormList on page 137

DPRGetHTMLForms on page 141

DPRSendMultiFiles

Use this rule to send multiple files to an attachment one by one, so they can be received at the other end. This rule supports text and binary files. The size of file is limited to the queue message size.

Syntax

```
long _DSIAPI DPRSendMultiFiles ( DSIHANDLE hdsi,
                                  char * pszParms,
                                  ULONG  ulMsg,
                                  ULONG long  ulOptions )
```

Parameters

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

Returns Success or failure

Errors This rule can return these messages:

Error	Description
DPR0007	The INI option #INIOPTION,# could not be located
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.
DPR0056	The INI option #INIOPTION,# does not contain a valid value in the group #INIGROUP,#.

Example Here is an example:

```
[ ReqType:WLGN ]
function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig
function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSendMultiFiles, DPRWIPTABLE
function = dprw32->DPRGetWipList
```

where DPRWIPTABLE specifies this INI control group:

```
< DPRWIPTable >
    WIPTABLE1 = WIPTABLE,wip.asp,TEXT
    WIPTABLE2 = ABCTABLE,test.asp,TEXT
```

The INI value is composed of attachment name, sending file, and file type.

DPRSendVersion

Use this rule to gather version information about these DLLs:

- DPRW32.DLL
- PDFW32.DLL

Syntax

```
long _DSIAPI DPRSendVersion ( 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

Attachment variables

This rule creates the attachment record LIBRARIES with these variables:

Variable	Description
NAME	name of the DLL (DPR or PDF)
VERSION	version of the DLL, string like 100.002.001
DATE	date the DLL was compiled as MMM DD YYYY
TIME	time the DLL was compiled as HH:MM:SS in 24-hour format

This rule creates an attachment variable RESULTS with the value SUCCESS.

Returns

Success or failure

DPRSet2ImageScope

Use this rule to change the scope of fields from form level to image level.

Syntax

```
long _DSIAPI DPRSet2ImageScope ( 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

Here is an example of how you can use this rule to change the scope of the ACCTNUM1 and Service Address fields:

```
function = dprw32->DPRSet2ImageScope,ACCTNUM1;Service Address
```

Returns Success or failure

DPRSetConfig

Use this rule to set the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.

This rule runs on DSI_SMGRUNF and DSI_MSGRUNR. On DSI_SMGRUNF it saves the current INI context in the DSI variable INICONTEXT. The rule then loads all of the INI files specified under the INIFile option in the Config:XXX control group.

The values assigned to this option indicate the value of the attachment variable CONFIG. If you have multiple INI file option lines, the system loads all of the lines.

The latter in the INI file is appended to the end of INI context in memory. After all the INI files are loaded, the current INI context (usually from the DAP.INI file) is appended to the resulting list.

On DSI_SMGRUNR, the system restores the current INI context saved in the DSI variable INICONTEXT, destroys DSI variable INICONTEXT, and destroys the INI context created on DSI_RUNF message.

Syntax

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

Parameters

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

This rule uses the following control group setting in the DAP.INI file to define the INI file name, where CGF is the name of a configurations group.

```
< Config:CFG >
    INIFile =
```

For each Config:CFG control group, you must make an entry in the Configurations control group, as shown below:

```
< Config:UTILITY >
    INIFile = utility.ini
< Configurations >
    Config = UTILITY
```

Detecting MRL changes

Documaker Bridge automatically detects changes made to a Studio master resource library (MRL) and flashes cached files. This keeps you from having to manually restart IDS when MRL updates are made.

The DPRSetConfig rule detects the update and flashes cached files. Instances of IDS running Documaker Server (GenData) using the same MRL are terminated and then restarted so the GenData program will realize the change to the MRL.

Keep in mind the only updates to files in Library manager are detected. MRL changes that are not part of Library manager are ignored.

NOTE: This is helpful in situations where your MRL changes frequently. Once you are in production mode, you should schedule updates to your production MRL at times when no one is using the system.

Detecting INI changes

The DPRSetConfig rule reloads the DAP.INI file when a change is detected. The system also updates the list of files loaded for CONFIG based on any changes to the files listed as INIFile option entries in DAP.INI file.

If any of the configuration files that correspond to INIFile option entries for a CONFIG change, these files are also reloaded.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0013	The initialization file, FILENAME, could not be loaded.
DPR0024	Cannot create DSI variable VARIABLE.

DPRSpellCheck

Use this rule to spell check an XML document tree. The user dictionary (USER.TLX) and main dictionaries (SSCEAM.TLX and SSCEAM2.CLX) are required. If a hyphen is at the end of current text line, the rule removes the hyphen and moves the first word on the next text line to the end of current line.

Syntax

```
long _DSIAPI DPRSpellCheck ( DSIHANDLE hdsi,
                             char * pszParms,
                             unsigned ulMsg,
                             unsigned ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
ImportFile	The name of the input XML file to check spelling with a full path, such as: `d:\ids2.0\spell\spellXML_input.xml`

Attachment outputs

Variables	Description
ExportFile	The name of the output XML file for new XML document tree that includes spelling check information, such as `d:\ids2.0\spell\spellXML_output.xml`

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIABLE,# in the attachment list.
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.
DPR0051	Failed to unload XML file #FILE,# in #LOCATION,#.
DPR0063	Failed to load XML file #FILE,# in #LOCATION,#.

Here is an export file layout:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SPELLER TYPE="IENTRY" VERSION="3.1">
<FIELD NAME="document.forms[0].elements[6].value">
<P>Now is the timme for
<WORD VALUE="timme" POS="11">
<CHOICE MATCH="TRUE">time</CHOICE>
<CHOICE>timed</CHOICE>
<CHOICE>timer</CHOICE>
<CHOICE>times</CHOICE>
<CHOICE>timber</CHOICE>
<CHOICE>timbre</CHOICE>
</WORD>
</P>
<P>a new begning for<WORD VALUE="begning" POS="6">
<CHOICE MATCH="TRUE">begging</CHOICE>
</WORD>
</P>
<P>successs.<WORD VALUE="successs" POS="0">
<CHOICE MATCH="TRUE">success</CHOICE>
<CHOICE>successes</CHOICE>
</WORD>
</P>
</FIELD>
<FIELD NAME="document.forms[0].elements[7].value">
iPPS Livve
<WORD VALUE="iPPS" POS="0">
<CHOICE MATCH="TRUE">PP</CHOICE>
<CHOICE>PS</CHOICE>
<CHOICE>its</CHOICE>
</WORD>
<WORD VALUE="Livve" POS="5">
<CHOICE MATCH="TRUE">Live</CHOICE>
<CHOICE>Five</CHOICE>
<CHOICE>Give</CHOICE>
<CHOICE>Life</CHOICE>
<CHOICE>Like</CHOICE>
<CHOICE>Line</CHOICE>
<CHOICE>Love</CHOICE>
</WORD>
</FIELD>
<FIELD NAME="document.forms[0].elements[8].value">
begning
<WORD VALUE="begning" POS="0">
<CHOICE MATCH="TRUE">begging</CHOICE>
</WORD>
</FIELD>
<FIELD NAME="document.forms[0].elements[9].value">2727 Paces Ferry
Road</FIELD>
<FIELD NAME="document.forms[0].elements[10].value">Suite II-900</
FIELD>
<FIELD NAME="document.forms[0].elements[11].value">Atlanta</FIELD>
<FIELD NAME="document.forms[0].elements[12].value">GA</FIELD>
<FIELD NAME="document.forms[0].elements[13].value">30339</FIELD>
<WORD VALUE="spellinng" POS="10">
```

```
<CHOICE MATCH="TRUE">spelling</CHOICE>
<CHOICE>spellings</CHOICE>
<CHOICE>speckling</CHOICE>
<CHOICE>spellbind</CHOICE>
<CHOICE>spewing</CHOICE>
<CHOICE>telling</CHOICE>
</WORD>
<WORD VALUE="neew" POS="39">
<CHOICE MATCH="TRUE">nee</CHOICE>
<CHOICE>new</CHOICE>
<CHOICE>need</CHOICE>
<CHOICE>knew</CHOICE>
<CHOICE>news</CHOICE>
</WORD>
</SPELLER>
*
*****************************************************************/
```

Here is an example of the request type in the docserve.xml file:

```
<section name="ReqType:SPELL">
    <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->;DPRSpellCheck</entry>
</section>
```

Attachment variables:

- CONFIG
- ImportFile
- ExportFile

DPRSortFormsetForms

Use this rule to sort the form list.

Syntax

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

Parameters

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

The DPRORDERBY attachment variable is checked and you can have the following values FORMNAME and FORMDESCRIPTION in any order and just like in SQL keyword DESC or DESCENDING.

Here are some examples:

```
DPRORDERBY=FORMNAME DESC,FORMDESCRIPTION
DPRORDERBY=FORMDESCRIPTION
DPRORDERBY=FORMDESCRIPTION DESC, FORMNAME DESC
```

The real sorting is done within groups, the same as if the SQL had ORDER BY KEY1, KEY2 ... (value of DPRORDERBY).

See also

DPRLoadXMLAttachment on page 188

DPRLoadedXML2Formset on page 184

DPRSendFormsetXML on page 251

DPRUpdateFromMRL on page 282

DPRFilterFormsetForms on page 122

DPRGetFormList on page 137

DPRGetHTMLForms on page 141

DPRTemporaryXMLFile

Use this rule to load and unload XML files into or from a temporary file.

Syntax

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

Parameters

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

When loading an XML file, this rule locates the DSI variable *DPRXMLFORMSET* to retrieve the XML document handle. It then unloads it into a temporary XML file with a unique name.

The file name is assigned as the value of a new attachment variable. The new attachment variable name is taken from pszParms. If pszParms is empty, the system uses *XMLFORMSETFILE* as the default variable name.

When unloading an XML file, this rule locates the temporary XML file and then converts it back into XML document format. If the debug option is off, the temporary XML file is then removed.

NOTE: You can use this rule with your Java rules instead of using SENDBACKPAGE attachment variable.

Returns

Success or failure

DPRTblLookUp

Use this rule to generate an XML document that contains table entries for a table ID in a table file.

Syntax

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

Parameters

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

This rule creates a DSI global variable TEMPLATESOURCEDOCUMENT for the document handle on the RUNF message for other rules that follow. The global variable is removed on the RUNR message.

The table entries are added as records to the output message. If the KEEP rule argument or input attachment variable is present, the rule also writes the XML document to disk and adds the TBLLKUPFILE attachment variable to the input and output messages for other rules that follow.

Each table entry returned contains these elements:

Element	Description
ENTRY_NAME	The key.
DESCRIP	A description.
RETURNS	The value returned. See the TABLERETURNS input attachment variable.

Attachment variables

Attachment	Description
CONFIG	A configuration value for a master resource library in the DAP.INI file for IDS.
TABLEFILE	The name and path of a table file accessible to IDS. If you omit the path, the DPRTblLookUp rule looks for the table file path in the following manner (using the first path found): - Look in the TableLib option in the MasterResource control group - Look in the DefLib option in the MasterResource control group - Set the path to the current IDS directory path value
TABLEID	The name of a table in the table file for which to retrieve the entries.

Attachment	Description
TABLERETURNS	(required) An indicator of how to return the entries for a table. You can specify these values: KEY - Return the key value in the returns element for each entry. KEY only - Return the key value in the returns element for each entry. Description - Return the description value in the returns element for each entry. Description only - Return the description value in the returns element for each entry. Key & Description - Return the key followed by a space followed by the description in the returns element for each entry. Key and description - Return the key followed by a space followed by the description in the returns element for each entry. Description & Key - Return the description followed by a space followed by the key in the returns element for each entry. Description and key - Return the description followed by a space followed by the key in the returns element for each entry. Nothing - Do not return anything for the returns element for each entry.
PRINTPATH	(Optional) A path accessible to IDS where the output file will be written to if the KEEP rule argument or input attachment variable is present. If you omit this value, the rule uses the current IDS directory.
TBLLKUPFILE	(Optional) A path and file name for the output file that will be written to disk if the KEEP rule argument or input attachment variable is present.
KEEP	(Optional) If this variable is present, the rule writes the XML document to disk and adds the TBLLKUPFILE input/output attachment variable with the path and file name of the output file.

Attachment outputs

Attachment	Description
TBLLKUPFILE	Only present if the KEEP input attachment variable or rule argument is present. It contains the path and file name of the output file.
RESULTS	Success or failure

Arguments

Argument	Description
KEEP	(Optional) If this rule argument is present the rule writes the XML document to disk and adds the TBLLKUPFILE input/output attachment variable with the full/relative path and file name of the output file.

Example 1

Here is the request type for this example:

```
<section name="ReqType:TBLLKUP">
        <entry name="function">atcw32->;ATCLoadAttachment</entry>
        <entry name="function">atcw32->;ATCUnloadAttachment</entry>
        <entry name="function">dprw32->;DPRSetConfig</entry>
        <entry name="function">dprw32->;DPRInitLby</entry>
        <entry name="function">dprw32->;DPRTblLookUp</entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,SOURCE,SOURCE</entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,DOCTYPE,DOCTYPE</entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,TEMPLATE,XSLTFILE</entry>
        <entry
name="function">java;com.docucorp.ids.rules.XsltTransformRule;TF1;t
ransaction;transform;</entry>
</section>
Here is the input message for Example 1:
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLLKUP</REQTYPE>
                    <UNIQUE_ID>5533591529132872004-0-Thread-1</
UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="CONFIG">AMERGEN</VAR>
                    <VAR NAME="KEEP"></VAR>
                    <VAR
NAME="TABLEFILE">C:\rp\mstrres\insure\table\mktmsg.dbf</VAR>
                    <VAR NAME="TABLEID">mktmsg</VAR>
                <VAR NAME="TABLERETURNS">KEY &amp; DESCRIPTION</VAR>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the output message for this example:

```
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLLKUP</REQTYPE>
                    <UNIQUE_ID>5533591529132872004-0-Th</UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="DOCTYPE">htm</VAR>
                    <VAR NAME="RESULTS">SUCCESS</VAR>
                    <VAR NAME="SERVERTIMESPENT">0.203</VAR>
                    <VAR NAME="SOURCE">TBLLKUPFILE</VAR>
                    <VAR NAME="TBLLKUPFILE">0rc74eSla-
Bh5yuEiiOczVSVP9hIrvVaIyXg0PoiSFo8Y.xml</VAR>
```

```
                    <VAR NAME="XSLOUTPUT">7706561529132872004-0-BLP-
0.htm</VAR>
                    <VAR NAME="XSLTFILE">tbllkup.xsl</VAR>
                    <ROWSET NAME="RECORDS">
                        <ROW NUM="1">
                            <VAR NAME="ENTRY_NAME">Coverage</VAR>
                         <VAR NAME="DESCRIP">Did you know you could
save 5% off your policy premium if you place more than one policy
with Amergen?</VAR>
                          <VAR NAME="RETURNS">Coverage Did you know
you could save 5% off your policy premium if you place more than one
policy with Amergen?</VAR>
                        </ROW>
                        <ROW NUM="2">
                            <VAR NAME="ENTRY_NAME">Greeting</VAR>
                            <VAR NAME="DESCRIP">Hello World</VAR>
                      <VAR NAME="RETURNS">Greeting Hello World</VAR>
                        </ROW>
                        <ROW NUM="3">
                            <VAR NAME="ENTRY_NAME">Technique</VAR>
                           <VAR NAME="DESCRIP">Are you using the 5
techniques to manage risk?</VAR>
                            <VAR NAME="RETURNS">Technique Are you
using the 5 techniques to manage risk?</VAR>
                        </ROW>
                    </ROWSET>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the Xslt template, which is used by the XsltTransformRule:

```
    <xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">


      <xsl:output method="html"/>
      <xsl:template match="/">
        <html>
        <head>
        <xsl:element name="script">
            <xsl:attribute name="language">JavaScript 1.2</
xsl:attribute>
        <xsl:attribute name="type">text/javascript</xsl:attribute>
            <xsl:comment>
                <![CDATA[

                function setValue(obj){

                    if (obj.value != null)
                        window.returnValue = obj.value;
                    else
                        window.returnValue = "";

                    window.close();
                }

                ]]>
             </xsl:comment>
        </xsl:element>
        </head>
        <body bgcolor="#f2eddb" onload="window.focus();">
            <table width="100%" height="100%">
                <tr>
                    <td align="center" valign="top">
                   <select name="Lookup" onChange="setValue(this);"
value="">
                            <xsl:call-template name="process" />
                        </select>
                    </td>
                </tr>
                <tr>
                    <td align="center" valign="center">
                    <input type="button" value="close" name="close"
onclick="self.close();"/>
                    </td>
                </tr>
            </table>
        </body>
        </html>
      </xsl:template>

      <xsl:template name="process">
            <br/>
            <xsl:for-each select="//DOCUMENT/ENTRIES/INDEX">
                <xsl:variable name="key"
select="COLUMN[@NAME='ENTRY_NAME']/."/>
                <xsl:variable name="description"
select="COLUMN[@NAME='DESCRIP']/."/>
                <option value="{$description}"><xsl:value-of
select="$key"/></option><br/>
            </xsl:for-each>
      </xsl:template>

    </xsl:stylesheet>
```

Example 2 Here is the request type for this example:

```
     <section name="ReqType:TBLKUP2">
         <entry name="function">atcw32->;ATCLoadAttachment</entry>
         <entry name="function">atcw32->;ATCUnloadAttachment</entry>
         <entry name="function">dprw32->;DPRSetConfig</entry>
         <entry name="function">dprw32->;DPRInitLby</entry>
         <entry name="function">dprw32->;DPRTblLookUp</entry>
         <entry name="function">dprw32-
>;DPRGetInitValue,TBLLKUP,DOCTYPE,FILETYPE</entry>
         <entry name="function">dprw32-
>;DPRGetInitValue,TBLLKUP,HTMTEMPLATE,TEMPLATE</entry>
         <entry name="function">dprw32->;DPRTransform</entry>
     </section>
```

Here is the input message for this example:

```
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLKUP2</REQTYPE>
                    <UNIQUE_ID>4809681331132872004-0-Thread-2</
UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="CONFIG">AMERGEN</VAR>
                    <VAR
NAME="TABLEFILE">C:\rp\mstrres\insure\table\mktmsg.dbf</VAR>
                    <VAR NAME="TABLEID">TEST</VAR>
                <VAR NAME="TABLERETURNS">KEY &amp; DESCRIPTION</VAR>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the output message for this example:

```
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLKUP2</REQTYPE>
                    <UNIQUE_ID>4809681331132872004-0-Th</UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="FILETYPE">htm</VAR>
                    <VAR NAME="RESULTS">SUCCESS</VAR>
                    <VAR NAME="SERVERTIMESPENT">0.094</VAR>
                    <VAR NAME="TEMPLATE">tbllkup.htm</VAR>
                    <VAR
NAME="TRANSFORMFILE">0uyQNhTch_idAmANizRkyh3CMnFQX5j7n_BcXZC0lRMaX.
htm</VAR>
                    <ROWSET NAME="RECORDS">
                        <ROW NUM="1">
                            <VAR NAME="ENTRY_NAME">Entry1</VAR>
                          <VAR NAME="DESCRIP">Entry Number One</VAR>
                            <VAR NAME="RETURNS">Entry1 Entry Number
One</VAR>
                        </ROW>
                        <ROW NUM="2">
                            <VAR NAME="ENTRY_NAME">Entry2</VAR>
                          <VAR NAME="DESCRIP">Entry Number two</VAR>
                            <VAR NAME="RETURNS">Entry2 Entry Number
two</VAR>
```

```
                        </ROW>
                        <ROW NUM="3">
                            <VAR NAME="ENTRY_NAME">Entry3</VAR>
                        <VAR NAME="DESCRIP">Entry Number three</VAR>
                            <VAR NAME="RETURNS">Entry3 Entry Number
three</VAR>
                        </ROW>
                    </ROWSET>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the HTML template used by the DPRTransform rule for this example:

```
    <html>
    <head>
    <script language="javascript">

        function setValue(obj){

            if (obj.value != null)
                window.returnValue = obj.value;
            else
                window.returnValue = "";

            window.close();

        }

    </script>
    </head>
    <body bgcolor="#f2eddb" onload="window.focus();">
        <table width="100%" height="100%">
            <tr>
                <td align="center" valign="top">
                   <select name="Lookup" onChange="setValue(this);"
value="">
                    <!-- DCL BEGIN
SECTION;NAME=descendant::ENTRIES;LOOP=descendant::INDEX;FOR-
EACH=INDEX;-->
                        <option value="<%./
descendant::COLUMN[attribute::NAME="DESCRIP"],%>">
                        <%./
descendant::COLUMN[attribute::NAME="ENTRY_NAME"],%>
                        </option>
                    <!-- DCL END SECTION -->
                    </select>
                </td>
            </tr>
            <tr>
                <td align="center" valign="center">
                    <input type="button" value="close" name="close"
onclick="self.close();"/>
                </td>
            </tr>
        </table>
    </body>
    </html>
```

Example 3 Here is the request type for this example:

```
<section name="ReqType:TBLLKUP3">
        <entry name="function">atcw32->;ATCLoadAttachment</entry>
     <entry name="function">atcw32->;ATCUnloadAttachment</entry>
        <entry name="function">dprw32->;DPRSetConfig</entry>
        <entry name="function">dprw32->;DPRInitLby</entry>
        <entry name="function">dprw32->;DPRTblLookUp</entry>
        <entry name="function">atcw32->;ATCDumpAttachment,ATC1</
entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,SOURCEVAR,SOURCE</entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,DOCTYPE,FILETYPE</entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,HTMTEMPLATE,TEMPLATE</entry>
        <entry name="function">atcw32->;ATCDumpAttachment,ATC2</
entry>
        <entry name="function">dprw32->;DPRTransform</entry>
</section>
Here is the input message for this example:
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLLKUP3</REQTYPE>
                    <UNIQUE_ID>5060623132132872004-0-Thread-3</
UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="CONFIG">AMERGEN</VAR>
                    <VAR NAME="KEEP"></VAR>
                    <VAR
NAME="TABLEFILE">C:\rp\mstrres\insure\table\mktmsg.dbf</VAR>
                    <VAR NAME="TABLEID">mktmsg</VAR>
                    <VAR NAME="TABLERETURNS">KEY</VAR>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the output message for this example:

```
     Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
```

```
                    <REQTYPE>TBLLKUP3</REQTYPE>
                    <UNIQUE_ID>5060623132132872004-0-Th</UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="FILETYPE">htm</VAR>
                    <VAR NAME="RESULTS">SUCCESS</VAR>
                    <VAR NAME="SERVERTIMESPENT">0.093</VAR>
                   <VAR NAME="SOURCE">LOOKUPVAR.OUTPUT.TBLLKUPFILE</
VAR>
                    <VAR
NAME="TBLLKUPFILE">0swwpsCxVzAQvwEKFySYeoeIKkRN7wGG3_ScpmwGuKqLZ.xm
l</VAR>
                    <VAR NAME="TEMPLATE">tbllkup.htm</VAR>
                    <VAR NAME="TRANSFORMFILE">0pDp_S_-
UehF1YuqKukd0oR6pqgrTMle4AZxuwguYRrXj.htm</VAR>
                    <ROWSET NAME="RECORDS">
                        <ROW NUM="1">
                            <VAR NAME="ENTRY_NAME">Coverage</VAR>
                         <VAR NAME="DESCRIP">Did you know you could
save 5% off your policy premium if you place more than one policy
with Amergen?</VAR>
                            <VAR NAME="RETURNS">Coverage</VAR>
                        </ROW>
                        <ROW NUM="2">
                            <VAR NAME="ENTRY_NAME">Greeting</VAR>
                            <VAR NAME="DESCRIP">Hello World</VAR>
                            <VAR NAME="RETURNS">Greeting</VAR>
                        </ROW>
                        <ROW NUM="3">
                            <VAR NAME="ENTRY_NAME">Technique</VAR>
                           <VAR NAME="DESCRIP">Are you using the 5
techniques to manage risk?</VAR>
                            <VAR NAME="RETURNS">Technique</VAR>
                        </ROW>
                    </ROWSET>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the HTML template used by the DPRTransform rule for this example:

```
<html>
    <head>
    <script language="javascript">

        function setValue(obj){

            if (obj.value != null)
                window.returnValue = obj.value;
            else
                window.returnValue = "";

            window.close();

        }

    </script>
    </head>
    <body bgcolor="#f2eddb" onload="window.focus();">
        <table width="100%" height="100%">
            <tr>
                <td align="center" valign="top">
                   <select name="Lookup" onChange="setValue(this);"
value="">
                    <!-- DCL BEGIN
SECTION;NAME=descendant::ENTRIES;LOOP=descendant::INDEX;FOR-
EACH=INDEX;-->
                        <option value="<%./
descendant::COLUMN[attribute::NAME="DESCRIP"],%>">
                        <%./
descendant::COLUMN[attribute::NAME="ENTRY_NAME"],%>
                        </option>
                    <!-- DCL END SECTION -->
                    </select>
                </td>
            </tr>
            <tr>
                <td align="center" valign="center">
                    <input type="button" value="close" name="close"
onclick="self.close();"/>
                </td>
            </tr>
        </table>
    </body>
    </html>
```

DPRTransform

Use this rule to transform an XML document into an output file with embedded data. The rule uses a template with embedded XSL to transform the output template file.

Syntax

```
long _DSIAPI DPRTransform ( DSIHANDLE hdsi,
                            char * pszParms,
                            unsigned long  ulMsg,
                            unsigned long  ulOptions )
```

Parameters

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

Attachment variables

Attachment	Description
CONFIG	A configuration value for a master resource library in the DAP.INI for IDS.
SOURCE	(Optional) A path and file name of an XML document to process. It must be accessible to IDS. If this variable is not present the rule will look for a DSI global variable TEMPLATESOURCEDOCUMENT which must be set by a rule run in the same request type prior to this rule (see DPRTblLookUp rule, feature 1612 for an example).
TEMPLATE	A full path and file name of a template with embedded XSL to use for the transformation. It must be accessible to IDS.
FILETYPE	The extension of the output file.
PRINTPATH	(Optional) A path accessible to IDS where the output file will be written to. If a value is not provided the rule will use the current IDS directory.
TRANSFORMFILE	(Optional) The path and file name of the output file.

Attachment outputs

Attachment	Description
TRANSFORMFILE	The path and file name of the output file.
RESULTS	Success or failure.
DPRTRANSFORMFI LE	The output template, sent as an output attachment, which is part of the output message.

Example 1

Here is the request type for Example 1:

```
<section name="ReqType:TBLKUP2">
```

```
    <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
    <entry name="function">dprw32->;DPRSetConfig</entry>
    <entry name="function">dprw32->;DPRInitLby</entry>
    <entry name="function">dprw32->;DPRTblLookUp</entry>
    <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,DOCTYPE,FILETYPE</entry>
    <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,HTMTEMPLATE,TEMPLATE</entry>
    <entry name="function">dprw32->;DPRTransform</entry>
</section>
```

Here is the input message for Example 1:

```
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLKUP2</REQTYPE>
                    <UNIQUE_ID>4809681331132872004-0-Thread-2</
UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="CONFIG">AMERGEN</VAR>
                    <VAR
NAME="TABLEFILE">C:\rp\mstrres\insure\table\mktmsg.dbf</VAR>
                    <VAR NAME="TABLEID">TEST</VAR>
                <VAR NAME="TABLERETURNS">KEY &amp; DESCRIPTION</VAR>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the output message for Example 1:

```
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLKUP2</REQTYPE>
                    <UNIQUE_ID>2399062548162892004-0-Th</UNIQUE_ID>
                    <ATTACHMENT TYPE="BINARY">
                        <DELIMITER>DPRTRANSFORMFILE</DELIMITER>
                    </ATTACHMENT>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="FILETYPE">htm</VAR>
                    <VAR NAME="RESULTS">SUCCESS</VAR>
                    <VAR NAME="SERVERTIMESPENT">0.094</VAR>
                    <VAR NAME="TEMPLATE">tbl1kup.htm</VAR>
                  <VAR NAME="TRANSFORMFILE">0vQo1gBFkriVOqxB4wBd5XU-
An7I2-Dhdpq-alQGA53LY.htm</VAR>
                    <ROWSET NAME="RECORDS">
                        <ROW NUM="1">
                            <VAR NAME="ENTRY_NAME">Entry1</VAR>
                          <VAR NAME="DESCRIP">Entry Number One</VAR>
                            <VAR NAME="RETURNS">Entry1 Entry Number
One</VAR>
                        </ROW>
                        <ROW NUM="2">
                            <VAR NAME="ENTRY_NAME">Entry2</VAR>
                          <VAR NAME="DESCRIP">Entry Number two</VAR>
                            <VAR NAME="RETURNS">Entry2 Entry Number
two</VAR>
                        </ROW>
                        <ROW NUM="3">
                            <VAR NAME="ENTRY_NAME">Entry3</VAR>
                        <VAR NAME="DESCRIP">Entry Number three</VAR>
                            <VAR NAME="RETURNS">Entry3 Entry Number
three</VAR>
                        </ROW>
                    </ROWSET>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the HTML template for Example 1:

```
    <html>
    <head>
    <script language="javascript">

        function setValue(obj){

            if (obj.value != null)
                window.returnValue = obj.value;
            else
                window.returnValue = "";

            window.close();

        }

    </script>
    </head>
    <body bgcolor="#f2eddb" onload="window.focus();">
        <table width="100%" height="100%">
            <tr>
                <td align="center" valign="top">
                   <select name="Lookup" onChange="setValue(this);"
value="">
                    <!-- DCL BEGIN
SECTION;NAME=descendant::ENTRIES;LOOP=descendant::INDEX;FOR-
EACH=INDEX;-->
```

```
                        <option value="<%./
descendant::COLUMN[attribute::NAME="DESCRIP"],%>">
                        <%./
descendant::COLUMN[attribute::NAME="ENTRY_NAME"],%>
                        </option>
                    <!-- DCL END SECTION -->
                    </select>
                </td>
            </tr>
            <tr>
                <td align="center" valign="center">
                    <input type="button" value="close" name="close"
onclick="self.close();"/>
                </td>
            </tr>
        </table>
    </body>
    </html>
```

Example 2

Here is the request type for Example 2:

```
    <section name="ReqType:TBLLKUP3">
        <entry name="function">atcw32->;ATCLoadAttachment</entry>
    <entry name="function">atcw32->;ATCUnloadAttachment</entry>
        <entry name="function">dprw32->;DPRSetConfig</entry>
        <entry name="function">dprw32->;DPRInitLby</entry>
        <entry name="function">dprw32->;DPRTblLookUp</entry>
        <entry name="function">atcw32->;ATCDumpAttachment,ATC1</
entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,SOURCEVAR,SOURCE</entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,DOCTYPE,FILETYPE</entry>
        <entry name="function">dprw32->
;DPRGetInitValue,TBLLKUP,HTMTEMPLATE,TEMPLATE</entry>
        <entry name="function">atcw32->;ATCDumpAttachment,ATC2</
entry>
        <entry name="function">dprw32->;DPRTransform</entry>
    </section>
```

Here is the input message for Example 2:

```
    Content-Type: text/xml
    Content-Transfer-Encoding: 8bit

    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLLKUP3</REQTYPE>
                    <UNIQUE_ID>5060623132132872004-0-Thread-3</
UNIQUE_ID>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="CONFIG">AMERGEN</VAR>
                    <VAR NAME="KEEP"></VAR>
                    <VAR
NAME="TABLEFILE">C:\rp\mstrres\insure\table\mktmsg.dbf</VAR>
                    <VAR NAME="TABLEID">mktmsg</VAR>
                    <VAR NAME="TABLERETURNS">KEY</VAR>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the output message for Example 2:

```
    Content-Type: text/xml
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
        <SOAP-ENV:Body>
            <DSIMSG VERSION="100.020.0">
                <CTLBLOCK>
                    <REQTYPE>TBLLKUP3</REQTYPE>
                    <UNIQUE_ID>4157034449162892004-0-Th</UNIQUE_ID>
                    <ATTACHMENT TYPE="BINARY">
                        <DELIMITER>DPRTRANSFORMFILE</DELIMITER>
                    </ATTACHMENT>
                </CTLBLOCK>
                <MSGVARS>
                    <VAR NAME="FILETYPE">htm</VAR>
                    <VAR NAME="RESULTS">SUCCESS</VAR>
                    <VAR NAME="SERVERTIMESPENT">0.094</VAR>
                   <VAR NAME="SOURCE">LOOKUPVAR.OUTPUT.TBLLKUPFILE</
VAR>
                    <VAR NAME="TBLLKUPFILE">0kCIZfRhu_QkisrZ6tCkg-
ScKnfxexBzy0EwXmCPRMaX2.xml</VAR>
                    <VAR NAME="TEMPLATE">tbllkup.htm</VAR>
                <VAR NAME="TRANSFORMFILE">0FS7HpzYXvT33h_JxsFsQgV_p-
UZmoUEn-OZyu5jrBLOK.htm</VAR>
                    <ROWSET NAME="RECORDS">
                        <ROW NUM="1">
                            <VAR NAME="ENTRY_NAME">Coverage</VAR>
                         <VAR NAME="DESCRIP">Did you know you could
save 5% off your policy premium if you place more than one policy
with Amergen?</VAR>
                            <VAR NAME="RETURNS"Coverage</VAR>
                        </ROW>
                        <ROW NUM="2">
                            <VAR NAME="ENTRY_NAME">Greeting</VAR>
                            <VAR NAME="DESCRIP">Hello World</VAR>
                            <VAR NAME="RETURNS">Greeting</VAR>
                        </ROW>
                        <ROW NUM="3">
                            <VAR NAME="ENTRY_NAME">Technique</VAR>
                           <VAR NAME="DESCRIP">Are you using the 5
techniques to manage risk?</VAR>
                            <VAR NAME="RETURNS">Technique</VAR>
                        </ROW>
                    </ROWSET>
                </MSGVARS>
            </DSIMSG>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Here is the HTML Template for Example 2:

```
    <html>
    <head>
    <script language="javascript">

        function setValue(obj){

            if (obj.value != null)
                window.returnValue = obj.value;
            else
                window.returnValue = "";

            window.close();

        }

    </script>
    </head>
    <body bgcolor="#f2eddb" onload="window.focus();">
        <table width="100%" height="100%">
            <tr>
                <td align="center" valign="top">
                   <select name="Lookup" onChange="setValue(this);"
value="">
                    <!-- DCL BEGIN
SECTION;NAME=descendant::ENTRIES;LOOP=descendant::INDEX;FOR-
EACH=INDEX;-->
                        <option value="<%./
descendant::COLUMN[attribute::NAME="DESCRIP"],%>">
                        <%./
descendant::COLUMN[attribute::NAME="ENTRY_NAME"],%>
                        </option>
                    <!-- DCL END SECTION -->
                    </select>
                </td>
            </tr>
            <tr>
                <td align="center" valign="center">
                    <input type="button" value="close" name="close"
onclick="self.close();"/>
                </td>
            </tr>
        </table>
    </body>
    </html>
```

DPRUnloadExportFile

Use this rule to unload an export file from a form set (FAP file) in memory. This rule runs on DSI_MSGRUNR. The output file format is controlled by the FILETYPE attachment variable. Set it to *XML* to create XML files, otherwise the system creates a V2 file.

NOTE: You can use the DPRPrint and DPRUnloadExportFile rules to 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. See Generating File Names Based on Transaction Values on page 215 for more information.

Syntax

```
long _DSIAPI DPRUnloadExportFile ( DSIHANDLE hdsi,
                                   char * pszParms,
                                   unsigned long  ulMsg,
                                   unsigned long  ulOptions )
```

Parameters

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

To use this rule you will need to specify the following rule name:

```
dprw32->DPRUnloadExportFile
```

Attachment variables

This rule expects these attachment variables:

Variable	Description
DPRFORMSET	The form set to export. This form set is created by some other rule, such as the DPRLoadImportFile rule.
EXPORT	The name of the output file.
APPENDEDEXPORT	If this variable is present in the attachment, the output is appended to the file specified in the EXPORT attachment variable. If you omit this value, the system uses the AppendedExport option in the ExpFile_CD control group to determine if the output should be appended to the export file. The default for the AppendedExport option is No.
EXPORTRECIPS	If this variable is present in the attachment, the output export file will contain recipient information. If you omit this value, the system uses the value in the AFEExportRecips option in the ExpFile_CD control group to determine if the output should contain the recipient information. The default for AFEExportRecips option is No.

Variable	Description
KEYID	Specifies the KeyID. If you omit this value the system uses the attachment variable specified in the TransactionID option in the DocSetNames control group.
TRANCODE	Specifies the WIP transaction code.
STATUSCODE	Specifies the WIP status code.
FILETYPE	Set this to XML to create an XML export file. Set to CMBNA to create a combined NA/POL file. The default is to create a V2 export file.
XMLALLOBJECTS	See XMLALLFIELDS. If you set FILETYPE to XML, use this variable to control how much information is output. If you include this attachment variable, the system includes additional Documaker attributes, such as coordinates, in the output XML file.
XMLALLFIELDS	Include this attachment variable to include empty fields as well as fields with data in an extended XML file. Use this attachment variable instead of the XMLALLOBJECTS attachment variable. The latter results in overly large XML files.
DESC	(Optional) Specifies the WIP description.

Errors

This rule can return these messages:

Message	Description
DPR0001	Cannot locate the variable #VARIABLE,# in the attachment list.
DPR0017	Cannot locate the DSI variable #VARIABLE,#.
DPR0022	Cannot add the variable #VARIABLE,# to the attachment list.
DPR0035	Cannot open the export file #FILENAME,#. Error reported by OS #ERRORNO,#
DPR0036	The DSI variable #VARIABLE,# does not contain a valid form set (FAP file).

Returns

Success or failure

See also

DPRPrint on page 211

DPRUnloadXMLFormset

Use this rule to unload different versions of an XML form set based on different options passed in as input attachment variables. The form set unloaded is a sub-form set based on GROUP1 and GROUP2 input attachment variables.

Syntax

```
long _DSIAPI DPRUnloadXMLFormset ( DSIHANDLE hdsi,
                                   char * pszParms,
                                   unsigned long  ulMsg,
                                   unsigned long  ulOptions )
```

Parameters

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

Attachment variables

Variable	Description
GROUP1	The Key1 for a form set.
GROUP2	The Key2 for a form set.
PRINTPATH	(Optional) Specifies the print path location of the XML form set.
XMLIMAGEOPTIONS	(Optional) Unloads all image options for a form set.
XMLALLFIELDS	(Optional) Unloads all empty field information for a form set.
XMLALLOBJECTS	(Optional) Unloads all objects for a form set.

Attachment outputs

Variable	Description
XMLFORMSET	Contains the full path and file name of the unloaded XML form set.
RESULTS	Success or failure

NOTE: You must pass a CONFIG attachment variable to DPRSetConfig rule in the same request type so it can find the form set it needs to unload.

See also DPRLoadXMLFormset on page 189

DPRUnlockWip

Use this rule to unlock a WIP record after it has been edited so other users can make changes to the record.

Syntax

```
long _DSIAPI DPRUnlockWip ( 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

Attachment variables

This rule expects these attachment variables:

Variable	Description
USERID	The user ID you want to unlock.
RECNUM or UNIQUE_ID	Lets the rule find the correct WIP record.

See also

DPRAddWipRecord on page 71

DPRApproveWipRecords on page 73

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDelMultiWipRecords on page 109

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRGetOneWipRecord on page 143

DPRIni2XML on page 159

DPRLockWip on page 191

DPRModifyWipData on page 205

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRUpdateFromMRL

Use this rule to get group and form lists from IDS. You can use this rule to get the...

- Group list
- Form list for a specific group or groups
- Forms with image and field information
- HTML representation of FAP images

This rule locates the form set in the DSI variable DPRFORMSET. If there is no form set, this rule creates the form set with group information only. If the form set has groups but no forms, the rule updates it with a list of forms for the groups.

If the form set has forms, DPRUpdateFromMRL updates it with image and required field information.

You can use the DPRUpdateFromMRL rule with these rules on the same request type:

- DPRLoadXMLAttachment
- DPRLoadedXML2Formset
- DPRSortFormsetForms
- DPRFilterFormsetForms

Syntax

```
long _DSIAPI DPRUpdateFromMRL ( 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

Attachment variables

Variable	Description
RECORDS	The attachment variable RECORDS is created with the value of the total number of returned groups or the total number of forms in the list of groups provided as input. MAXRECORDS and PAGE values do not affect this number, however, searching for forms does affect it.
XMLFORMID	If the XMLFORMID attachment variable is checked, the unique form ID is generated for each form. When this variable is set to No the form ID is not generated. It is only applicable when forms are returned.

Variable	Description
MAXRECORDS	If the attachment variable MAXRECORDS is checked, the number of forms returned is limited to its value. If this variable is missing, all forms will be returned. When getting the group list this variable is ignored as the number of groups is usually small and can be returned at once.
PAGE	When the attachment variable PAGE is checked, the form starting at the position of MAXRECORDS times PAGE number is the first form to be returned. This does not apply to group list.
STARTRECORD	Enter the record you with which you want the rule to start.

Using MAXRECORDS and PAGE lets the application implement paging in case the total number of forms is large. For example, if the passed in values are PAGE=20 and MAXRECORDS=10 the forms 191-200 will be returned.

The form set is updated from MRL on DSI_MSGRUNF and the forms are removed from it based on PAGE and MAXRECORDS values on the DSI_MSGRUNR message.

See also

DPRLoadXMLAttachment on page 188

DPRLoadedXML2Formset on page 184

DPRSendFormsetXML on page 251

DPRFilterFormsetForms on page 122

DPRSortFormsetForms on page 260

DPRGetFormList on page 137

DPRGetHTMLForms on page 141

DPRUpdateFormsetFields

Use this rule to update form set fields in memory with values specified in attachment variables. Attachment variable names must start with FORMSETUPDATEFIELD and are in the following format:

```
\FORM\IMAGE\FIELD\FieldData
```

The form and image names are optional but the format of the value must be the same. Here is an example:

```
\\\FIELD\FieldData
```

If no attachment variables named FORMSETUPDATEFIELD are found, no error is produced and there is no modification to the form set.

All matching fields will be updated in case there is more then one with the same name. Updating fields that are embedded into text areas will force the reformatting and might create more pages.

The form set in memory is located in the DSI variable DPRFORMSET. If the particular request type uses a different DSI variable to store the form set, the rule parameter in the INI file should provide the name of the DSI variable.

Syntax

```
long _DSIAPI DPRUpdateFormsetFields ( 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
ULONG ulMsg	DSI_MSG???? message, such as DSI_MSGRUNF
ULONG ulOptions	options

Returns

Success or failure

Errors

This rule can return these messages:

Error	Description
DPR0017	Cannot locate DSI variable #VARIABLE,#.
DPR0036	The DSI variable #VARIABLE,# does not contain a valid form set

DPRUpdateFormsetFromXML

Use this rule to update forms in the form set based on an XML document in memory. This rule updates form set data during form selection when using iPPS or iDocumaker and the WIP Edit plug-in. You can update all fields or only global scope fields.

NOTE: This rule is also used by iPPS and iDocumaker to do form selection when you are using the WIP Edit plug-in.

With Shared Objects version 11.2 and higher, you can use this rule with HTML entry. When you use this rule with HTML entry, it acts like the DPRLoadImportFile rule.

Syntax

```
long _DSIAPI DPRAUpdateFormsetFromXML ( 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

The rule expects the XML to be loaded into the DSI variable DPRXMLFORMSET by DPRLoadXMLAttachment rule. It also expects the form set (NA and POL files) to be loaded into the DSI variable DPRFORMSET by a rule such as the DPRGetWipFormset rule.

This rule is executed on DSI_MSGRUNF.

This rule only allows you to add information. You cannot use it to remove information or change the order of forms or modify image and field information.

Attachment variables

Variable	Description
DPRSETALLFIELDS	To update all fields, set this variable to Yes. If set to Yes, the DPRSETGLOBALFIELDS value passed on the same request is ignored and assumed to be Yes as well. Keep in mind the DPRSETALLFIELDS value updates regular variable fields but not multi-line variable fields. Using this variable helps in situations where processing outside the Documaker environment provides additional field data and you must apply this additional data to the document. For example, if you have a rating engine evaluate a transaction and you now need to add the rating information to the transaction.

DPRSETGLOBALFIELDS	To update global scope fields, the XML file sent to IDS should provide the values for these fields and should also set the DPRSETGLOBALFIELDS attachment variable to Yes.
DPRIFORMSPROTOCOL	This attachment variable determines 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 and iDocumaker provide the DPRIFORMSPROTOCOL value.

NOTE: This is relevant only when you are using the WIP Edit plug-in. These attachment variables affect only DPRUpdateFormsetFromXML rule.

See also

DPRGetWipFormset on page 153

DPRLoadXMLAttachment on page 188

DPRLoadImportFile on page 186

DPRUpdateWipRecords

Use this rule to update multiple WIP records. It retrieves a record each time based on the user's selection, and replaces one or more fields with a user-specified value. It then updates the record.

This rule accepts the minimum required fields, such as UniqID and Status Code, as input attachments when retrieving records. Other fields are optional. The Status Code field can also be optional if goChange is set to Yes.

NOTE: Normally, goChange is left blank and defaults to No. Only when the provided status code and status code from record file differ—such as when the status code is changed by another user while the status code remains unchanged on your local machine — should it be set to Yes. This makes sure that during the next submission, the new status code is used to update the record.

You must include the UniqID field to retrieve the record. You can also include other fields as input attachments to update the original fields in the record. Here is an example for Print Preview to update status code:

```
WIPS=1&WIPS1.StatusCode=W&WIPS1.RecNum=5&NEWWIP1.StatusCode=RJ
```

In this case *WIPS1.RecNum* is required and *WIPS1.StatusCode=W* is recommended. *NEWWIP1.StatusCode=RJ* is the only field that provides a new status code to update the original one.

Syntax

```
long _DSIAPI DPRUpdateWipRecords ( DSIHANDLE hdsi,
                                   char * pszParms,
                                   unsigned long  ulMsg,
                                   unsigned long  ulOptions )
```

Parameters

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

This rule expects these attachment variables:

Variable	Description
WIPS	The number of records to be updated.
WIPSX.FieldName	The value of the field to be updated in the original record.

Variable	Description
NEWWIPX.FieldName	The new value of the field to update the original one. Where the affix X (WIPSX.FieldName) is the number of WIP records to be updated, counting from 1 to WIPS; FieldName is the field defined in the WIP DFD file. All fields are expected even though some may be empty. In absence of the DFD file, FieldName takes default field names, such as Key1, Key2, KeyID, RecType, and so on.
GOCHANGE	This input attachment variable can be set to Yes or No. The default value is No. You can use this attachment variable in situations where the STATUS CODE of the selected record may have been changed by another user. In the retrieved record list, the STATUS CODE may still be the old value. When you try to update the STATUS CODE, the system will not do it since it has been updated. After you realize it, you can update STATUS CODE by setting GOCHANGE to Yes. This input attachment variable is normally used with Print Preview. For more information, refer to the Internet Document Server Guide.
ACTION	This input attachment variable has these values: UPDATE, ADD, or DELETE. The default is UPDATE. This lets you create one piece of code that can, for instance, update, add, and delete records. When you set ACTION to UPDATE, you have to input both WIPSX.fieldnames set and NEWWIPX.fieldnames set. When you set ACTION to ADD or DELETE, you only have to input WIPSX.fieldnames set. If you have multiple records to update, add, or delete, specify WIPS=number of records, and WIPS1.fieldnames, WIPS2.fieldnames, and so on, along with NEWWIP1.fieldnames, NEWWIP2.fieldnames, and so on.

Request types

ReqType = WST

The requested type is required in the DOCSERV.INI file. Here is an example:

```
< ReqType:WST >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRUpdateWipRecords
```

INI options

Use these INI options in the WIPData control group with this rule:

Option	Description
File	Specifies the name of the WIP file.
Path	Specifies the path to the WIP file.
MaxWIPRecords	Specifies the maximum number of records to read into the processQ. This prevents it from slowing down because of a large volume of records.

Here is an example:

```
< WIPData >
    File     = WIP
    Path     = mstrres\sampco\wip\
```

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0001	Cannot locate variable #VARIBALE,# in the attachment list.
DPR0006	The virtual memory management API #AOINAME,# failed.
DPR0007	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.
DPR0022	Cannot add variable #BARIABLE,# to the attachment list.
DPR0025	Cannot add variable #VARIABLE,# to the attachment record #RECORD,#
DPR0039	The call by #LOCATION,# to API #APINAME,# failed.

NOTE: This rule can update any field in a record, but it is typically used to change the status code.

Remember that WIPS1.fieldnames set is for the original fields in the selected record, while NEWWIP1.fieldnames set is for the new fields. In the new fields, you can specify the new values you want to replace the old values.

This rule can add or delete records. To add or delete records, it expects the attachment variable ACTION with the value UPDATE, ADD or DELETE. The default is UPDATE. This rule is tested only for updating the status code.

See also DPRApproveWipRecords on page 73

DPRCheckWipRecords on page 88

DPRGetWipList on page 150

DPRGetWipFormset on page 153

DPRGetWipRecipients on page 156

DPRSearchWip on page 246

DPRWip2Dpw

Use this rule to create a DPW file from WIP. The DPW file will contain the following:

- WIP index - in XML format (created by the DPRWipIndex2XML rule)
- Menu file - path defined by INI option
- NA file - from WIP
- POL file - from WIP
- FAP files - all FAP files within the form set
- LOG files - all logos used in the form set.

Syntax

```
long _DSIAPI DPRWip2Dpw ( 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

This rule creates the final section of the DPW file. Use this rule with these other rules:

- DPRWipIndex2XML - to get the XML portion of the DPW file.
- DPRGetWipFormset - to get the form set handle needed to get the FAP files and logos in the DPW file
- ATCSendFile - to send the DPW file back to the client.

Errors

This rule can return these messages:

Message	Description
DPR0039	The API failed.
DPR0001	Failed to get the variable attachment.
DPR0056	Could not get the file name.

See also

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRGetOneWipRecord on page 143

DPRGetWipFormset on page 153

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRWipIndex2XML on page 296

DPRWipBatchPrint

Use this rule to print multiple transactions from WIP. This rule is used with iDocumaker 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 variables

This rule expects these input attachment variables:

Variable	Description
PrtType	(Optional) This specifies the print type, such as PCL, PCL6 (PXL), or PostScript. If omitted, 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 the PrtDevice variable is 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 DPRAddLogo on page 65 for information on 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.

Variable	Description

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`

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.

Option	Description
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

This rule can return these messages:

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

See also

DPRAddLogo on page 65

DPRWipIndex2XML

Use this rule to create the XML portion of DPW file. Other rules can get the variables through WIPXMLVAR. Be sure to set up the menu file as shown here:

```
< WIP2DPW >
    Menu = wipedit.res
```

Syntax

```
long _DSIAPI DPRWipIndex2XML ( 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

Attachment variables

This rule expects this input attachment variable:

Variable	Description
RECNUM or UNIQUE_ID	Lets the rule find the correct WIP record.

The WIP record is broken into attachment variables.

Errors

This rule can return these messages:

Message	Description
DPR0039	The API failed.
DPR0001	The CONFIG variable was not found.
DPR0026	Could not open DWP file for creation.
DPR0010	Could not create WIPXMLVAR.
DPR0006	VMMMalloc failed.

Attachment outputs

This rule creates these DSI variables:

Variable	Description
WIPDATAPTR	An internal variable that contains the WIP buffer.
WIPXMLVAR	The XML version of the WIP record.

The rule writes out the WIP index portion of the DPW file on run-reverse.

See also

DPRAddWipRecord on page 71

DPRApproveWipRecords on page 73

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDelMultiWipRecords on page 109

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRGetOneWipRecord on page 143

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRModifyWipData on page 205

DPRWip2Dpw on page 290

DPRWipTableParms on page 298

DPRWipTableParms

Use this rule to update the parameters for the WIP table shown on the WIP List page. This rule is expected for Print Preview in all required REQTYPEs.

Syntax

```
    long _DSIAPI DPRWipTableParms ( DSIHANDLE hdsi,
                                     char * pszParms,
                                     unsigned long  ulMsg,
                                     unsigned long  ulOptions )
```

Parameters

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

Attachment variables

This rule expects this input attachment variable:

Variable	Description
CONFIG	The user configuration.

Attachment outputs

This rule expects these output attachments:

Variable	Description
Fields	Specifies the WIP fields as defined in the WIP.DFD file.
Table	Specifies the WIP fields in the WIP table.
WIPKey	Specifies the WIP fields to fill in the WIP table.
OptKey	Specifies the action keys (status code) for the SELECTION options.
AppTxt	Specifies the application options.
ShwTxt	Specifies the show options.
EntryTbl	Specifies the entry table for searching table.
EntryKey	Specifies the entry keys for search records.

INI options

This rule reads the WIP table parameters from the PRTView_WIPTable control group and add the text strings to output queue. If the control group is missing, the rule uses the default WIP parameters.

You use these options in the PrtView_WIPTable control group to define the output attachments:

Option	Description
Fields	Specifies the WIP fields as defined in the WIP.DFD file.
Table	Specifies the WIP fields in the WIP table.
WIPKey	Specifies the WIP fields to fill in the WIP table.
OptKey	Specifies the action keys (status code) for the SELECTION options.
AppTxt	Specifies the application options.
ShwTxt	Specifies the show options.
EntryTbl	Specifies the entry table for searching table.
EntryKey	Specifies the entry keys for search records.

Here is an example:

```
< PrtView_WIPTable >
;table
    Fields = KEY1,KEY2,KEYID,RECTYPE,CREATETIME,ORIGUSER,CURRUSER,
        MODIFYTIME,FORMSETID,TRANCODE,STATUSCODE,FROMUSER,FROMTIME,
        TOUSER,TOTIME,DESC,INUSE,ARCKEY,APPDATA,RECNUM
    Table = KEY1,KEY2,KEYID,RT,CT,OU,CU,MT,ID,TR,ST,DESC,RECNUM
    WIPKey = KEY1,KEY2,KEYID,RECTYPE,CREATETIME,ORIGUSER,CURRUSER,
        MODIFYTIME,FORMSETID,TRANCODE,STATUSCODE,DESC,RECNUM
;dropdown
    OptKey = AP,AR
    AppTxt = Approve,Archive only
    ShwTxt = Approved,Archived
;entry table
    EntryTbl = Key 1,Key 2,Key ID,Record Type,Formset ID,Tran
        Code,Status Code
    EntryKey = KEY1,KEY2,KEYID,RECTYPE,FORMSETID,TRANCODE,STATUSCODE
```

If you omit this control group, the default arrays are used. Be sure to include all INI options shown here.

Returns Success or failure

Errors This rule can return these messages:

Message	Description
DPR0022	Cannot add variable #VARIABLE,# to the attachment list.

See also DPRAddWipRecord on page 71

DPRApproveWipRecords on page 73

DPRAssignWipRecord on page 78

DPRDeleteWipRecord on page 105

DPRDelMultiWipRecords on page 109

DPRDpw2Wip on page 112

DPRFile2Dpw on page 120

DPRGetOneWipRecord on page 143

DPRIni2XML on page 159

DPRLockWip on page 191

DPRUnlockWip on page 281

DPRModifyWipData on page 205

DPRWip2Dpw on page 290

DPRWipIndex2XML on page 296

DPRXMLDiff

Use this rule after the DPRCompareXMLFiles rule to unload the XML file that rule created.

Syntax

```
long _DSIAPI DPRXMLDiff ( 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

When this rule is called, it first locates the DSI variable *DPRXMLFORMSET* to retrieve the XML document handle. If the XML document handle does not exist, the rule returns without output.

To unload the XML file, it will locate the attachment variable *PRINTFILE* to get a user defined file name. If the file name does not exist, a unique file name will be generated for the unloading. If the defined file name includes a path, use it, otherwise it will locate the attachment variable *PRINTPATH* for the user-defined path.

RPDCheckAttachments

Use this rule to check the required input attachment variables and INI options before starting the GenData program.

Syntax

```
_DSIEXPORT DWORD _DSIAPI RPDCheckAttachments (DSIHANDLE hdsi,
           char * pszParms,
           ULONG ulMsg,
           ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

This rule runs before the RPDCheckRPRun rule. Using this rule, ReqType becomes:

```
< ReqType:RPD >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = irlw32->IRLCopyAttachment
    function = dprw32->DPRSetConfig
    function = RPDW32->RPDCheckAttachments
    function = RPDW32->RPDCheckRPRun
    function = RPDW32->RPDCreateJob
    function = RPDW32->RPDProcessJob
```

The expected attachment variables are checked only if they are in the RPDAttachments control group. Here is an example:

```
< RPDAttachments >
    Variable = ReqType
    Variable = Config
    Variable = PrintBatches
    Variable = ExtrFile
```

If the ExtrFile option is required, the rule checks to see if it exists. Keep in mind the ExtrFile option includes a full path. If you omit the path, the system uses the path specified in the ExtrPath option as the default path.

This rule also checks these options in the RPDRunRPcontrol group:

```
< RPDRunRP >
    Executable  = d:\RP\Mstrres\gendaw32.exe
    Directory   = d:\RP\Mstrres\rpex1\
    UserINI     =..\..\fsiuser
```

If the UserINI option does not include a drive letter, the system will look at the Directory option to find the path, so the full UserINI name becomes:

```
d:\ProgIDS\RP\Mstrres\Validate\W32exe\..\..\fsiuser
```

In other cases, you can set the UserINI option, as shown here:

```
Directory = d:\ProgIDS\RP\Mstrres\Validate\W32exe\..\..\fsiuser
UserINI = ..\..\fsiuser
```

So the full UserINI name becomes:

```
d:\ProgIDS\RP\Mstrres\Validate\W32exe\..\..\fsiuser
```

This rule also makes sure the USERINI.INI file exists. For UNIX, if the first byte is "/", the system looks at the UserINI option for the full path, for example:

```
UserINI=/ProgIDS/RP/Mstrres/Deflib
```

Otherwise, the system uses the path specified in the Directory option. Keep in mind that if the UserINI option is omitted, the FSIUSER.INI file is used as the default USERINI.INI file.

INI options

You can use these INI options:

```
< RPDAttachments >
    Variable    = ReqType
    Variable    = Config
    Variable    = PrintBatches
    Variable    = ExtrFile
< IDSServer >
    ExtrPath    = d:\fap\mstrres\rpex1\extract\
< RPDRunRP >
    Executable  = d:\rel101\rps100\shipw32\gendaw32.exe
    Directory   = d:\fap\mstrres\rpex1\
    UserINI     = fsiuser
< Debug >
    RPDCheckAttachments =
```

Option	Description
RPDAttachments control group	
Variable	The name of the variable.
IDSServer control group	
ExtrPath	The default path for the ExtrFile option.
RPDRunRP control group	
Executable	The name and path of the program you want to execute, such as d:\rpsetup\gendaw32.exe.
Directory	The path to the master resource library, where you want to run Documaker.
UserINI	(Optional) The name and path of the INI file you want to use. The default is the FSIUSER.INI located in the directory specified by the Directory option.
Debug control group	
RPDCheckAttachments	Enter Yes to append errors to the ErrFile.

Returns Success or failure

Errors This rule can return these messages:

Message	Description
RPD0001	Can not locate variable #VARIABLE,# in the attachment list at #LOCATION,#.
RPD0004	Can not add variable #VARIABLE,# to attachment at #LOCATION,#.
RPD0007	File #FILENAME,# does not exists. Failed to #LOCATION,#.
RPD0009	The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.

See also RPDCheckRPRun on page 305

RPDCreateJob on page 308

RPDDeleteFiles on page 313

RPDProcessJob on page 315

RPDRunRP on page 318

RPDSetPDFAttachmentVariables on page 323

RPDStopRPRun on page 325

RPDCheckRPRun

Use this rule to make sure Documaker is running. If Documaker is not running, this rule starts it.

Syntax

```
_DSIEXPORT DWORD _DSIAPI RPDCheckRPRun (DSIHANDLE hdsi,
           char * pszParms,
           ULONG ulMsg,
           ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

To determine if Documaker is running, the rule looks at the CONFIG value. If the CONFIG value is not the same as it was in the previous run, this rule stops and then restarts Documaker.

On the RUNF message, this rule looks to see if a Documaker process exists and starts one if needed. On the RUNR message, this rule stops the Documaker process if there was an error.

On DSI_MSGRUNF, this rule first checks to see if Documaker is running by detecting the *gendata* semaphore created by RULServerBaseProc rule. If the semaphore does not exist, Documaker is not running. This rule then starts Documaker and creates a semaphore called *rpdrunrp*.

This lets Documaker check the status of the IDS by detecting the existence of the semaphore. It also lets Documaker terminate normally in case IDS stops.

To handle situations where you have multiple master resource libraries (MRLs), the rule checks the CONFIG value for every job process to see if a new MRL is requested. If the CONFIG value changes, the rule stops the current Documaker process and starts another one which uses the new MRL.

On DSI_MSGRUNR, this rule terminates Documaker if errors occur.

Attachment variables

Variable	Description
CONFIG	The configuration for the master resource library (MRL). See also the DPRSetConfig rule and the setup with multiple master resource directories.

Attachment outputs

Variable	Description
RPDRunProcess	This value is the process ID for the Documaker process.

Variable	Description
RPDSemaphoreName	The semaphore name from the RPDSemaphore INI option.
GENSemaphoreName	The semaphore name from the GENSemaphore INI option.
RPDRunSemaphore	Stores the RPDSemaphore handle.
RPDJobLogName	The name of the job log file name to use.
RPDJobTicketName	The name of the job ticket file name to use.

INI options

You can use these INI options:

```
< RPDRunRP >
    Executable =
    Directory =
    UserINI =
< IDSServer >
    GENSemaphoreName =
    RPDSemaphoreName =
```

Option	Description
RPDRunRP control group	
Executable	The name and path of the program you want to execute, such as d:\rpsetup\gendaw32.exe.
Directory	The path to the master resource library, where you want to run Documaker.
UserINI	(Optional) The name and path of the INI file you want to use. The default is the FSIUSER.INI located in the directory specified by the Directory option.
IDSServer control group	
GENSemaphoreName	The name of the semaphore. The default is *gendata*.
RPDSemaphoreName	The name of the semaphore. The default is *rpdrunrp*.
MaxConfigAllowed	A number of maximum configurations allowed for multiple processes. If a configuration is not found in a list in memory, start a new process and save the configuration in the list.
Debug control group	
RPDCheckRPRun	Enter Yes if you want errors appended to the ErrFile and the LogTrace file to record the trace.
RPDErrFile	Specify a name for the RPDErrfile. Include the full path

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
RPD0001	Cannot locate variable #VARIABLE,# in the attachment list at #LOCATION,#.
RPD0004	Cannot add variable #VARIABLE,# to attachment at #LOCATION,#.
RPD0008	The call by #LOCATION,# to API #APINAME,# failed.
RPD0009	The INI option #INIOPTION,# can not be located in the group #INIGROUP,#.
RPD0010	Cannot create DSI variable #VARIABLE,#. #LOCATION,# failed.
RPD0013	Cannot open the RPD error file #VARIABLE,# at #LOCATION,#.

See also

RPDCheckAttachments on page 302

RPDCreateJob on page 308

RPDDeleteFiles on page 313

RPDProcessJob on page 315

RPDRunRP on page 318

RPDSetPDFAttachmentVariables on page 323

RPDStopRPRun on page 325

RPDCreateJob

Use this rule to find the attachment variables for each of the values in the job ticket and add them to the XML tree. The XML tree is added to the RPDJOBTICKET DSI variable so the next rule can use it.

Syntax

```
_DSIEXPORT DWORD _DSIAPI RPDCreateJob (DSIHANDLE hdsi,
           char * pszParms,
           ULONG ulMsg,
           ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

On DSI_MSGRUNF, this rule creates the XML document for the job ticket that triggers the job processing. You should direct your results to designated directories and use unique file names, especially if you want to support multiple MRL setups, multiple RP processes, or multiple job processes.

You can change INI options via attachment variables. These changes are added onto the XML tree so Documaker can update the INI options in memory.

On DSI_MSGRUNR, this rule processes the XML document of the job log, and all values of the XML tree are added to the output attachment.

NOTE: See also the ServerFilterFromRecipient rule in the Rules Reference.

Attachment variables

You can use these input attachment variables:

Variable	Description
ExtrFile	Extract file name and path. This is a required input file.
MsgFile	(Optional) Message file name and path. If you omit the path, the PrintPath attachment variable is used. If the PrintPath was omitted, the system uses the PrintPath defined in the IDSServer control group. If the file name is omitted, the system creates a 46-byte unique file name.
ErrFile	(Optional) Error file name and path. If you omit the path, the PrintPath attachment variable is used. If the PrintPath was omitted, the system uses the PrintPath defined in the IDSServer control group. If the file name is omitted, the system creates a 46-byte unique file name.

Variable	Description
LogFile	(Optional) Log file name and path. If you omit the path, the PrintPath attachment variable is used. If the PrintPath was omitted, the system uses the PrintPath defined in the IDSServer control group. If the file name is omitted, the system creates a 46-byte unique file name.
DBLogFile	(Optional) DB log file name and path. If you omit the path, the PrintPath attachment variable is used. If the PrintPath was omitted, the system uses the PrintPath defined in the IDSServer control group. If the file name is omitted, the system creates a 46-byte unique file name.
NAFile	(Optional) NA file name and path. If you omit the path, the PrintPath attachment variable is used. If the PrintPath was omitted, the system uses the PrintPath defined in the IDSServer control group. If the file name is omitted, the system creates a 46-byte unique file name.
POLFile	(Optional) POL file name and path. If you omit the path, the PrintPath attachment variable is used. If the PrintPath was omitted, the system uses the PrintPath defined in the IDSServer control group. If the file name is omitted, the system creates a 46-byte unique file name.
NewTrn	(Optional) NewTrn file name and path. If you omit the path, the PrintPath attachment variable is used. If the PrintPath was omitted, the system uses the PrintPath defined in the IDSServer control group. If the file name is omitted, the system creates a 46-byte unique file name.
PrintBatchPath	The default path for print batches.

PrintBatches	The number of batches to print. Your entry cannot exceed the number of printers listed in the PrinterInfo control group in the FSISYS.INI file.
	If you do not set this attachment variable. Documaker Bridge looks in the Documaker INI files and determines the correct value.
	The system determines the value based on the number of Printer options in the PrinterInfo control group of your Documaker (GenData) INI files (FSIUSER.INI and FSISYS.INI):
	`< PrinterInfo >`
	`    Printer =`

PrintBatchesX	The name of a print batch, where *X* denotes the number of the print batch, continuing from one to *PrintBatches*. If omitted, the system creates a 46-byte unique name for the print batch. A print batch can have a full path. If it does not have a path, PrintPath is used. If PrintPath is omitted, the system uses the path specified in the PrintPath option in the Data control group.
BatchFiles	The number of batch files. If you enter zero or omit this option, no batch file information is updated. Your entry should not exceed the number of batch files listed in the Print_Batches control group in the FSISYS.INI file.

Variable	Description
BatchFilesX	The name of the batch file. *X* denotes the number of the batch file, counting from one to the maximum. If you omit this option, the system creates a 46-byte unique name for the batch file. You can include a full path. If you omit the path, the system uses the PrintPath. If the PrintPath is omitted, the system uses the path specified in the PrintPath option in the IDSServer control group.
INIOptions	The number of other INI options to update.
INIOptionsX.Group	The INI group name you want to update.
INIOptionsX.Option	The INI option name you want to update.
INIOptionsX.Value	The value of the INI option you want to update. *X* indicates the number of INI options, counting from one to the maximum.

Output DSI variables

Variable	Description
RPDJOBTICKET	Job ticket variable. Its value is a XML document handle for the job ticket.

Input DSI variables

Variable	Description
RPDJOBLOG	Job log variable. Returns an XML document handle for the job log.

Attachment outputs

Variable	Description
ExtrFile	Extract file name and path.
MsgFile	Message file name and path.
ErrFile	Error file name and path.
LogFile	Log file name and path.
DBLogFile	DB log file name and path.
NAFile	NA file name and path.
POLFile	POL file name and path.
NewTrn	NewTrn file name and path.
PrinterX	Name and path of print batches. *X* denotes the number of the print batches from one to the maximum.
BatchX	The name and path of the batch files. *X* denotes the number of batch files, from one to the maximum.
Results	Success or an error code from the IDS rules.

Variable	Description
RPResults	An error code from Documaker: 0=Success, 4=Warning, 8 or 16=Failure.

Note that the input attachments for PrintBatchX should be in the same order as those for PrinterX, as defined in the PrintInfo control group in the FSISYS.INI file. Also keep in mind that *PrinterX* and *BatchX* are option names you define in the PrintInfo and Print_Batches control groups.

INI options

You can use these INI options:

```
< IDSServer >
    PrintPath          =
    PrintFileCacheTime =
    TextFileCacheTime  =
< Printer >
    PrtType            =
< RPDRunRP >
    BaseLocation       =
```

Option	Description
IDSServer control group	
PrintPath	Used as a default path for print batches and the rest of the output files.
PrintFileCacheTime	The length of time, in seconds, you want the system to store the print files. At expiration time, the system removes the print batch files. The default is 1800 (30 minutes). Note that only print files with the 46-byte unique name created by the system are cached.
TextFileCacheTime	The length of time, in seconds, you want the system to store the text files. At expiration time, the system removes the text files. The default is 1800 (30 minutes). Note that only text files with the 46-byte unique name created by the system are cached.
FileExt	The file extension you want to use if an output file specified by input attachment, such as ExtrFile, MsgFile, ErrFile, LogFile, DbLogFile, NaFile, PolFile, and PrtLog, does not have an extension. The default is *.dat*.
Printer control group	
PrtType	The type of print batch file. Your entry must be consistent with the control group defined in the FSISYS.INI file. For instance, if you set up a PrtType:PDF control group there, enter PDF here.
RPDRunRP control group	
BaseLocation	The URL to the output data directory. Your entry must be consistent with the PrintPath or other defined data path.
Debug control group	
RPDCreateJob	Enter Yes if you want errors appended to the ErrFile and the LogTrace file to record the trace.

Returns Success or failure

Errors This rule can return these messages:

Message	Description
RPD0002	Cannot create #TAGNAME,# at #LOCATION,#.
RPD0003	Cannot create DSI variable #VARIABLE,# at #LOCATION,#.
RPD0004	Cannot add variable #VARIABLE,# to attachment at #LOCATION,#.
RPD0005	Cannot locate DSI variable #VARIABLE,# at #LOCATION,#.
RPD0006	DSI variable #VARIABLE,# does not contain valid data. Failed to #LOCATION,#.
RPD0013	Cannot open the RPD error file #VARIABLE,# at #LOCATION,#.
RPD0020	A GenData program error that is passed back via the JobLog file.

See also RPDCheckAttachments on page 302

RPDCheckRPRun on page 305

RPDDeleteFiles on page 313

RPDProcessJob on page 315

RPDRunRP on page 318

RPDSetPDFAttachmentVariables on page 323

RPDStopRPRun on page 325

RPDDeleteFiles

Use this rule to delete files created by the RPDRunRP rule.

Syntax

```
_DSIEXPORT DWORD _DSIAPI RPDDeleteFiles (DSIHANDLE hdsi,
            char * pszParms,
            ULONG ulMsg,
            ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

This rule gets the values for the attachment variables RETCODE and RESULTS which were set in the RPDRunRP rule. It then gets the INI setting for the SaveOnErrors option.

If the RETCODE is greater than or equal to 8 and the SaveOnErrors option is True, the rule does not delete the files. If the SaveOnErrors option is set to True and RESULTS contains FAILURE, the rule does not delete the files. The rule then gets the INI setting for the KeepAll option. If this option is set to True, the rule does not delete the files.

If the files should be deleted, the rule deletes the extract, NA, POL, NEWTRN, TRN, DBLog, LOG, MSG, and print batch files. It also deletes the ERRFILE if the other files are deleted and RETCODE. And finally, the rule deletes the FSIUSER.INI file for the request.

Attachment variables

You can use these input attachment variables:

Variable	Description
RETCODE	Returned code from a prior RPD rule.
RESULTS	Success or failure from a prior RPD rule.

You have these output attachment variables:

Variable	Description
TEMPNAME	A 46-byte unique name for creating temporary output files, such as *.usr*, *.sys*, and so on.

INI options

You can use these INI options with this rule:

```
< RPRun >
   SaveOnErrors=
   KeepAll     =
```

Option	Description
SaveOnErrors	When the returned error code is greater than 4 or RESULTS returns a FAILURE and if SaveOnErrors is set to Yes, the Delete flag is set to No and temporary files are saved. Otherwise, temporary files are deleted. The temporary files include Extrfile, Nafile, PolFile, NewTrn, TrnFile, DbLogFile, LogFile and MsgFile. The default is No, with the Delete flag defaulting to TRUE.
KeepAll	If the Delete flag is Yes and KeepAll is Yes, the Delete flag is set to No to keep all temporary files.

Use the following option in the request INI to determine if the files should be saved on error (defaults to false if there is no entry in the INI file):

```
< RPRun >
    SaveOnErrors =
```

Use these settings in the request INI to determine if all files should be kept:

```
< RPRun >
    KeepAll =
```

To trigger this rule, add this line in the DOCSERV.INI file:

```
function = RPDW32->RPDDeleteFiles
```

Returns Success or failure

Errors You can get these errors:

- Cannot find the RETCODE attachment variable
- Cannot find the RESULTS attachment variable
- Cannot find the TEMPNAME attachment variable
- Cannot load the FSISYS file for the request
- Unable to locate RETCODE attachment variable.
- Unable to locate Results attachment variable.
- Unable to locate TEMPNAME attachment variable.
- Error in DeleteFiles

See also RPDCheckAttachments on page 302

RPDCheckRPRun on page 305

RPDCreateJob on page 308

RPDProcessJob on page 315

RPDRunRP on page 318

RPDSetPDFAttachmentVariables on page 323

RPDStopRPRun on page 325

RPDProcessJob

Use this rule to get the XML tree from the DSI variable RPDJobTicket and write it to a file written on the RUNF message. On the RUNR message, this rule waits for the job log file. The job log file is located in the same directory and is loaded as an XML file on the RUNR message.

Syntax

```
_DSIEXPORT DWORD _DSIAPI RPDProcessJob (DSIHANDLE hdsi,
           char * pszParms,
           ULONG ulMsg,
           ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

The IDS variable RPDJobLog is created with the XML job log. The RPDJobLog variable and the XML tree associated with it is destroyed in this rule on the TERM message.

You can set the maximum amount of time to wait using the MaxWaitTime option. On the RUNR message, this rule also removes the job log file from disk. You can also control the removal of the job log file with the RPDProcessJob INI option. This option is for debugging purposes only.

On DSI_MSGRUNF, this rule receives the XML document handle from the DSI variable RPDJobTicket, and writes the XML tree into the JOBTICKET.XML file specified in the Directory option.

On DSI_MSGRUNR, this rule waits until it receives the job log file (JOBLOG.XML), from Documaker. You specify how long the system should wait using the SleepingTime INI option. If the waiting time exceeds the limit, the rule stops Documaker.

The system locates a job log placed in the directory specified in the Directory INI option. The job log file is loaded into an XML document so the XML tree can be written out in attachments. Whether the JOBLOG.XML file should be removed, depends on your entry in the RPDProcessJob INI option.

Attachment variables

Variable	Description
RPDJobTicket	A job ticket variable. It returns the XML document handle for the job ticket.

Output files

File	Description
JOBTICKET.XML	A job ticket, which is a trigger for the RP process. It contains request information and information used to update INI options.

Attachment outputs

Variable	Description
RPDJobLog	The job log variable. Its value is an XML document handle for the job log.

INI options

You can use these INI options:

```
< RPDRunRP>
    Directory      =
< IDSServer >
    MaxWaitTime    =
    SleepingTime   =
    WaitForStart   =
< Debug >
    RPDProcessJob  =
```

Option	Description
RPDRunRP control group	
Directory	Enter the path where you want to load and unload the JOBTICKET.XML and JOBLOG.XML files.
IDSServer control group	
MaxWaitTime	Enter, in seconds, the maximum length of time you want IDS to wait for the JOBLOG.XML file. The default is 60 seconds.
SleepingTime	Enter the time, in milliseconds, to specify how often IDS should check for a job ticket. The default is 1000 (1 second).
WaitForStart	The length of time IDS should wait for Documaker to start before assuming RP is not running. The default is 10 seconds. Adjust this value if the Documaker requires more time to start. If Documaker does not start within the allotted time, this rule returns an error and stops processing.
Debug control group	
RPDProcessJob	Enter Yes if you want errors appended to the ErrFile, the LogTrace file to record the trace, and the JobLog file to be renamed and saved.

Returns

Success or failure

Errors

This rule can return these messages:

Message	Description
RPD0003	Cannot create the DSI variable #VARIABLE,# at #LOCATION,#.
RPD0004	Cannot add the variable #VARIABLE,# to attachment at #LOCATION,#.
RPD0005	Cannot locate the DSI variable #VARIABLE,# at #LOCATION,#.
RPD0006	The DSI variable #VARIABLE,# does not contain valid data. Failed to #LOCATION,#.
RPD0007	The file #FILENAME,# does not exist. Failed to #LOCATION,#.
RPD0008	The call by #LOCATION,# to API #APINAME,# failed.
RPD0009	The INI option #INIOPTION,# cannot be located in the group #INIGROUP,#.
RPD0011	Records the unexpected termination of the GenData program.
RPD0013	Cannot open the RPD error file #VARIABLE,# at #LOCATION,#.

See also

RPDCheckAttachments on page 302

RPDCheckRPRun on page 305

RPDCreateJob on page 308

RPDDeleteFiles on page 313

RPDRunRP on page 318

RPDSetPDFAttachmentVariables on page 323

RPDStopRPRun on page 325

RPDRunRP

Use this rule to run Documaker's Rules Processor. It will either run the GenTrn, GenData, and GenPrint program, depending on how you set the SingleStepGenData INI option.

Syntax

```
_DSIEXPORT DWORD  _DSIAPI RPDRunRP (DSIHANDLE hdsi,
           char * pszParms,
           ULONG ulMsg,
           ULONG ulOptions)
```

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

To trigger this rule, set the following option in the DOCSERV.INI file:

```
function = RPDW32->RPDRunRP
```

Attachment variables

This rule expects these input attachment variables:

Variable	Description

CONFIG	This identifier specifies the identity of a specific application configuration. You must have a corresponding entry in the DAP.INI file. For example, if CONFIG=ABC you would need this entry in the DAP.INI file:
	`< CONFIG:ABC >`
	`    INIFile = ABC.INI`
	There must also be an ABC.INI file in the document server root directory. This ABC.INI file would contain application specific implementation details. See INI File Options below for more information.

EXTRACT	The full name and path of the extract file you want to process.

This rule creates these attachment variables:

Variable	Description
ERRORFILE	The URL of the error file created by the Rules Processor. This variable only exists if there is an error or a warning. If no error file has been created, this variable will be blank.
ERRORMSG	This variable only exists if there was an error and will contain the error message.
RESULTS	Success or failure

Variable	Description
RETCODE	The code returned from the Rules Processor. If the error occurred before the Rules Processor was called, the return code contains FAIL.
TEMPNAME	This variable contains the path and unique 4 character hex base name of the output files created by the Rules Processor. Here is an example: `/docserv/tempdata/04AD.`

This rule copies input attachment into the output attachment.

Returns

Success or failure

INI options

This rule uses these options in the RPRun control group:

Option	Description
BaseDirectory	This is the path for the output files created by this rule and Documaker. This is a required entry with no default. For example: `d:/docserv/tempdata/`
BaseLocation	This is the URL for the base directory. This is a required entry with no default. This gets used for the error file (if applicable). For example: `http://205.176.142.5./doc-data/` (where doc-data is the alias for d:\docserv\tempdata)
CacheTime	If an error file is created by the rule, it is cached for this length of time in minutes.The default is 60.
Debug	Set this to Yes to create a debug log which will be created in the doc server directory. This file will have a four character hex unique name with a *DBG* extension. The default is No.
SingleStepGenData	Set this to Yes to run GenData only. The default is No.
Startup	This is the path to run Documaker from. This is helpful when Documaker is a different release from the Doc Server. Default is the current directory. Include a slash at the end of the path, as shown here: `Startup = e:\dap\dll\`
TempRetries	This is the max number of times an attempt will be made to find a unique name and create a temporary file with that name and an *RPD* extension. (This temporary file is used as a place holder for that Unique Name). The default is 128.
UserINI	This is the name and path of the FSIUSER INI file to be used by Documaker. This is a required entry with no default. Here is an example: `d:/docserv/mstrres/rpd/ini/fsiuser.ini`
Debug	Enter Yes to generate a file containing trace information in the IDS directory. The file will have a 46-byte unique name with a *.dbg* extension. The default is No.

The INI file loaded by the Doc Server for the request that uses this rule (either DAP.INI or the INI listed in the Config control group for the request), must contain an RPRun control group as described above.

The FSIUSER.INI file listed in this INI, the FSISYS.INI file (listed in FSISYSINI control group of the FSIUSER.INI file), and the extract file (named in the Extract attachment variable) are copied to the directory listed in the BaseDirectory control group of the INI file. These copied files are renamed to use a four-character UniqName that was generated by the rule. The copied files will have these new extensions:

```
extract = .ext
fsiuser =-.usr
fsisys =. sys
```

The new FSIUSER.INI file is then updated to rename the output files listed in the Data control group.

NOTE: After the update, all entries from the FSISYS will be included in the FSIUSER. The FSISYSINI entry in FSIUSER is cleared to prevent it from being loaded in again by Documaker.

Each of the renamed output files will contain the BaseDirectory path followed by the unique name and the following extensions:

NOTE: This BaseDirectory followed by the Unique Name is placed in the TempName attachment variable.

```
Extrfile       =ext
Nafile         =na
PolFile        =pol
NewTrn         =.ntn
Trnfile        =trn (for SingleStepGendata, this is renamed to NUL)
NewTrn         =ntn
DBLogFile      =dbl
Errfile        =.err
MsgFile        =.msg
PrintBatches   =.bn (for each batch where n is sequential from 1)
PrinterInfo    =for each printer listed under printerinfo
port           =.xxx where xxx is the PrtType.
```

NOTE: Since every port is getting the same name, this only works with one printer. Likewise, it will only work for one batch.

The new FSIUSER is then passed in the command line to run Documaker. If you set the SingleStepGendata option to Yes, only the GenData program is executed. Otherwise, the GenTrn program is executed first. If it completes successfully, the GenData program is then executed. Finally if the GenData program completes successfully, GenPrint is executed.

If Documaker creates warnings or errors, the error file is converted to an HTML page and the URL is placed in the ERRORFILE attachment variable. The original error file is deleted.

If the process is successful, the RESULTS attachment variable contains SUCCESS. Otherwise, it contains FAILURE.

Errors

This rule can return these messages:

```
RDP0001 RPDRunRP failed. #ERRORMSG#
```

One of the following messages will be substituted:

Message	Description
BaseDirectory not specified in RPRUN section of INI	You need to specify the BaseDirectory in the RPRun control group.
BaseLocation not specified in RPRUN section of INI	You need to specify the BaseLocation in the RPRun control group.
UserINI not specified in RPRun section of INI	The UserINI file was not specified in RPRun section of the INI file
BaseDirectory does not exist	Appears if the BaseDirectory listed in the INI does not exist. (The actual BaseDirectory is displayed)
UserINI does not exist	Appears if the UserINI listed in the INI does not exist. (The actual UserINI file name is displayed)
Unable to locate 'Extract' Attachment variable	Appears when unable to locate extract attachment variable
Empty extract file specification in 'Extract' attachment variable	Appears when there is an empty extract file specification in the extract attachment variable
Extract file does not exist	Appears if the extract file specified in the attachment variable does not exist (the actual extract file name is displayed)
Unable to create temporary file	Appears when the attempt to create a temporary file with the new unique name was unsuccessful.
Call to CopyFiles() failed	Appears when there was an error copying the extract file or INI files to the BaseDirectory using the new unique name.
Call to RPDGetFsisys failed. Check The <Environment> FSISYSINI entry in FSIUSER	Appears when the FSISYS.INI file listed in the FSISYS.INI section of the FSIUSER.INI file does not exist.
GENTRAN step failed	Appears when the GenTrn program completes with a return code that is greater than four (4).

Message	Description
GENDATA step failed	Appears when the GenData program completes with a return code that is greater than four (4) Note: if you use the GenDataStopOn option to bypass errors, the GenData program may complete processing and produce all expected output files, however since there were errors, the return code from GenData is 8 and this error message appears.
GENPrint step failed	Appears when the GenPrint program completes with a return code that is greater than four (4).
Call to CopyFiles() failed.	Appears when the call to CopyFiles fails.
Call to RPDGetFsisys failed.	Check the FSISYSINI option in the Environment control group in your FSIUSER.INI file.
GENTRAN step failed.	The GenTrn processing step failed.
GENDATA step failed.	The GenData processing step failed.
GENPRINT step failed.	The GenPrint processing step failed.
Startup path does not exist.	The startup path is incorrect.
Unable to create temporary file.	The system cannot create a temporary file.
Unknown critical error occurred.	Appears when there was a failure for an unknown reason.

See also

RPDCheckAttachments on page 302

RPDCheckRPRun on page 305

RPDCreateJob on page 308

RPDDeleteFiles on page 313

RPDProcessJob on page 315

RPDSetPDFAttachmentVariables on page 323

RPDStopRPRun on page 325

RPDSetPDFAttachmentVariables

Use this rule to create PDF file name and URL attachment variables. This rule is run after the RPDRunRP rule.

Syntax

```
_DSIEXPORT DWORD _DSIAPI RPDSetPDFAttachmentVariables (DSIHANDLE
hdsi,
           char * pszParms,
           ULONG ulMsg,
           ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

This rule creates the FILE and URL attachment variables in the DSI_OUTPUTQUEUE for PDF files generated by the Rules Processor (RPDRunRP) which was run as a prior rule.

This rule uses the TEMPNAME attachment variable from DSI_INPUTQUEUE which is a path and unique file name generated for this request, such as c:/docserv/data/0be4.

The rule uses it to generate a wildcard search mask to search for PDF files. For each file found by the search, the rule adds an attachment record to DSI_OUTPUTQUEUE and adds to that attachment record a FILE value and a URL value such as:

```
http://10.2.10.23/doc-prog/data/79eb.pdf
```

Use this option in the request INI to specify the base location to use:

```
< RPRun >
    BaseLocation  =
```

Use these INI settings in the request INI to specify how long to cache the PDF file. When the time expires, the file is deleted the next time SAR is triggered. If there is no entry in the INI file, the cache time defaults to one hour.

```
< RPRun >
    CacheTime =
```

You can trigger this rule by adding the following line in the DOCSERV.INI file:

```
function = RPDW32->RPDSetPDFAttachmentVariables
```

Attachment variables

Variable	Description
TEMPNAME	A unique 4-character hex name used as an output file. In PDF format.

Attachment outputs

Variable	Description
PDFS	The number of PDF files.
PDFSX.FILE	The output PDF file.
PDFSX.URL	A complete URL.

Returns Success or failure

Errors These errors can appear:

- Cannot find the TEMPNAME attachment variable in the DSI_INPUT queue
- Cannot find the BaseLocation option in the RPRun control group in the request INI
- 'BaseLocation' is not specified in RPRun control group of the INI file.
- Unable to locate TEMPNAME attachment variable.
- Warning - invalid cache time in INI file - Defaulting to 1 hour.

See also RPDCheckAttachments on page 302

RPDCheckRPRun on page 305

RPDCreateJob on page 308

RPDDeleteFiles on page 313

RPDProcessJob on page 315

RPDRunRP on page 318

RPDStopRPRun on page 325

RPDStopRPRun

Use this rule to stop Documaker. To do so, you need to execute the request type STOP as described in the topic, Setting Up IDS. in the Internet Document Server Guide.

This rule is also used as an INIT/TERM rule and is registered on IDS under the ReqType:INI control group. You can use this rule to make sure that when IDS stops, Documaker also stops.

Syntax

```
_DSIEXPORT DWORD _DSIAPI RPDStopRPRun (DSIHANDLE hdsi,
           char * pszParms,
           ULONG ulMsg,
           ULONG ulOptions)
```

Parameters

Parameter	Description
DSIHANDLE hdsi	DSI instance handle
char * pszParms	Pointer to rule parameter string unsigned long
ulMsg	DSI_MSG, such as DSI_MSGRUNF unsigned long
ulOptions	options

This rule receives the current process ID from the DSI variable RPDRunProcess and then terminates Documaker.

Attachment outputs

```
< Debug >
  RPDStopRPRun =
```

Option	Description
RPDStopRPRun	Enter Yes to append errors to the ErrFile and have the LogTrace file record the trace.

Returns

Success or failure

See also

RPDCheckAttachments on page 302

RPDCheckRPRun on page 305

RPDCreateJob on page 308

RPDDeleteFiles on page 313

RPDProcessJob on page 315

RPDRunRP on page 318

RPDSetPDFAttachmentVariables on page 323

Index