Oracle® Documaker

Using the Documaker Bridge

with Docupresentation version 2.2

Part number: E17552-01
March 2011
THIRD PARTY SOFTWARE NOTICES

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

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

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

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

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

Copyright © 2009, Berkeley Software Distribution (BSD)

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

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

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

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

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

Copyright © 2009 MIT

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

Copyright (c) 1995-2005 Jean-loup Gailly and Mark Adler
This software is based in part on the work of the Independent JPEG Group (http://www.iwg.org/).

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

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

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

This product includes software developed by W3C.

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

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

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

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

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

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

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABILITY QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.
This product includes software developed by PJ Arends and distributed via Code Project Open License (http://www.codeproject.com).

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

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

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

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

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

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

Copyright (c) 1988-1997 Sam Leffler
Copyright (c) 1991-1997 Silicon Graphics, Inc.

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

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

This product includes software components distributed by the Cryptix Foundation.

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

Copyright © 1995-2005 The Cryptix Foundation Limited. All rights reserved.
This product includes software components distributed by Sun Microsystems.

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

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

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

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

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

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

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

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

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

This product includes software components distributed by Geir Landro.

Copyright © 2001-2003 Geir Landro (drop@destroydrop.com) JavaScript Tree - www.destroydrop.com/hjavascripts/tree/version 0.96
Contents

Chapter 1, Using the Documaker Bridge

2 Overview
4 Setting Up Your Resources
5 Updating INI Files
6 Database-specific Administration Tasks
7 Customizing the Bridge
7 Returning Error Messages in Attachment Variables
8 Switching to Another DBMS
10 Using Library Manager
10 Preserving Output Files
11 Automatically Printing Upon Completion
13 Using Image Origins with XML Import
13 Detecting the Import File Type
15 Setting Up the DAP.INI File
15 Dynamic Configuration - Using the Config Control Group
15 PDF File Creation Options
16 Configuring INI Files for Each Config Control Group
18 Setting Up the DOCSERV.XML File
18 Basic Options
18 Advanced Options
20 Setting Up the Client Configuration Files
20 Basic Options
21 Advanced Options
23 Verifying Users
27 Using Manually-Edited HTML Forms with Real-Time HTML Processing

Chapter 2, Documaker Bridge Rules

30 List of Rules
102 DPRGenerateSeedValue
104 DPRGetConfigList
106 DPRGetDFDInfo
111 DPRGetFormList
112 DPRGetFormsetRecips
114 DPRGetHTMLForms
115 DPRGetInitValue
116 DPRGetOneWipRecord
118 DPRGetRecipients
120 DPRGetUserList
123 DPRGetWipList
126 DPRGetWipFormset
129 DPRGetWipRecipients
132 DPRIni2XML
134 DPRInit
135 DPRInitLby
136 DPRLbyCopy
138 DPRLbyDelete
140 DPRLbyGet
143 DPRLbyLock
145 DPRLbyMKCol
146 DPRLbyOptions
147 DPRLbyPropFind
150 DPRLbyPropPatch
151 DPRLbyPut
153 DPRLbyUnlock
155 DPRLoadDPA
157 DPRLoadedXML2Formset
158 DPRLoadFAPImages
159 DPRLoadImportFile
161 DPRLoadXMLAttachment
162 DPRLoadXMLFormset
163 DPRLocateOneRecord
164 DPRLockWip
166 DPRLog
167 DPRLogin
169  DPRLoginUser
170  DPRMail
172  DPRMapRecipData
174  DPRModifyUser
178  DPRModifyWipData
180  DPRPatchLevel
181  DPRParseRecord
182  DPRPostDMProcess
184  DPRPrint

186  Adding Logos when using DPRPrint
187  Adding Transaction Index Information to the XML Export File
188  Generating File Names Based on Transaction Values

192  DPRPrintDpw
194  DPRPrintFormset
197  DPRProcessTemplates
199  DPRRenameVars
200  DPRRetFromUserDict
203  DPRRetrieveDPA
204  DPRRetrieveFormset
207  DPRRotateFormsetPages
208  DPRSearch
210  DPRSearchLDAP
217  DPRSearchWip
222  DPRSendFormsetXML
223  DPRSendMultiFiles
224  DPRSendVersion
225  DPRSet2ImageScope
226  DPRSetConfig
228  DPRSpellCheck
231  DPRSortFormsetForms
232  DPRTemporaryXMLFile
233  DPRTblLookUp
243  DPRTransform
249  DPRUnloadExportFile
251  DPRUnloadXMLFormset
252  DPRUnlockWip
Appendix A, Reading Print Stream Files

308  Getting AFP Resources
     308    AFP Fonts
     308    AFP Overlays
     309    AFP Page Segments
     309    Archived AFP Print Streams

310  Getting Metacode Resources
     310    Xerox JSL
     310    Metacode Fonts and Images
     311    Archived Metacode Print Streams
Chapter 1

Using the Documaker Bridge

Via the Internet Document Server (IDS), Docupresentation 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 these topics:

- Overview on page 2
- Setting Up Your Resources on page 4
- Customizing the Bridge on page 7
- Setting Up the DAP.INI File on page 15
- Setting Up the DOCSERV.XML File on page 18
- Setting Up the Client Configuration Files on page 20
- Verifying Users on page 23
- Using Manually-Edited HTML Forms with Real-Time HTML Processing on page 27

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 29.

NOTE: See the SDK Reference for information on rules you can use to control IDS.
Chapter 1
Using the Documaker Bridge

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 an 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.
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/doc022/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/doc022/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 `\rpex2\html` directory for RPEX2 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/doc022/mstrres/rpex1/html`
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.

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

```ini
< 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.

```ini
< Archival >
ArchiveMem = Yes
< ArcRet >
AppIdx = APPIDX
AppIdx = c:\docserv\mstrres\rpex1\deflib\Appidxdfd
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\nFormLib = mstrres\rpex1\forms\nLbyLib = mstrres\rpex1\forms\nFormFile =
< Control >
```
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:

<table>
<thead>
<tr>
<th>This statement</th>
<th>Is used to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2CMD (Windows) DB2 (UNIX)</td>
<td>Invoke the DB2 command line processor</td>
</tr>
<tr>
<td>DB2 CONNECT TO (ARCDB)</td>
<td>Connect to the DB2 Database (substitute your database name for ARCDB)</td>
</tr>
<tr>
<td>DB2 BIND DB2LIB.BND</td>
<td>Bind the DB2LIB Package to DB2</td>
</tr>
</tbody>
</table>
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 7
- Switching to Another DBMS on page 8
- Using Library Manager on page 10
- Preserving Output Files on page 10
- Automatically Printing Upon Completion on page 11

Keep in mind as you read through the following examples that XML standards, as defined by the W3C, require you to substitute text characters that are not in XML tags (for example, between `<entry>` and `</entry>` tags) as escape sequences. The characters that require substitution are listed in the following table. If you cut and paste an XML example from this or other Docupresentation documentation into an XML configuration file, you will have to manually make these substitutions.

<table>
<thead>
<tr>
<th>For this character</th>
<th>Use this escape sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; (less than)</td>
<td><code>&lt;</code> or <code>&amp;lt;</code></td>
</tr>
<tr>
<td>&gt; (greater than)</td>
<td><code>&gt;</code> or <code>&amp;gt;</code></td>
</tr>
<tr>
<td>&amp; (ampersand)</td>
<td><code>&amp;</code> or <code>&amp;amp;</code></td>
</tr>
<tr>
<td>' (apostrophe)</td>
<td><code>'</code> or <code>&amp;apos;</code></td>
</tr>
<tr>
<td>&quot; (quotation mark)</td>
<td><code>&quot;</code> or <code>&amp;quot;</code></td>
</tr>
</tbody>
</table>

**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 writes 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
```
Chapter 1
Using the Documaker Bridge

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

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

RPD00201
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>).
RPD00210.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)
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
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionTimer</td>
<td>Enter the number of seconds you want the driver to remain idle before reconnecting.</td>
</tr>
</tbody>
</table>

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.
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\n
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 11 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 =
```
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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRFORMSET</td>
<td>DSI variable. The form set to print, created by another rule, such as DPRLoadImportFile, DPRGetWipFormset, MTCLoadFormset, and so on.</td>
</tr>
<tr>
<td>PRTTYPE</td>
<td>Attachment variable. For DPRPrint to call DPRComplete, set this to COMPLETE.</td>
</tr>
</tbody>
</table>

These INI options are required:

```
< Complete >
  CompleteType = XXX
< Complete:XXX >
  FileType =
  FileName =
  FileExt =
  FilePath =
  Recipient =
```
NOTE: If the Complete control group includes multiple complete types, the DPRComplete rule processes each complete type.

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

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

**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->DPRGetWipFormsset
  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:

```ini
< 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:

<table>
<thead>
<tr>
<th>For this kind of import</th>
<th>The file should begin with</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML file</td>
<td><code>&lt;?xml</code></td>
</tr>
<tr>
<td>Combined NA/POL file</td>
<td><code>WIP=</code></td>
</tr>
<tr>
<td>For this kind of import</td>
<td>The file should begin with</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>V2 import</td>
<td>if not WIP= or &lt;?xml, the system assumes the file is a V2 import format file</td>
</tr>
</tbody>
</table>

**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 CMBNA4, that format is assumed and no automatic check occurs.

For more information, see DPRLoadImportFile on page 159.
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 226.

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.

```ini
< Config:RPEX1 >
    INIFile = rpex1.ini
< Configurations >
    Config = RPEX1
```

**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:

<table>
<thead>
<tr>
<th>Choose</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (zero)</td>
<td>no compression</td>
</tr>
<tr>
<td>1</td>
<td>best speed</td>
</tr>
<tr>
<td>2</td>
<td>default compression</td>
</tr>
<tr>
<td>3</td>
<td>best compression</td>
</tr>
</tbody>
</table>

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

```ini
< 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:

```ini
< 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 = Level12
    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 = int1sm
    DefLib = mstrres\rpex1\deflib\n    FormLib = mstrres\rpex1\forms\n    LbyLib = mstrres\rpex1\forms\n    FormFile = master.lby
< Control >
    XRFExt = .fxr
    ImageEXT = .fap
    DateFormat = 24%
< ArcRet >
    APPIDX = mstrres\rpex1\arc\appidx
    Catalog = mstrres\rpex1\arc\catalog
    CARPath = mstrres\rpex1\arc\n
< 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 Administration Guide for more information.
Chapter 1
Using the Documaker Bridge

**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.

```xml
<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:

```xml
<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:

<table>
<thead>
<tr>
<th>If you enter</th>
<th>IDS treats this as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friday 13:00:00 AM</td>
<td>Fri 1:00:00 PM</td>
</tr>
<tr>
<td>Tue 15:00:00 PM</td>
<td>Tue 3:00:00 PM</td>
</tr>
<tr>
<td>Thur 17:00:00</td>
<td>Thu 5:00 PM</td>
</tr>
<tr>
<td>19:00:00</td>
<td>7:00 PM every day</td>
</tr>
</tbody>
</table>
The client configuration file, docclient.xml, is an initialization file used by client programs, such as executables, Java client programs, 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 the docclient.xml file.

### Basic Options

You can 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 18 for more information.

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

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

```
< ResultQ >
  Name = RESULTQ
</ 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
</ RESType:Default >
```

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
</ RESType:PRT >
```

**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
</ RESType:ERR >
```

**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= irctw32->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 = No
```

You must set the Post option to No 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
```

**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 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:

```
ATCUnloadAttachment
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
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultTimeout</td>
<td>Enter, in milliseconds, the time-out to use if the application did not specify one. The default is 15000L or 15 seconds.</td>
</tr>
<tr>
<td>MaxTimeout</td>
<td>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.</td>
</tr>
<tr>
<td>MinTimeout</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.
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 59
- DPRDecryptLogin on page 69
- DPRDefaultLogin on page 72
- DPRLoginUser on page 169
- DPRGenerateSeedValue on page 102

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.
Chapter 1
Using the Documaker Bridge

- 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 DSIErroMessage 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 DSIMSG_RUN) 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 DSIMSG_RUNR) 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:

```xml
<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.

```xml
<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->;DPRInitLiby</entry>
  <entry name="function">dprw32->;DPRPrintFormset</entry>
</section>

<section name="ReqType:RCP">
```

25
Customizing the login process

Chapter 1
Using the Documaker Bridge

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

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

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.
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:

```xml
< MasterResource >
  HTMLForms =
</ MasterResource >
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLForms</td>
<td>Enter the directory and path where the HTML forms reside. Documaker Bridge checks this directory for <code>filename.htm</code> and <code>filename.html</code> 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: <code>filename_pagenumber.htm</code> For example, <code>myfile_2.htm</code> indicates the second page of multi-page FAP file called <code>myfile.fap</code>. If you need version/revision numbers on the HTML files, use the naming convention Studio uses for FAP files checked out of the library: <code>filename_versionrevision_effdate.htm</code> Here is an example: <code>CANC201B_0000300005_20060101.htm</code> This references FAP file <code>CANC201B</code> 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: <code>CANC201B_0000300005_19800101_2.htm</code> 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.</td>
</tr>
</tbody>
</table>

**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:

```xml
< Debug >
  DPRGetHTMLForms = Yes
</ Debug >
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRGetHTMLForms</td>
<td>Enter Yes to create the log file with information about which file names were checked and which files were found.</td>
</tr>
</tbody>
</table>
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

Use the following rules when you use the Documaker Bridge. The rules are in alphabetical order.

- DPRAddBlankPages on page 35
- DPRAddLogo on page 37
- DPRAddText on page 39
- DPRAddToUserDict on page 41
- DPRAddWipRecord on page 43
- DPRApproveWipRecords on page 45
- DPRArchiveFormset on page 47
- DPRAssignWipRecord on page 50
- DPRBatchArchive on page 52
- DPRBuildGroupList on page 54
- DPRCheck on page 56
- DPRCheckLogin on page 59
- DPRCheckWipRecords on page 60
- DPRCompareXMLFiles on page 64
- DPRConvertGUID on page 66
- DPRCreateEMailAttachment on page 67
- DPRDebug on page 68
- DPRDecryptLogin on page 69
- DPRDecryptValue on page 71
- DPRDefaultLogin on page 72
- DPRDelBlankPages on page 74
- DPRDeleteFiles on page 76
- DPRDeleteWipRecord on page 77
- DPRDelFromUserDict on page 79
- DPRDelMultiWipRecords on page 81
- DPRDepagination on page 83
- DPRDpw2Wip on page 84
- DPREditUserDict on page 86
- DPRExecuteDAL on page 88
- DPRFap2Html on page 89
• DPRFile2Dpw on page 92
• DPRFilterFormsetForms on page 94
• DPRFindTemplate on page 95
• DPRFindWipRecords on page 96
• DPRFindWipRecordsByUser on page 97
• DPRGenerateDefinitionFile on page 100
• DPRGenerateSeedValue on page 102
• DPRGetConfigList on page 104
• DPRGetDFDInfo on page 106
• DPRGetFormList on page 111
• DPRGetFormsetRecips on page 112
• DPRGetHTMLForms on page 114
• DPRGetInitValue on page 115
• DPRGetOneWipRecord on page 116
• DPRGetRecipients on page 118
• DPRGetUserList on page 120
• DPRGetWipList on page 123
• DPRGetWipFormset on page 126
• DPRGetWipRecipients on page 129
• DPRIni2XML on page 132
• DPRInit on page 134
• DPRInitLby on page 135
• DPRLbyCopy on page 136
• DPRLbyDelete on page 138
• DPRLbyGet on page 140
• DPRLbyLock on page 143
• DPRLbyMKCol on page 145
• DPRLbyOptions on page 146
• DPRLbyPropFind on page 147
• DPRLbyPropPatch on page 150
• DPRLbyPut on page 151
• DPRLbyUnlock on page 153
• DPRLoadDPA on page 155
• DPRLoadXML2Formset on page 157
• DPRLoadFAPImages on page 158
• DPRLoadImportFile on page 159
• DPRLoadXMLAttachment on page 161
• DPRLoadXMLFormset on page 162
• DPRLocateOneRecord on page 163
• DPRLockWip on page 164
• DPRLog on page 166
• DPRLogin on page 167
• DPRLoginUser on page 169
• DPRMail on page 170
• DPRMapRecipData on page 172
• DPRModifyUser on page 174
• DPRModifyWipData on page 178
• DPRPatchLevel on page 180
• DPRParseRecord on page 181
• DPRPrint on page 184
• DPRPrintDpw on page 192
• DPRPrintFormset on page 194
• DPRProcessTemplates on page 197
• DPRRenameVars on page 199
• DPRRetFromUserDict on page 200
• DPRRetrieveDPA on page 203
• DPRRetrieveFormset on page 204
• DPRRotateFormsetPages on page 207
• DPRSearch on page 208
• DPRSearchLDAP on page 210
• DPRSearchWip on page 217
• DPRSendFormsetXML on page 222
• DPRSendMultiFiles on page 223
• DPRSendVersion on page 224
List of Rules

- DPRSet2ImageScope on page 225
- DPRSetConfig on page 226
- DPRSSpellCheck on page 228
- DPRSOrmsetForms on page 231
- DPRTemporaryXMLFile on page 232
- DPRTblLookUp on page 233
- DPRTransform on page 243
- DPRUnloadExportFile on page 249
- DPRUnloadXMLFormset on page 251
- DPRUnlockWip on page 252
- DPRUpdateFromMRL on page 253
- DPRUpdateFormsetFields on page 255
- DPRUpdateFormsetFromXML on page 256
- DPRUpdateWipRecords on page 258
- DPRWip2Dpw on page 261
- DPRWipBatchPrint on page 263
- DPRWipIndex2XML on page 267
- DPRWipTableParms on page 269
- DPRXMLDiff on page 272

Use these rules to convert a Metacode print stream into documents for the Internet Document Server:

- MTCLoadFormset on page 273
- MTCPrintFormset on page 275

Use these Documaker Bridge rules to control Documaker Server:

- RPDCheckAttachments on page 277
- RPDCheckRPRun on page 280
- RPDCreateJob on page 283
- RPDDeleteFiles on page 288
- RPDProcessJob on page 290
- RPDRunRP on page 293
- RPDSetPDFAttachmentVariables on page 298
- RPDStopRPRun on page 300
Use these rules to convert TIFF files into PDF documents for the Internet Document Server.

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

Beginning with Shared Objects version 11.2, the system lets you process multi-page CCITT Group 4 single strip TIFF files and other types of multi-page TIFF files. This lets the system print single page, multi-page, and a combination of single and multi-page TIFF files into a PDF file, including color TIFF, dual resolution TIFF, and 32-bit TIFF files.

- **TPDCreateFormset on page 301**
- **TPDCreateOutput on page 303**
- **TPDLoadFormset on page 304**
- **TPDInitRule on page 306**

**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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI(MSGRUNF)</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>If the current page is</th>
<th>Unknown</th>
<th>Front</th>
<th>Back</th>
<th>None</th>
<th>Short</th>
<th>Rolling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
<td>Blank</td>
<td>Blank</td>
<td>Blank</td>
</tr>
<tr>
<td>None</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
<td>Blank</td>
<td>Blank</td>
<td>Blank</td>
</tr>
<tr>
<td>Front</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
</tr>
<tr>
<td>Short</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
</tr>
<tr>
<td>Rolling (Front)</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
<td>Blank</td>
<td>Blank</td>
<td>As is</td>
</tr>
<tr>
<td>Back</td>
<td>As is</td>
<td>As is</td>
<td>Blank</td>
<td>As is</td>
<td>As is</td>
<td>As is</td>
</tr>
<tr>
<td>Rolling (Back)</td>
<td>As is</td>
<td>As is</td>
<td>Blank</td>
<td>As is</td>
<td>As is</td>
<td>As is</td>
</tr>
</tbody>
</table>

**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:

```c
DWORD _VMMAPI FAPAddBlankPages(
    VMHANDLE 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 74
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**

```c
long __DSIAPI DPRAddLogo ( DSIHANDLE hInstance, 
    char * pszParms, 
    unsigned long ulMsg, 
    unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

```ini
[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 rules other than the MTC rules, you must include the name of the form set, as shown here:

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

If you omit the form set, the system assumes MTCFORMSET is its name. You cannot use this rule with the TPDCreateFormset and TPDInitRule rules.
To add a logo, you must add a `AddLogo` control group to the master resource INI file. This control group will have these options:

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

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

**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 186.
**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**

```c
long _DSI_API DPRAddText ( DSIHANDLE hInstance, 
                          char * pszParms, 
                          unsigned long ulMsg, 
                          unsigned long ulOptions)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

```ini
< 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 rules other than the MTC rules, you must include the name of the form set, as shown here:

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

If you omit the form set, the system assumes MTCFORMSET is its name. You cannot use this rule with the TPDCreateFormset and TPDInitRule rules.
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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>The string you want to appear.</td>
</tr>
<tr>
<td>FontID</td>
<td>The font ID that identifies the font you want to use. This ID also specifies the point size of the font.</td>
</tr>
<tr>
<td>Top</td>
<td>Contains the top coordinate (position) of the text in FAP units (2400 units per inch)</td>
</tr>
<tr>
<td>Left</td>
<td>Contains the left coordinate (position) of the text in FAP units (2400 units per inch)</td>
</tr>
<tr>
<td>Pages</td>
<td>(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.</td>
</tr>
<tr>
<td>Angle</td>
<td>(Optional) The default is to display the text at a zero (0) degree angle. You can also enter 90, 180, and 270.</td>
</tr>
<tr>
<td>Color</td>
<td>(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.</td>
</tr>
</tbody>
</table>

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

```ini
< AddText >
  Text   = SAMPLE FORM
  FontID = 11020
  Top    = 12000
  Left   = 12000
  Color = 255
</ AddText >
```
**DPRAddToUserDict**

Use this rule to add words into the user dictionary.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>The pointer to the rule data.</td>
</tr>
<tr>
<td>char *pszParms</td>
<td>The pointer to the rule parameter string.</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>The DSI message.</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options.</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddLine</td>
<td>A line of words you want to add to the user dictionary. Separate the words with commas.</td>
</tr>
</tbody>
</table>
| 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:

```ini
< Spell >
    LanguageOpt =
    UserDict =
    UserDictPath =
```
This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
</tbody>
</table>

See also

- DPRDelFromUserDict on page 79
- DPRRetFromUserDict on page 200
- DPRSpellCheck on page 228
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects this IDS attachment variable:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRFormset</td>
<td>Passes the form set handle.</td>
</tr>
</tbody>
</table>

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique</td>
<td>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.</td>
</tr>
<tr>
<td>UserID</td>
<td>If the input fields do not include CurrUserID and OrigUserID, UserID is used.</td>
</tr>
<tr>
<td>(field names)</td>
<td>The fields are defined in the DFD file. To match a record, Key1, Key2, KeyID and RecType are required (DOC_TAG).</td>
</tr>
</tbody>
</table>

**Attachment outputs**

This rule provides these output attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordID</td>
<td>The record ID.</td>
</tr>
<tr>
<td>RECNUM or/ and UNIQUE_ID</td>
<td>The record ID as defined in the WIP.DFD file.</td>
</tr>
</tbody>
</table>
INI options

You can use these INI options:

```xml
< WIPData >
  File =
  Path =
</ WIPData >
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Enter the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path to the WIP file.</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

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

See also

- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRDelMultiWipRecords on page 81
- DPRDpw2Wip on page 84
- DPRFile2Dpw on page 92
- DPRIni2XML on page 132
- DPRLockWip on page 164
- DPRUnlockWip on page 252
- DPRUpdateWipRecords on page 258
- DPRWip2Dpw on page 261
- DPRWipIndex2XML on page 267
- DPRWipTableParms on page 269
**DPRApproveWipRecords**

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

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserID</td>
<td>The ID of the queue name for the user</td>
</tr>
<tr>
<td>Status</td>
<td>The new status (Approve or Reject)</td>
</tr>
</tbody>
</table>

**INI options**

You can use these INI options with this rule:

<table>
<thead>
<tr>
<th>Option</th>
<th>Control group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>WIPData</td>
<td>Specifies the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>WIPData</td>
<td>Specifies the path to the WIP file.</td>
</tr>
<tr>
<td>File</td>
<td>UserInfo</td>
<td>Specifies the name of the userinfo file.</td>
</tr>
<tr>
<td>Path</td>
<td>UserInfo</td>
<td>Specifies the path to the userinfo file.</td>
</tr>
<tr>
<td>WIP</td>
<td>Status_CD</td>
<td>Specifies the WIP status code.</td>
</tr>
</tbody>
</table>
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:

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

See also DPRCheckWipRecords on page 60
DPRGetWipList on page 123
DPRGetWipFormset on page 126
DPRGetWipRecipients on page 129
DPRSearchWip on page 217
DPRUpdateWipRecords on page 258
**DPRArchiveFormset**

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

**Syntax**

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

**Parameters**

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

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 Administration Guide for information on how you can use the AFEWIP2ArchiveRecord control group.

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormsetName</td>
<td>The DSI variable name from pszParms. The default is DPRFORMSET.</td>
</tr>
<tr>
<td>FieldNames</td>
<td>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.</td>
</tr>
</tbody>
</table>

**INI options**

You can use these INI options with this rule:
Chapter 2
Documaker Bridge Rules

- **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

<table>
<thead>
<tr>
<th>Control group</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArcRet</td>
<td>Appidx</td>
<td>Enter the path for the application index file, such as <code>mstrres\sampco\arc\appidx</code>.</td>
</tr>
<tr>
<td></td>
<td>ArcPath</td>
<td>Enter the path for the archive, such as <code>mstrres\sampco\arc\</code>.</td>
</tr>
<tr>
<td></td>
<td>CARFile</td>
<td>Enter the name and path for the CAR file, such as <code>mstrres\sampco\arc\archive</code>.</td>
</tr>
<tr>
<td></td>
<td>Catalog</td>
<td>Enter the name and path for the catalog file, such as <code>mstrres\sampco\arc\catalog</code>.</td>
</tr>
<tr>
<td></td>
<td>CARPath</td>
<td>Enter the path for the CAR file, such as <code>mstrres\sampco\arc\</code>.</td>
</tr>
<tr>
<td>Status_CD</td>
<td>Archive</td>
<td>The default is <code>AR</code>.</td>
</tr>
<tr>
<td>Debug</td>
<td>DeleteFiles</td>
<td>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.</td>
</tr>
<tr>
<td>Archival</td>
<td>ArchiveMem</td>
<td>Enter Yes to archive to a database. Enter No to archive to a file. The default is No.</td>
</tr>
</tbody>
</table>

**Returns**
- Success or failure

**Errors**
- This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0017</td>
<td>Cannot locate DSI variable #VARIABLE,#.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add the variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0036</td>
<td>The DSI variable #VARIABLE,# does not contain a valid FAP form set.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
<tr>
<td>DPR0042</td>
<td>Failed to unload File #FILE,# in #LOCATION,#.</td>
</tr>
<tr>
<td>DPR0043</td>
<td>Failed to DBQueryFormatInfo from #FILE,#.</td>
</tr>
<tr>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>DPR0044</td>
<td>Failed to DBInitializeFile #FILE,#.</td>
</tr>
<tr>
<td>DPR0045</td>
<td>Failed to DBOpen #FILE,#.</td>
</tr>
<tr>
<td>DPR0046</td>
<td>Failed to UTLLockARC #FILE,#.</td>
</tr>
<tr>
<td>DPR0047</td>
<td>Failed to ArcInit #FILE,#.</td>
</tr>
<tr>
<td>DPR0048</td>
<td>Failed to ArcArchiveDataFile #FILE,#.</td>
</tr>
</tbody>
</table>
**DPRAssignWipRecord**

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

**Syntax**

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

**Parameters**

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

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 RECNUNM 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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignUserID</td>
<td>Enter the user ID you want to assign the record to.</td>
</tr>
<tr>
<td>RecordID</td>
<td>Enter the record ID. You can define it as the RECNUNM or UNIQUEID in your DFD definition.</td>
</tr>
<tr>
<td>AssignDesc</td>
<td>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.</td>
</tr>
<tr>
<td>(field names)</td>
<td>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.</td>
</tr>
</tbody>
</table>
INI options

You can use these INI options:

```xml
< WIPData >
  File =
  Path =
</ WIPData >
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Enter the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path to the WIP file.</td>
</tr>
</tbody>
</table>

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIBALE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>The virtual memory management API #AOINAME,# failed.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
</tbody>
</table>

See also

- DPRAddWipRecord on page 43
- DPRDeleteWipRecord on page 77
- DPRDpw2Wip on page 84
- DPRFile2Dpw on page 92
- DPRIni2XML on page 132
- DPRLockWip on page 164
- DPRUnlockWip on page 252
- DPRUpdateWipRecords on page 258
- DPRWip2Dpw on page 261
- DPRWipIndex2XML on page 267
- DPRWipTableParms on page 269
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**

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

**Parameters**

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

**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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEWTRN</td>
<td>The name of the input NEWTRN.DAT file, with one record for each transaction that needs to be archived.</td>
</tr>
<tr>
<td>NAFILE</td>
<td>The name of the input NAFILE.DAT file.</td>
</tr>
<tr>
<td>POLFILE</td>
<td>The name of the input POLFILE.DAT file.</td>
</tr>
</tbody>
</table>

**Returns**

Success or failure
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate attachment variable #VARIABLE#</td>
</tr>
<tr>
<td>DPR0045</td>
<td>Failed to DBOpen file #FILE#.</td>
</tr>
<tr>
<td>DPR0060</td>
<td>Cannot open file #FILE#.</td>
</tr>
</tbody>
</table>
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>The configuration value in the DAP.INI file from which you want the rule to return a group list rowset.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUPS</td>
<td>An XML rowset containing the Group1/Group2 combinations for the form set.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>Success or failure</td>
</tr>
</tbody>
</table>

**Example**

Here is an example:

```xml
<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">
```

54
<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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

This rule expects these input attachments:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGINRESULT</td>
<td>(Optional) SUCCESS to continue or FAILURE to stop. Used when the rule follows others.</td>
</tr>
<tr>
<td>USERID</td>
<td>The ID of the queue name for user.</td>
</tr>
<tr>
<td>PARTIALMATCH</td>
<td>(Optional) If Yes, the rule conducts a partial match for the search values provided for this variable. The default is No.</td>
</tr>
<tr>
<td>CASESENSITIVE</td>
<td>(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.</td>
</tr>
<tr>
<td>TABLEINIGROUP</td>
<td>(Optional) The name of the INI control group to get the application index table name from. The default is ArcRet.</td>
</tr>
<tr>
<td>TABLEINIOPTION</td>
<td>(Optional) The name of the INI option to get the application index table name from. The default is AppIdx.</td>
</tr>
<tr>
<td>FIELDNAME</td>
<td>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.</td>
</tr>
<tr>
<td>CHECKINARC</td>
<td>(Optional) If Yes, the rule searches archived records. This variable overwrites the CheckInArc INI option. The default is No.</td>
</tr>
</tbody>
</table>

Returns

This rule returns these output attachments:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDS</td>
<td>Total found records from WIP and ARC.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>SUCCESS or FAILURE</td>
</tr>
</tbody>
</table>
INI options

Use these INI options with this rule:

```ini
< Control >
    CaseSensitiveKeys = No
    CheckInArc = No

< WIPData >
    MaxWIPRecords = 200
    File =
    Path =

< ArcRet >
    MaxRecords = 200
    AppIdx = mstrres/formmaker/arc/appidx
    ArcPath = mstrres/formmaker/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**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CaseSensitiveKeys</td>
<td>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.</td>
</tr>
<tr>
<td>CheckInArc</td>
<td>Enter Yes to search archived records The default is No.</td>
</tr>
</tbody>
</table>

**WIPData control group**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxWIPRecords</td>
<td>Enter the maximum number of WIP records to return. The default is 200.</td>
</tr>
<tr>
<td>File</td>
<td>Enter the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the complete path to the WIP file.</td>
</tr>
</tbody>
</table>

**ArcRet control group**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxRecords</td>
<td>Enter the maximum number of archive records to return. The default is 200.</td>
</tr>
<tr>
<td>AppIdx</td>
<td>Enter the name and path for the AppIdx file. Here is an example: mstrres/formmaker/arc/appidx</td>
</tr>
<tr>
<td>ArcPath</td>
<td>Enter the path to the archive files. Here is an example: mstrres/formmaker/arc/</td>
</tr>
<tr>
<td>CARFile</td>
<td>Enter the name of the archive file. Here is an example: mstrres/formmaker/arc/archive</td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog</td>
<td>Enter the name and path for the catalog file. Here is an example: mstrres/formmaker/arc/catalog</td>
</tr>
<tr>
<td>CARPath</td>
<td>Enter the path for the archive file. Here is an example: mstrres/formmaker/arc/</td>
</tr>
<tr>
<td>AppIdxDFD</td>
<td>Enter the name and path for the AppIdxDFD file. Here is an example: mstrres/formmaker/deflib/appidx.dfd</td>
</tr>
</tbody>
</table>

MasterResource control group

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefLib</td>
<td>Enter the path to the DefLib directory. Here is an example: mstrres/formmaker/deflib</td>
</tr>
</tbody>
</table>

Errors

This rule can return these messages:

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

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 Docupresentation authentication model. For more information, see Authenticating Users in the Internet Document Server Guide.

**Syntax**

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

**Attachment variables**

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

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGINRESULT</td>
<td>If there is an error, this variable is created with the value FAILURE.</td>
</tr>
</tbody>
</table>

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable VARIABLE in the attachment list.</td>
</tr>
<tr>
<td>DPR0053</td>
<td>Unable to get random seed value in #LOCATION,#.</td>
</tr>
<tr>
<td>DPR0054</td>
<td>Invalid login.</td>
</tr>
</tbody>
</table>

**See also**

- DPRDecryptLogin on page 69
- DPRDefaultLogin on page 72
- DPRLoginUser on page 169
- DPRGenerateSeedValue on page 102
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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEYNAME</td>
<td>The name of one of the fields in WIP DFD.</td>
</tr>
<tr>
<td>USERID</td>
<td>The ID of the required name for the user.</td>
</tr>
<tr>
<td>STARTRECORD</td>
<td>The starting record number (default is 1).</td>
</tr>
<tr>
<td>MAXRECORDS</td>
<td>The maximum number of records to be retrieved (default is 20).</td>
</tr>
<tr>
<td>STATUS</td>
<td>A status code specified by the WIP, Approve, and Reject INI options (W, AP, and RJ)</td>
</tr>
<tr>
<td>FIELDNAME</td>
<td>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.</td>
</tr>
<tr>
<td>PARTIALMATCH</td>
<td>If present, the rule includes partial matches.</td>
</tr>
</tbody>
</table>
Attachment outputs

The output attachment variables include:

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

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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Control group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormLib</td>
<td>MasterResource</td>
<td>Specifies the path to the forms.</td>
</tr>
<tr>
<td>ImageExt</td>
<td>Control</td>
<td>Specifies the type of image file.</td>
</tr>
<tr>
<td>LogoExt</td>
<td>Control</td>
<td>Specifies the type of logo image.</td>
</tr>
</tbody>
</table>
Here is an example:

```xml
< MasterResource >
    FormLib = mstrres\sampco\forms\n
    < Control >
        ImageEXT = .fap
        LogoExt = .log
        CaseSensitiveKeys = Yes

    < WIPData >
        File = WIP
        Path = mstrres\sampco\wip\n        MaxWIPRecords = 200

    < UserInfo >
        File = userinfo
        Path = mstrres\n
    < Status_CD >
        WIP = W
        Approve = AP
        Reject = RJ
```

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

**Returns**

Success or failure

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIBLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>The virtual memory management API #AOINAME,# failed.</td>
</tr>
<tr>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DPR0007</td>
<td>The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0025</td>
<td>Cannot add variable #VARIABLE,# to the attachment record #RECORD,#</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
</tbody>
</table>

See also

- DPRAddWipRecord on page 43
- DPRApproveWipRecords on page 45
- DPRAssignWipRecord on page 50
- DPRCheckWipRecords on page 60
- DPRDeleteWipRecord on page 77
- DPRDelMultiWipRecords on page 81
- DPRFindWipRecordsByUser on page 97
- DPRGetWipList on page 123
- DPRGetWipFormset on page 126
- DPRGetWipRecipients on page 129
- DPRModifyWipData on page 178
- DPRSearchWip on page 217
- DPRUpdateWipRecords on page 258
- DPRWipTableParms on page 269
**DPRCompareXMLFiles**

Use this rule to compare XML files.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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
</ ReqType:XML >
```

Set up the ATTACH.MSG file as shown here:

```
USERID=USER
DOCTYPE=DAP
REQTYPE=PRT
CONFIG=RPEX1
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
</ Attachments >
```

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:

```c
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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Can not locate ArcKey variable</td>
</tr>
<tr>
<td>DPR0011</td>
<td>ArcKey does not contain valid data</td>
</tr>
<tr>
<td>DPR0024</td>
<td>Can not create DPRXMLFORMSET variable, and the platform error for DPRCompareXMLFiles resulted from DIFCompareXMLFiles failure.</td>
</tr>
</tbody>
</table>

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**

```c
long __DSI_API DPRConvertGUID ( DSIHANDLE hInstance,
                                char * pszParms,
                                unsigned long ulMsg,
                                unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot find the variable VARIABLE in the attachment list.</td>
</tr>
<tr>
<td>DPR0038</td>
<td>The rule parameters PARAMETERS for rule RULE are not correct or empty.</td>
</tr>
</tbody>
</table>
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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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

**See also** [DPRMail on page 170](#)
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>
**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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGINRESULT</td>
<td>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.</td>
</tr>
<tr>
<td>USERID</td>
<td>The user ID of the requestor.</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>The password of the requestor. A hash value is sent to output attachment.</td>
</tr>
<tr>
<td>PASSWORDENCRIPTED</td>
<td>A flag. If USERID and PASSWORD are encrypted values, set to Yes. The default is Yes.</td>
</tr>
</tbody>
</table>

You have these output attachments on RUNR:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>The user ID of the requestor. It is an encrypted value of the encrypted USERID and hash PASSWORD.</td>
</tr>
<tr>
<td>LOGINRESULT</td>
<td>If an error occurs, it is created with the value FAILURE.</td>
</tr>
</tbody>
</table>

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate the variable VARIABLE in the attachment list.</td>
</tr>
<tr>
<td>DPR0004</td>
<td>The user ID USERID is invalid.</td>
</tr>
<tr>
<td>DPR0005</td>
<td>The password specified for user USERID is incorrect.</td>
</tr>
</tbody>
</table>
### Message Description

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0053</td>
<td>Unable to get a random seed value in #LOCATION,#.</td>
</tr>
</tbody>
</table>

**See also**  
- DPRCheckLogin on page 59  
- DPRDefaultLogin on page 72  
- DPRLoginUser on page 169  
- DPRGenerateSeedValue on page 102
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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

```ini
< 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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot find the variable VARIABLE in the attachment list.</td>
</tr>
<tr>
<td>DPR0037</td>
<td>The attachment value VARIABLE with the value VALUE is not a valid encrypted string.</td>
</tr>
<tr>
<td>DPR0038</td>
<td>The rule parameters PARAMETERS for rule RULE are not correct or empty.</td>
</tr>
</tbody>
</table>
Chapter 2
Documaker Bridge Rules

**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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGINRESULT</td>
<td>If this variable exists and its value is anything other than SUCCESS, the rule does nothing.</td>
</tr>
<tr>
<td>USERID</td>
<td>The user ID of the requestor.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

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

**INI options**

You can use this INI option:

```xml
<UserInfo>
    <UserInfo =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserInfo</td>
<td>Enter the name of the userinfo database file.</td>
</tr>
</tbody>
</table>
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate the variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0003</td>
<td>The user information database FILENAME could not be opened.</td>
</tr>
<tr>
<td>DPR0004</td>
<td>The user ID USERID is invalid.</td>
</tr>
</tbody>
</table>

See also

- DPRCheckLogin on page 59
- DPRDecryptLogin on page 69
- DPRLoginUser on page 169
- DPRGenerateSeedValue on page 102
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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,

```c
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:

```c
DWORD __VMLAPI FAPDelBlankPages(
    VMHANDLE objectH,     /* form set or form handle */
)
```

See also  
- DPRAddBlankPages on page 35  
- DPRLoadFAPImages on page 158
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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEN_TARGETFILENAME</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>Success or failure</td>
</tr>
</tbody>
</table>
**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**

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

**Parameters**

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

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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordID</td>
<td>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.</td>
</tr>
<tr>
<td>(field name)</td>
<td>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.</td>
</tr>
</tbody>
</table>

**INI options**

You can use these INI options:

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Enter the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path to the WIP file.</td>
</tr>
</tbody>
</table>

**Returns**

Success or failure
Chapter 2
Documaker Bridge Rules

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>The virtual memory management API #AOINAME,# failed.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
</tbody>
</table>

See also

DPRAddWipRecord on page 43
DPRAssignWipRecord on page 50
DPRDelMultiWipRecords on page 81
DPRDpw2Wip on page 84
DPRFile2Dpw on page 92
DPRIni2XML on page 132
DPRLockWip on page 164
DPRUnlockWip on page 252
DPRUpdateWipRecords on page 258
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
DPRWipTableParms on page 269
DPRDelFromUserDict

Use this rule to delete words from the user dictionary.

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>The pointer to the rule data.</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>The pointer to the rule parameter string.</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>The DSI message.</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options.</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelLine</td>
<td>A line of words you want deleted from the user dictionary. Separate the words with commas.</td>
</tr>
<tr>
<td>UserDict</td>
<td>The name of the user dictionary. The default is user.tlx.</td>
</tr>
</tbody>
</table>

Attachment outputs

None

INI options

You can use these INI options with this rule:

< Spell >

LanguageOpt =
UserDict =
UserDictPath =
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
</tbody>
</table>

See also

DPRAddToUserDict on page 41
DPRRetFromUserDict on page 200
DPRSpellCheck on page 228
**DPRDelMultiWipRecords**

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

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_ message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects one of these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDID</td>
<td>Multiple record numbers separated by commas.</td>
</tr>
<tr>
<td>RECNUM</td>
<td>Multiple record numbers separated by commas.</td>
</tr>
<tr>
<td>UNIQUE_ID</td>
<td>Multiple record numbers separated by commas for SQL databases.</td>
</tr>
</tbody>
</table>

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 =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Enter the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path to the WIP file.</td>
</tr>
</tbody>
</table>

**Returns**

Success or failure
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>The virtual memory management API #AOINAME,# failed.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
<tr>
<td>DPR0061</td>
<td>Unable to WIPFind the record #RECORDID,#.</td>
</tr>
<tr>
<td>DPR0062</td>
<td>Unable to WIPDelete the record #RECORDID,#.</td>
</tr>
</tbody>
</table>

See also

DPRAddWipRecord on page 43
DPRAssignWipRecord on page 50
DPRDeleteWipRecord on page 77
DPRDpw2Wip on page 84
DPRFile2Dpw on page 92
DPRIni2XML on page 132
DPRLockWip on page 164
DPRUnlockWip on page 252
DPRUpdateWipRecords on page 258
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
DPRWipTableParms on page 269
**DPRDepagination**

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

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>The pointer to the rule data.</td>
</tr>
<tr>
<td>char *pszParms</td>
<td>The pointer to the rule parameter string.</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>The DSI message.</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options.</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRFORMSET</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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->DPRLoadExportFile
```

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0017</td>
<td>Cannot locate DSI variable #VARIABLE,#.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0024</td>
<td>Cannot create DSI variable #VARIABLE,#.</td>
</tr>
<tr>
<td>DPR0036</td>
<td>The DSI variable #VARIABLE,# does not contain a valid FAP form set.</td>
</tr>
</tbody>
</table>
**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**

```c
long __DSI_API DPRDpw2Wip ( DSIHANDLE hInstance,
                           char * pszParms,
                           unsigned long ulMsg,
                           unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects this input attachment variable:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECNUM or UNIQUE_ID</td>
<td>Lets the rule find the correct WIP record.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

The WIP record is stored in this attachment variable:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF_POSTFILE</td>
<td>The file name of DPW file.</td>
</tr>
</tbody>
</table>

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRR0001</td>
<td>The variable RF_POSTFILE is not present.</td>
</tr>
<tr>
<td>DPRR0031</td>
<td>Could not open the DPW file</td>
</tr>
<tr>
<td>DPRR0007</td>
<td>Could not find &lt;WipData&gt; &lt;File&gt;</td>
</tr>
<tr>
<td>DPRR0039</td>
<td>The API failed.</td>
</tr>
</tbody>
</table>

**See also**

- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRFile2Dpw on page 92
- DPRGetOneWipRecord on page 116
- DPRIIni2XML on page 132
DPRLockWip on page 164
DPRUnlockWip on page 252
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
**DPREditUserDict**

Use this rule to edit a user dictionary.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>The pointer to the rule data.</td>
</tr>
<tr>
<td>char *pszParms</td>
<td>The pointer to the rule parameter string.</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>The DSI message.</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options.</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditFile</td>
<td>The name of the input XML file you want to use to edit the user dictionary.</td>
</tr>
<tr>
<td>UserDict</td>
<td>The name of the user dictionary. The default is user.tlx.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

None

**INI options**

You can use these INI options with this rule:

```xml
<Spell>
  LanguageOpt =
  UserDict   =
  UserDictPath =
</Spell>
```
This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
</tbody>
</table>

Here is an example of the edit file layout:

```xml
<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. Use the parameters to 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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

This rule returns an error code if DAL had errors.

**Attachment variables**

None.

**Example**

Here is an example:

```c
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**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0064</td>
<td>Failed to execute DAL script #NAME</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these input attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>(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.</td>
</tr>
</tbody>
</table>
| 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. |
Chapter 2  
Documaker Bridge Rules

Returns
This rule outputs these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>Success or failure.</td>
</tr>
</tbody>
</table>

Errors
This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Missing input attachment variable.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>Virtual memory API failed. Returns the name of the API that failed.</td>
</tr>
<tr>
<td>DPR0010</td>
<td>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.</td>
</tr>
<tr>
<td>DPR0016</td>
<td>Failed to unload the HTML template. Returns the name and location of the template that failed to unload.</td>
</tr>
<tr>
<td>DPR0017</td>
<td>Failed to locate the DSI variable DPRformset.</td>
</tr>
<tr>
<td>DPR0027</td>
<td>Failed to load the form set.</td>
</tr>
<tr>
<td>DPR0028</td>
<td>Failed to load the image.</td>
</tr>
</tbody>
</table>
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**

```c
long _DSIAPI DPRFile2Dpw( DSIHANDLE hInstance,
                         char * pszParms,
                         unsigned long ulMsg,
                         unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**INI options**

Be sure to include these INI options:

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

**Attachment variables**

This rule expects this attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF_POSTFILE</td>
<td>The path to the DPW file.</td>
</tr>
</tbody>
</table>

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0039</td>
<td>The API failed.</td>
</tr>
<tr>
<td>DPR0001</td>
<td>Failed to get the attachment variable.</td>
</tr>
<tr>
<td>DPR0056</td>
<td>File not present.</td>
</tr>
</tbody>
</table>

**See also**

- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRDpw2Wip on page 84
- DPRIni2XML on page 132
- DPRGetOneWipRecord on page 116
- DPRLockWip on page 164
- DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
DPRUnlockWip on page 252
**DPRFilterFormsetForms**

Use this rule to search for forms.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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 161
- DPRLoadedXML2Formset on page 157
- DPRSendFormsetXML on page 222
- DPRUpdateFromMRL on page 253
- DPRSortFormsetForms on page 231
- DPRGetFormList on page 111
- DPRGetHTMLForms on page 114
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

```c
long _DSI_API DPRFindTemplate ( DSIHANDLE hInstance,
char * pszParms,
unsigned long  ulMsg,
unsigned long  ulOptions )
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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

### Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>cannot find attachment variable</td>
</tr>
<tr>
<td>DPR0007</td>
<td>INI option not defined</td>
</tr>
<tr>
<td>DPR0010</td>
<td>API failed, reason unknown</td>
</tr>
</tbody>
</table>
**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 217.
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_ message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

The system expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>The ID of the queue name for the user.</td>
</tr>
<tr>
<td>STARTRECORD</td>
<td>The starting record number. The default is one (1).</td>
</tr>
<tr>
<td>MAXRECORDS</td>
<td>The maximum number of records to be retrieved. The default is 20.</td>
</tr>
<tr>
<td>STATUSCODE</td>
<td>One of statuses specified by WIP, Approve, and Reject (W, AP, and RJ).</td>
</tr>
<tr>
<td>CURRUSER</td>
<td>While this rule does not support USEREPORTTOLIST and paging, if you specify:</td>
</tr>
<tr>
<td></td>
<td>CURRUSER=&quot;UNKNOWN&quot;; the system generates the same unknown user WIP list as does the DPRFindWipRecords/DPRSearchWip rule.</td>
</tr>
<tr>
<td>FIELDNAME=Value</td>
<td>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.</td>
</tr>
</tbody>
</table>
Attachment outputs

The system creates these output attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIP</td>
<td>The WIP status generated from WIP option in the Status_CD control group.</td>
</tr>
<tr>
<td>APPROVE</td>
<td>The approve status generated from Approve option in the Status_CD control group.</td>
</tr>
<tr>
<td>REJECT</td>
<td>The reject status generated from Reject option in the Status_CD control group.</td>
</tr>
<tr>
<td>RECORDS</td>
<td>The number of selected records.</td>
</tr>
<tr>
<td>RECORDSX .FieldName</td>
<td>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.</td>
</tr>
</tbody>
</table>

Request types

ReqType = WFD

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

```ini
< 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:

```ini
< 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.
DPRFindWipRecordsByUser

Option | Description
--- | ---
ImageExt | Specifies the type of image file.
LogoExt | Specifies the type of logo image.

WIPData control group

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Specifies the WIP file name.</td>
</tr>
<tr>
<td>Path</td>
<td>Specifies the path to the WIP file</td>
</tr>
<tr>
<td>MaxWIPRecords</td>
<td>Specifies the maximum records to read in the processQ. Use this to prevent it from slowing due to volume records.</td>
</tr>
</tbody>
</table>

UserInfo control group

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Specifies the USERINFO file name.</td>
</tr>
<tr>
<td>Path</td>
<td>Specifies the path to the USERINFO file. If the USERINFO file is missing, USERID is added to the user list.</td>
</tr>
</tbody>
</table>

Status_CD control group

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIP</td>
<td>Specifies the WIP status code.</td>
</tr>
<tr>
<td>Approve</td>
<td>Specifies the approve status code.</td>
</tr>
<tr>
<td>Reject</td>
<td>Specifies the reject status code.</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

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

See also

DPRSearchWip on page 217
DPRGenerateDefinitionFile

To use Word to create Documaker sections, forms, and paragraph lists, you typically first create for Word a Workspace Definition file (WDF) in Documaker Studio or using this rule. This file, which is in XML format, contains the following:

- Field entries from the Common Fields Dictionary
- DAL triggers
- Recipient information from the BDF file
- Form metadata information
- A list of fonts
- A list of the graphics found in the library
- A time stamp, including a date which identifies when the WDF file was created

The Documaker Add-in for Microsoft Word uses the information in the Workspace Definition file to provide content for the selection lists you use when creating sections, forms, or paragraph lists in Word.

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>The pointer to the rule data.</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>The pointer to the rule parameter string.</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>The DSI message.</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options.</td>
</tr>
</tbody>
</table>

Be sure to set up a request type to handle the request for the file. Here is an example:

```xml
<section name="ReqType:GENDEFXML">
    <entry name="function">atcw32->ATCLoadAttachment</entry>
    <entry name="function">atcw32->ATCUnloadAttachment</entry>
    <entry name="function">dprw32->DPRSetConfig</entry>
    <entry name="function">dprw32->DPRGenerateDefinitionFile</entry>
</section>
```

Input variables

In addition, you must supply these message variables to the request.
Output variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config</td>
<td>Enter the name of the configuration defined in the IDS INI files.</td>
</tr>
<tr>
<td>BDFName</td>
<td>(Optional) Enter the name of the BDF file if you have multiple BDF files. The default is the name of the BDF file specified in the configuration.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefinitionFile</td>
<td>This rule returns the definition file as an attachment in the results message. The file is removed from the IDS system.</td>
</tr>
</tbody>
</table>

Keep in mind that the DPRSetConfig rule and the Config message variable are all that is necessary for IDS to load the configuration requested by the user. The configuration must be loaded for the rule to work.

**NOTE:** See also the *Introduction to Enterprise Web Publishing Services* and the *Documaker Add-In for Microsoft Word User Guide* for more information.
**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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0055</td>
<td>Unable to DSIGlobalDataCreate in #LOCATION,#. The global data setup is probably incorrect.</td>
</tr>
</tbody>
</table>

See also

DPRCheckLogin on page 59
DPRDecryptLogin on page 69
DPRDefaultLogin on page 72
DPRLoginUser on page 169
**DPRGetConfigList**

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

**Syntax**

```c
long __DSI_API DPRGetConfigList(DSIHANDLE hdsi,
                               char * pszParms,
                               ULONG ulMsg,
                               ULONG ulOptions)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

None

**Attachment outputs**

<table>
<thead>
<tr>
<th>Attachment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>Success or failure.</td>
</tr>
<tr>
<td>CONFIGLIST</td>
<td>List of configuration information from the DAP.INI file.</td>
</tr>
</tbody>
</table>

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0013</td>
<td>The initialization file, FILENAME could not be loaded.</td>
</tr>
</tbody>
</table>

**Example**

Here is the request type for docserv.xml:

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

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

```ini
[ ReqType:ewps_doGetLibraries ]
  function = atcw32->ATCLoadAttachment
  function = atcw32->ATCUnloadAttachment
  function = dprw32->DPRGetConfigList
```
Here is an example of the returned attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>SERVERTIMESPENT</td>
<td>0.010</td>
</tr>
<tr>
<td>CONFIGLIST</td>
<td>11</td>
</tr>
<tr>
<td>CONFIGLIST1.CONFIG</td>
<td>AFP2PDF</td>
</tr>
<tr>
<td>CONFIGLIST2.CONFIG</td>
<td>amergen</td>
</tr>
<tr>
<td>CONFIGLIST3.CONFIG</td>
<td>DOCUMERGE</td>
</tr>
<tr>
<td>CONFIGLIST4.CONFIG</td>
<td>EBPPTEST</td>
</tr>
<tr>
<td>CONFIGLIST5.CONFIG</td>
<td>FINANCE</td>
</tr>
<tr>
<td>CONFIGLIST6.CONFIG</td>
<td>INSURE</td>
</tr>
<tr>
<td>CONFIGLIST7.CONFIG</td>
<td>PPDemo</td>
</tr>
<tr>
<td>CONFIGLIST8.CONFIG</td>
<td>RPEX1</td>
</tr>
<tr>
<td>CONFIGLIST9.CONFIG</td>
<td>sampco</td>
</tr>
<tr>
<td>CONFIGLIST10.CONFIG</td>
<td>TIFF2PDF</td>
</tr>
<tr>
<td>CONFIGLIST11.CONFIG</td>
<td>RPEX1</td>
</tr>
</tbody>
</table>
DPRGetDFDInfo

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>The configuration for which to get DFD information.</td>
</tr>
</tbody>
</table>
| 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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>A value of success or failure.</td>
</tr>
<tr>
<td>DFDINFO</td>
<td>The path and file name of the XML document that contains the DFD information requested for a configuration.</td>
</tr>
</tbody>
</table>
This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Unable to find the input attachment variable.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>Virtual memory API error.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>API error.</td>
</tr>
<tr>
<td>DPR0043</td>
<td>Failed to DBQueryFormatInfo for DFD file.</td>
</tr>
<tr>
<td>DPR0049</td>
<td>Failed to create the XML document.</td>
</tr>
<tr>
<td>DPR0071</td>
<td>The value for attachment variable is not supported.</td>
</tr>
<tr>
<td>DPR0073</td>
<td>Failed to retrieve field information for DFD.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>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: &lt; WIPData &gt; Key1 = Company</td>
</tr>
<tr>
<td>KEY</td>
<td>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.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>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: &lt; AFEWIPDisplay &gt; 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 DOCSETTHEADINGS attribute.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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:  
  
  ```xml
  <AFEWIPDisplay>
  FIELD = KEY1,%-30.30s,Company
  </AFEWIPDisplay>
  ```
  
  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
  ``` |
Here is an example of an output file for WIP:

```xml
<?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>
  </WIPKEYS>
</DOCUMENT>
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The actual name in the user-defined DFD.</td>
</tr>
<tr>
<td>KEY</td>
<td>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.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>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:</td>
</tr>
<tr>
<td>DOCSETHEADINGS</td>
<td>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:</td>
</tr>
</tbody>
</table>
**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:

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

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_ message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**See also**

- DPRLoadXMLAttachment on page 161
- DPRLoadedXML2Formset on page 157
- DPRSendFormsetXML on page 222
- DPRUpdateFromMRL on page 253
- DPRFilterFormsetForms on page 94
- DPRSortFormsetForms on page 231
- DPRGetHTMLForms on page 114
**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to the rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRFORMSET</td>
<td>This DSI variable supplies the name of the form set to print, which has</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECIPIENT</td>
<td>The name of the recipient from the POL file.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The recipient description, if specified in the Recip_Names control group,</td>
</tr>
<tr>
<td></td>
<td>or if it is the same as the recipient name in the POL file. The application</td>
</tr>
<tr>
<td></td>
<td>should use DESCRIPTION for displaying the recipient list.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0010</td>
<td>System encountered an internal error of unknown type. The call by LOCATION</td>
</tr>
<tr>
<td></td>
<td>to APINAME failed.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable VARIABLE to the attachment list.</td>
</tr>
<tr>
<td>DPR0025</td>
<td>Cannot add variable VARIABLE to the attachment record RECORD</td>
</tr>
</tbody>
</table>
See also DPRGetRecipients on page 118
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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 161
- DPRLoadedXML2Formset on page 157
- DPRSendFormsetXML on page 222
- DPRUpdateFromMRL on page 253
- DPRFilterFormsetForms on page 94
- DPRSortFormsetForms on page 231
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIGROUP</td>
<td>The name of the INI control group to retrieve the value from.</td>
</tr>
<tr>
<td>INIOPTION</td>
<td>The name of the INI option to retrieve the value from.</td>
</tr>
<tr>
<td>INIVALUE</td>
<td>The name of the attachment variable generated in the input/output queues which will hold the INI value.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>Success or failure</td>
</tr>
</tbody>
</table>

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:

```plaintext
[ReqType:TEST8]
function = atcw32->ATCLogTransaction
function = dsijrule->JavaRunRule,;com/docucorp/ids/rules/
IDSTransactionRule;static;reportTimes;INCLUDEMS
function = atcw32->ATCUnloadAttachment
function = atcw32->ATCUnloadAttachment
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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Attachment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECNUM or UNIQUE_ID</td>
<td>Lets the rule find the correct WIP record.</td>
</tr>
</tbody>
</table>

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>The attachment variable was not found.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The API failed.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>The system could not add the attachment variable.</td>
</tr>
</tbody>
</table>

**See also**

- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRDpw2Wip on page 84
- DPRFile2Dpw on page 92
- DPRGetWipList on page 123
- DPRIni2XML on page 132
- DPRLockWip on page 164
DPRUnlockWip on page 252
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long  ulMsg</td>
<td>DSI_MSGRUNF message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long  ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>ID of the requester</td>
</tr>
<tr>
<td>ARCKEY</td>
<td>Archive key value used to retrieve the data</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECIPIENT</td>
<td>The name of the recipient from POL file</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate the variable named VARIABLE in the attachment list</td>
</tr>
<tr>
<td>DPR0010</td>
<td>The system encountered an internal error of unknown type. The call by LOCATION to APINAME failed</td>
</tr>
<tr>
<td>DPR0011</td>
<td>The attachment field VARIABLE does not contain valid data</td>
</tr>
<tr>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DPR0019</td>
<td>Cannot retrieve data into the FILE file. ARCRetrieveDoc API failed.</td>
</tr>
<tr>
<td></td>
<td>CATALOGKEY = CATALOGKEY CARID = CARID.</td>
</tr>
<tr>
<td>DPR0020</td>
<td>DSLoadFormList API failed on the file named FILE.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add the variable named VARIABLE to the attachment list.</td>
</tr>
<tr>
<td>DPR0025</td>
<td>Cannot add the variable named VARIABLE to the attachment record RECORD.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Maximum size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECURITY</td>
<td>64 bytes</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>64 bytes</td>
</tr>
<tr>
<td>LEVEL</td>
<td>1 byte</td>
</tr>
<tr>
<td>REPORTTO</td>
<td>64 bytes</td>
</tr>
<tr>
<td>USERNAME</td>
<td>25 bytes</td>
</tr>
<tr>
<td>INUSE</td>
<td>1 byte</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>128 bytes</td>
</tr>
</tbody>
</table>

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>Success or an error code.</td>
</tr>
<tr>
<td>RECORDS</td>
<td>The total number of user records.</td>
</tr>
<tr>
<td>RECORDS.X.ID</td>
<td>The user ID of the Xth user record.</td>
</tr>
<tr>
<td>RECORDS.X.USERNAME</td>
<td>The user name for the Xth user record.</td>
</tr>
</tbody>
</table>

$X$ 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:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0003</td>
<td>The user information database, #FILENAME, # could not be opened.</td>
</tr>
<tr>
<td>DPR0004</td>
<td>The user ID #USERID,# is invalid.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0025</td>
<td>Cannot add variable #VARIABLE,# to the attachment record #RECORD,#</td>
</tr>
</tbody>
</table>

Example

For this example, you need this input attachment variable:
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 174
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

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

**Attachment outputs**

The output attachment variables include:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIP</td>
<td>The WIP status generated from WIP option in the Status_CD control group</td>
</tr>
<tr>
<td>Approve</td>
<td>The status generated from Approve option in the Status_CD control group</td>
</tr>
<tr>
<td>Reject</td>
<td>The status generated from Reject option in the Status_CD control group</td>
</tr>
<tr>
<td>Records</td>
<td>The number of selected records.</td>
</tr>
</tbody>
</table>
Chapter 2
Documaker Bridge Rules

Request types

ReqType = WLT

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

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

INI options

You can use these INI options with this rule:

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

Here is an example:

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

Returns

Success or failure
### Errors

This rule can return these messages:

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

**See also**

- DPRApproveWipRecords on page 45
- DPRCheckWipRecords on page 60
- DPRFindWipRecordsByUser on page 97
- DPRGetWipFormset on page 126
- DPRGetWipRecipients on page 129
- DPRSearchWip on page 217
- DPRUpdateWipRecords on page 258
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_ message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>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.</td>
</tr>
</tbody>
</table>

The output attachment variables include:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRFormset</td>
<td>The form set handle used to extract the form set for printing.</td>
</tr>
</tbody>
</table>

**Request types**

**ReqType = WFS**

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

```xml
< ReqType:WFS >
  function = atcw32->ATCLogTransaction
  function = atcw32->ATCLoadAttachment
  function = dprw32->DPRSetConfig
  function = atcw32->ATCUnloadAttachment
  function = dprw32->DPRGetWipFormset
</ ReqType:WFS >
```
## INI options

You can use these INI options with this rule:

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

Here is an example:

```xml
< MasterResource >
  FormLib = mstrres\samco\forms\n< Control >
  ImageExt  = .fap
  LogoExt   = .log
< WIPData >
  File      = WIP
  Path      = mstrres\samco\wip\n```

### Returns

**Success or failure**

### Errors

This rule can return these messages:

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

DPRApproveWipRecords on page 45
DPRCheckWipRecords on page 60
DPRGetWipList on page 123
DPRGetWipRecipients on page 129
DPRSearchWip on page 217
DPRUpdateWipRecords on page 258
DPRGetWipRecipients

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_ message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule expects these input attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields</td>
<td>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.</td>
</tr>
</tbody>
</table>

Attachment outputs

The output attachment variables include:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDS</td>
<td>The number of recipients in the recipient list.</td>
</tr>
<tr>
<td>RECORDSX.RECIPIENT</td>
<td>The name of the recipient from the POL file.</td>
</tr>
<tr>
<td>RECORDSX.DESCRIPTION</td>
<td>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.</td>
</tr>
</tbody>
</table>

Request types

reqType = WRC

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

```plaintext
< 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
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIPData control group</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>Specifies the WIP file name.</td>
</tr>
<tr>
<td>Path</td>
<td>Specifies the path to the WIP file.</td>
</tr>
<tr>
<td>Recip_Names control group</td>
<td></td>
</tr>
<tr>
<td>(recipients)</td>
<td>Include the recipient name on the left and the description on the right of the equals sign.</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

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

- DPRAddWipRecord on page 43
- DPRApproveWipRecords on page 45
- DPRAssignWipRecord on page 50
- DPRCheckWipRecords on page 60
- DPRDeleteWipRecord on page 77
- DPRDelMultiWipRecords on page 81
- DPRGetWipList on page 123
- DPRGetWipFormset on page 126
- DPRSearchWip on page 217
- DPRUpdateWipRecords on page 258
**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**

```c
long _DSIAPI DPRIni2XML ( DSIHANDLE hInstance,
    char * pszParms,
    unsigned long  ulMsg,
    unsigned long  ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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.

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

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:

```bash
cryruw32.exe password
```

you will see the output similar to the following:

```
Encrypted string (2XAUnkxUY1x7i5AnQ4m4E1m00)
```

**INI options**

Include this INI option:

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

**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:

```xml
<INI2XML>
    PutURL = localhost
</INI2XML>
```
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Failed to get attachment variable.</td>
</tr>
</tbody>
</table>

See also

DPRAssignWipRecord on page 50
DPRDeleteWipRecord on page 77
DPRDpw2Wip on page 84
DPRFile2Dpw on page 92
DPRGetOneWipRecord on page 116
DPRLockWip on page 164
DPRUnlockWip on page 252
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Returns**

Success or failure

**See also**

DPRInit on page 134
DPRSetConfig on page 226


**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:

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBYFILE</td>
<td>The resource you want to use for the copy operation. A full path and file name generated by DRPLbyGet rule, which should be run before this rule in the WEBDAVCOPY request type.</td>
</tr>
<tr>
<td>DESTINATIONURI</td>
<td>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</td>
</tr>
<tr>
<td>OVERWRITE</td>
<td>(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.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USERID</td>
<td>(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.</td>
</tr>
<tr>
<td>ARCEFFECTIVEDATE</td>
<td>(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.</td>
</tr>
</tbody>
</table>

### Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>SUCCESS or ERROR.</td>
</tr>
<tr>
<td>WEBDAVERRORCODE</td>
<td>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.</td>
</tr>
</tbody>
</table>
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. |
### Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBDAVERRORCODE</td>
<td>(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.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>Success or Error.</td>
</tr>
<tr>
<td>WEBDAVERRORCODE</td>
<td>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.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Use this command</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>get [path] file</td>
<td>Get a resource.</td>
</tr>
<tr>
<td>head [path] file</td>
<td>Get header info for a resource. (currently works same as get)</td>
</tr>
</tbody>
</table>

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECKOUT</td>
<td>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.</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.dtt  
/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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| **PROPERTIES** | A rowset with a row for the resource specified in RESOURCEURI. The row contains the following properties for a file resource:  <br> **supportedlock** - If locking is allowed, this XML string appears:  
|  | property: &lt;lockentry&gt;  
|  | &lt;lockscope&gt;  
|  | &lt;exclusive/&gt;  
|  | &lt;/lockscope&gt;  
|  | &lt;locktype&gt;  
|  | &lt;write/&gt;  
|  | &lt;/locktype&gt;  
|  | &lt;/lockentry&gt;  
|  | getContentTypeLanguage - 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 |
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.
**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**

```c
long _DSI_API DPRLbyLock ( DSIHANDLE hInstance,  
char * pszParms,  
unsigned long ulMsg,  
unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 URL. If the user ID is omitted from the attachment variable and from the resource URI, the rule will fail. |

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCKOWNER</td>
<td>The user ID that owns the lock.</td>
</tr>
<tr>
<td>LOCKSCOPE</td>
<td>The scope of the lock (exclusive).</td>
</tr>
<tr>
<td>LOCKSUBJECT</td>
<td>The name of the resource locked.</td>
</tr>
<tr>
<td>LOCKDEPTH</td>
<td>The depth of the resource locked (0).</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOCKTYPE</td>
<td>The type of lock (write).</td>
</tr>
<tr>
<td>LOCKTIMEOUT</td>
<td>The time-out value after which the lock will expire (infinity).</td>
</tr>
<tr>
<td>LOCKTOKEN</td>
<td>A unique ID that identifies the resource locked.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>SUCCESS or ERROR.</td>
</tr>
<tr>
<td>WEBDAVERRORCODE</td>
<td>This attachment variable only exists if RESULTS equals ERROR. It can contain one of these values:</td>
</tr>
<tr>
<td></td>
<td>404 - (WebDav 'not found' error code) - The RESOURCEURI cannot be found.</td>
</tr>
<tr>
<td></td>
<td>409 - (WebDav 'conflict' error code) - The RESOURCEURI specified is invalid.</td>
</tr>
<tr>
<td></td>
<td>420 - (WebDav 'method error' error code) - An internal API error or memory error occurred.</td>
</tr>
<tr>
<td></td>
<td>423 - (WebDav 'locked' error code) - The resource is already locked.</td>
</tr>
</tbody>
</table>
**DPRLbyMKCol**

Use this rule to create a collection in Library Manager. This rule supports this WebDav command:

<table>
<thead>
<tr>
<th>Use this command</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>mkcol</td>
<td>Not supported by Library Manager.</td>
</tr>
</tbody>
</table>

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**

```c
long _DSI_API DPRLbyMKCol ( DSIHANDLE hInstance, char * pszParms, unsigned long ulMsg, unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

None

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>ERROR.</td>
</tr>
<tr>
<td>WEBDAVERRORCODE</td>
<td>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.</td>
</tr>
</tbody>
</table>
**DPRLbyOptions**

Use this rule to display the WebDav commands supported by Library Manager. This rule supports this WebDav command:

<table>
<thead>
<tr>
<th>Use this command</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>options [path / url]</td>
<td>Display the options available for a path or URL.</td>
</tr>
</tbody>
</table>

This rule displays the following WebDav commands that are supported by Library Manager:

<table>
<thead>
<tr>
<th>options</th>
<th>get</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>propfind</td>
<td>propgetall</td>
<td>lock</td>
</tr>
<tr>
<td>unlock</td>
<td>delete</td>
<td>copy</td>
</tr>
<tr>
<td>move</td>
<td>proppatch</td>
<td>mkcol</td>
</tr>
</tbody>
</table>

**Syntax**

```c
long _DSI_API DPRLbyOptions ( DSIHANDLE hInstance, char * pszParms, unsigned long ulMsg, unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

None

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIONS</td>
<td>A comma-delimited string of WebDav commands supported by Library Manager.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>SUCCESS.</td>
</tr>
</tbody>
</table>
**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:

<table>
<thead>
<tr>
<th>Use this command</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls [path]</td>
<td>List the contents of a collection.</td>
</tr>
<tr>
<td>cd [path]</td>
<td>Change directories.</td>
</tr>
<tr>
<td>propget [path] [property]</td>
<td>Get a property.</td>
</tr>
<tr>
<td>propfind [path] [property]</td>
<td>Find a property.</td>
</tr>
<tr>
<td>propgetall [path]</td>
<td>List all properties for a resource.</td>
</tr>
</tbody>
</table>

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESOURCEURI</td>
<td>A resource URI specifying a user ID, config, file type, and resource. Here are some examples of resource URIs:</td>
</tr>
<tr>
<td></td>
<td>/userid/config/filetype/resource/</td>
</tr>
<tr>
<td></td>
<td>/userid/config/filetype/</td>
</tr>
<tr>
<td></td>
<td>/userid/config/</td>
</tr>
<tr>
<td></td>
<td>/userid/</td>
</tr>
<tr>
<td>DEPTH</td>
<td>Enter a depth of one (1) for collections or file types in Library Manager. Enter a depth of zero (0) for file resources.</td>
</tr>
</tbody>
</table>
Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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                                                                                                                           |
### INI options

Use these options in the DAP.INI file to see a listing of the configurations that support Library Manager.

```xml
<LbyConfigs>
    <Config>RPEX1</Config>
    <Config>RPEX2</Config>
</LbyConfigs>
```

---

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBDAVERRORCODE</td>
<td>This attachment variable is only present if RESULTS equals ERROR. It can contain one of these values:</td>
</tr>
<tr>
<td></td>
<td>404 - (WebDav 'not found' error code) - The RESOURCEURI cannot be found.</td>
</tr>
<tr>
<td></td>
<td>409 - (WebDav 'conflict' error code) - The RESOURCEURI specified is invalid.</td>
</tr>
<tr>
<td></td>
<td>420 - (WebDav 'method error' error code) - An internal API error or memory error occurred.</td>
</tr>
</tbody>
</table>
**DPRLbyPropPatch**

Use this rule to set or remove properties defined on the resource identified by the RESOURCEURI. This rule supports this WebDav command:

<table>
<thead>
<tr>
<th>Use this command</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>proppatch</td>
<td>Not supported by Library Manager.</td>
</tr>
</tbody>
</table>

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**

```c
long _DSI_API DPRLbyPropPatch ( DSIHANDLE hInstance,
                                char * pszParms,
                                unsigned long ulMsg,
                                unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

None

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>ERROR.</td>
</tr>
<tr>
<td>WEBDAVERRORCODE</td>
<td>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.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Use this command</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>put [path]</td>
<td>Put a file into Library Manager.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>SUCCESS or ERROR.</td>
</tr>
</tbody>
</table>
| 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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESOURCEURI</td>
<td>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.</td>
</tr>
<tr>
<td>USERID</td>
<td>(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.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>SUCCESS or ERROR.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WEBDAVERRORCODE</td>
<td>This attachment variable only exists if RESULTS equals ERROR. It can contain one of these values:</td>
</tr>
<tr>
<td></td>
<td>404 - (WebDav 'not found' error code) - The RESOURCEURI cannot be found.</td>
</tr>
<tr>
<td></td>
<td>409 - (WebDav 'conflict' error code) - The RESOURCEURI specified is invalid.</td>
</tr>
<tr>
<td></td>
<td>420 - (WebDav 'method error' error code) - An internal API error or memory error occurred.</td>
</tr>
<tr>
<td></td>
<td>423 - (WebDav 'locked' error code) - The resource is locked by another user.</td>
</tr>
</tbody>
</table>
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

```c
long _DSI_API DPRLoadDPA ( DSIHANDLE hdsi,
                         char * pszParms,
                         ULONG ulMsg,
                         ULONG ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>Pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSARCFILE</td>
<td>The path to the DPA file that has been retrieved.</td>
</tr>
</tbody>
</table>

Attachment outputs

This rule creates these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLDCONFIG</td>
<td>CONFIG is set to the value in the DPA file during the run forward. The run reverse step returns it to its original value.</td>
</tr>
</tbody>
</table>

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:

```c
[ 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   DPRInitLib on page 135
DPRRetrieveDPA on page 203
DPRRetrieveFormset on page 204
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

```c
long _DSIAPI DPRLoadedXML2Formset ( DSIHANDLE hInstance,
                                  char * pszParms,
                                  unsigned long  ulMsg,
                                  unsigned long  ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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 161
- DPRSendFormsetXML on page 222
- DPRUpdateFromMRL on page 253
- DPRFilterFormsetForms on page 94
- DPRSortFormsetForms on page 231
- DPRGetFormList on page 111
- DPRGetHTMLForms on page 114
**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Returns**

Success or failure

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0017</td>
<td>DSI variable VARIABLE cannot be located. Usually means the form set was not created correctly because of some other error</td>
</tr>
<tr>
<td>DPR0036</td>
<td>The DSI variable #VARIABLE,# does not contain a valid FAP form set</td>
</tr>
</tbody>
</table>

**See also**

- DPRRetrieveFormset on page 204
- DPRDelBlankPages on page 74
- DPRRotateFormsetPages on page 207
**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 Workstation Administration Guide for more information on import file formats.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPORTFILE</td>
<td>The name of the import file</td>
</tr>
<tr>
<td>PRINTFILE</td>
<td>The name of the output file, if omitted the system will generate it.</td>
</tr>
</tbody>
</table>

**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 `WTP=`, 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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate attachment variable IMPORTFILE</td>
</tr>
<tr>
<td>DPR0010</td>
<td>DLL version mismatch, internal API failure</td>
</tr>
<tr>
<td>DPR0018</td>
<td>Cannot load font cross reference file</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable RESULTS to the attachment</td>
</tr>
<tr>
<td>DPR0024</td>
<td>Cannot create DSI variable DPRFORMSET</td>
</tr>
<tr>
<td>DPR0026</td>
<td>Cannot load import file</td>
</tr>
<tr>
<td>DPR0027</td>
<td>Cannot load the form set definition (FORM.DAT) file</td>
</tr>
<tr>
<td>DPR0028</td>
<td>Cannot FAP file for image IMAGENAME.</td>
</tr>
<tr>
<td>DPR0029</td>
<td>Loading of the submitted import file resulted in an empty form set.</td>
</tr>
<tr>
<td>DPR0030</td>
<td>The combination of group names (GROUPNAME1) and (GROUPNAME2) and the form name (FORMNAME) does not exist in the form definition file. Invalid import data.</td>
</tr>
<tr>
<td>DPR0031</td>
<td>Cannot open import file FILE.</td>
</tr>
<tr>
<td>DPR0032</td>
<td>No form name provided in the import file.</td>
</tr>
<tr>
<td>DPR0033</td>
<td>Cannot parse import file. Line: LINEDATA</td>
</tr>
<tr>
<td>DPR0034</td>
<td>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.</td>
</tr>
</tbody>
</table>

See also DPRUnloadExportFile on page 249
**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 _DSI_API DPRLoadXMLAttachment ( DSIHANDLE hdsi,
                                      char * pszParms,
                                      unsigned long  ulMsg,
                                      unsigned long  ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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,MYOWNDELMETER
```

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 157
- DPRSendFormsetXML on page 222
- DPRUpdateFromMRL on page 253
- DPRFilterFormsetForms on page 94
- DPRSorFormsetForms on page 231
- DPRGetFormList on page 111
- DPRGetHTMLForms on page 114
**DPRLoadXMLFormset**

Use this rule to load an XML form set into memory for the DPRPrint rule.

**Syntax**

```c
long __DSI_API DPRLoadXMLFormset ( DSIHANDLE hdsi,
                        char * pszParms,
                        unsigned long  ulMsg,
                        unsigned long  ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_ message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLFORMSET</td>
<td>Specifies the full path and file name of the XML form set.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>Success or failure</td>
</tr>
</tbody>
</table>

**NOTE:** You must pass a CONFIG attachment variable to the DPRSetConfig rule.

**See also**

- DPRPrint on page 184
- DPRSetConfig on page 226
- DPRUnloadXMLFormset on page 251
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

```ini
< 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 208](#)
- [DPRRetrieveFormset on page 204](#)
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**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:

```ini
< WIPLock >
    MatchUserID = Warning
    UnMatchUserID = Error
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MatchUserID</td>
<td>If the record is locked and the user IDs match, present an error, warning, or ignore.</td>
</tr>
<tr>
<td>UnMatchUserID</td>
<td>If the record is locked and user IDs do not match, present an error, warning, or ignore.</td>
</tr>
</tbody>
</table>

**Attachment variables**

Expects these attachment variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVERRIDELOCK</td>
<td>Lets the rule continue even if it's locked. Used if a warning was returned.</td>
</tr>
<tr>
<td>USERID</td>
<td>The user ID you want to lock.</td>
</tr>
<tr>
<td>RECNUM or UNIQUE_ID</td>
<td>Lets the rule find the correct WIP record.</td>
</tr>
</tbody>
</table>

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0041</td>
<td>If the record is locked but should only return a warning.</td>
</tr>
<tr>
<td>DPR0042</td>
<td>If the record is locked but should return an error.</td>
</tr>
</tbody>
</table>
See also
DPRAssignWipRecord on page 50
DPRDeleteWipRecord on page 77
DPRDpw2Wip on page 84
DPRFile2Dpw on page 92
DPRGetOneWipRecord on page 116
DPRIni2XML on page 132
DPRUnlockWip on page 252
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
**DPRLog**

Use this rule to confirm whether an email was sent by the Internet Document Server.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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 =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>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.</td>
</tr>
</tbody>
</table>

See also DPRParseRecord on page 181
DPRMail on page 170
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>user ID of the requestor</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>password of the requestor</td>
</tr>
</tbody>
</table>

This rule creates attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>SUCCESS or an error code.</td>
</tr>
<tr>
<td>RIGHTS, REPORTTO,</td>
<td>values from corresponding columns in the Documaker user table.</td>
</tr>
<tr>
<td>SECURITY and USRMESSENGE</td>
<td></td>
</tr>
</tbody>
</table>

If execution is successful, this rule copies the input attachment into the output attachment.

Returns

Success or failure
This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate the variable <code>VARIABLE</code> in the attachment list.</td>
</tr>
<tr>
<td>DPR0003</td>
<td>The user information database, <code>FILENAME</code>, could not be opened.</td>
</tr>
<tr>
<td>DPR0004</td>
<td>The user ID <code>USERID</code> is invalid.</td>
</tr>
<tr>
<td>DPR0005</td>
<td>The password specified for the user <code>USERID</code> is incorrect.</td>
</tr>
</tbody>
</table>
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**

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

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate the variable VARIABLE in the attachment list.</td>
</tr>
<tr>
<td>DPR0053</td>
<td>Cannot get the random seed value in #LOCATION,#.</td>
</tr>
<tr>
<td>DPR0054</td>
<td>Invalid login.</td>
</tr>
</tbody>
</table>

**See also**

- DPRCheckLogin on page 59
- DPRDecryptLogin on page 69
- DPRDefaultLogin on page 72
- DPRGenerateSeedValue on page 102
**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long  ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long  ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>email address</td>
</tr>
<tr>
<td>Msgbody</td>
<td>body of email</td>
</tr>
<tr>
<td>Subject</td>
<td>subject of email</td>
</tr>
<tr>
<td>Attachment</td>
<td>file used as attachment no template processing</td>
</tr>
<tr>
<td>HTMLBodyFile</td>
<td>file of HTML included in the body</td>
</tr>
<tr>
<td>HTMLAttachFile</td>
<td>HTML attachment from template processing</td>
</tr>
</tbody>
</table>

**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:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Used by DPRParseRecord to define the attachment record.</td>
</tr>
<tr>
<td>Data</td>
<td>Directory to store temporary files.</td>
</tr>
<tr>
<td>Message</td>
<td>Maps variables in DFD to the attachment variables expected by Subject and Address.</td>
</tr>
<tr>
<td>Subject</td>
<td>The DPRMail and DPRCreateEMailAttachment.</td>
</tr>
<tr>
<td>Address</td>
<td>The address.</td>
</tr>
<tr>
<td>T1</td>
<td>Sets template for all request type T1.</td>
</tr>
<tr>
<td>T2</td>
<td>Sets template for all request type T2.</td>
</tr>
</tbody>
</table>
**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**

```c
long _DSI_API DPRMapRecipData ( DSIHANDLE hInstance,
        char * pszParms,
        unsigned long  ulMsg,
        unsigned long  ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Example**

Here is an example that uses the following example INI options and data:

```ini
< 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:**

```ini
...\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,HS," *,600,400
P,(6924,3484,7236,13084),(16010,392,352,312),40,CORRESPOND_MAILTOADDR01
A,F6," *,0,1,0,0,0," *,0,600,0,0,0,0,0,0,0,0,0,0," "
P,(7324,3484,7636,13084),(16010,392,352,312),40,CORRESPOND_MAILTOADDR02
A,F6," *,0,1,0,0,0," *,0,600,0,0,0,0,0,0,0,0,0,0," "
P,(7735,3482,8047,13082),(16010,392,352,312),40,CORRESPOND_MAILTOADDR03
A,F6," *,0,1,0,0,0," *,0,600,0,0,0,0,0,0,0,0,0,0," 
```
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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>Configuration</td>
</tr>
<tr>
<td>ACTION</td>
<td>The type of action you want performed, such as Update, Add or Delete. You can perform one action at a time.</td>
</tr>
<tr>
<td>USERS</td>
<td>The number of user records you want to update, add, or delete. The default is one (1).</td>
</tr>
<tr>
<td>USERS.X:FieldName</td>
<td>Field name of Xth user record. ID is a required field and others are optional.</td>
</tr>
<tr>
<td>NEWUSER.X:FieldName</td>
<td>The field name of Xth new user record to modify. Here are the field names and lengths:</td>
</tr>
<tr>
<td></td>
<td>SECURITY = 64 bytes</td>
</tr>
<tr>
<td></td>
<td>PASSWORD = 64 bytes</td>
</tr>
<tr>
<td></td>
<td>LEVEL = 1 byte</td>
</tr>
<tr>
<td></td>
<td>REPORTTO = 64 bytes</td>
</tr>
<tr>
<td></td>
<td>USERNAME = 25 bytes</td>
</tr>
<tr>
<td></td>
<td>INUSE = 1 byte</td>
</tr>
<tr>
<td></td>
<td>MESSAGE = 128 bytes</td>
</tr>
</tbody>
</table>

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:

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

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_DFRModifyUser]
  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
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Enter the name of the UserInfo file.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path to the UserInfo file you entered in the File option.</td>
</tr>
<tr>
<td>UserInfo</td>
<td>Enter the name and full path of the UserInfo file.</td>
</tr>
</tbody>
</table>

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

Returns

Success or failure
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0003</td>
<td>The user information database #FILENAME,# could not be opened.</td>
</tr>
<tr>
<td>DPR0004</td>
<td>The user ID #USERID,# is invalid.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0025</td>
<td>Cannot add variable #VARIABLE,# to the attachment record #RECORD,#</td>
</tr>
<tr>
<td>DPR0089</td>
<td>Unable to update #USERID,# to the userinfo database.</td>
</tr>
<tr>
<td>DPR0090</td>
<td>Unable to add #USERID,# to the userinfo database.</td>
</tr>
<tr>
<td>DPR0091</td>
<td>User #USERID,# exists already. Cannot add it again.</td>
</tr>
<tr>
<td>DPR0092</td>
<td>Unable to delete #USERID,# from the userinfo database.</td>
</tr>
<tr>
<td>DPR0093</td>
<td>Cannot modify user records requested by #USERID,# because the action mode was not specified.</td>
</tr>
</tbody>
</table>

See also DPRGetUserList on page 120
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordID</td>
<td>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.</td>
</tr>
<tr>
<td>(field names)</td>
<td>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.</td>
</tr>
<tr>
<td>NEWWIP. FieldName</td>
<td>Used to update the record. Note that RECORDID (or RECNUM, or UNIQUE_ID) and FORMSETID will be ignored.</td>
</tr>
</tbody>
</table>

**INI options**

You can use these INI options:

- `< WIPData >
  File =
  Path =`

**Option**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Enter the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path to the WIP file.</td>
</tr>
</tbody>
</table>

**Returns**

Success or failure
This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIBALE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>The virtual memory management API #AOINAME,# failed.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APiname,# failed.</td>
</tr>
</tbody>
</table>

See also DPRAddWipRecord on page 43
DPRApproveWipRecords on page 45
DPRAssignWipRecord on page 50
DPRDeleteWipRecord on page 77
DPRDelMultiWipRecords on page 81
DPRDpw2Wip on page 84
DPRFile2Dpw on page 92
DPRGetOneWipRecord on page 116
DPRIni2XML on page 132
DPRLockWip on page 164
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
**DPRPatchLevel**

Use this rule to get a Summary Patch Report for IDS (Docupresentment) and for Documaker.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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**

```c
long __DSIAPI DPRParseRecord ( DSIHANDLE hInstance,
                    char * pszParms,
                    unsigned long ulMsg,
                    unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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 67
**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSGRUNF message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

  ```
  ```

**Attachment variables**

These variables are used as input:
These variables are created by this rule:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTEPRINTFILE</td>
<td>This value is created by the DPRPrint rule. This attachment variable is used on RUNR message.</td>
</tr>
<tr>
<td>PRINTPATH</td>
<td>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.</td>
</tr>
<tr>
<td>GEN_TEMPFILE</td>
<td>Used by the client of Documanage bridge. This value is copied from REMOTEPRINTFILE attachment variable and is added to output attachment on RUNR message.</td>
</tr>
<tr>
<td>GEN_DESTINATION</td>
<td>Used by the client of Documanage bridge. This value is built from PRINTFILE attachment variable. Added to the output attachment on RUNR message.</td>
</tr>
</tbody>
</table>

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot find variable <em>VARIABLE</em> in the attachment.</td>
</tr>
</tbody>
</table>

See also DPRPrint on page 184
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

```c
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>Pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

### Attachment variables

This rule expects:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRFORMSET</td>
<td>This DSI variable contains the form sets to print and is created by some other rule, such as the DPRLoadImportFile rule.</td>
</tr>
<tr>
<td>PRINTFILE</td>
<td>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.</td>
</tr>
</tbody>
</table>
Returns

Success or failure

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0010</td>
<td>Internal API failure, usually indicates the system is not configured correctly</td>
</tr>
<tr>
<td>DPR0022</td>
<td>The attachment variable cannot be located</td>
</tr>
</tbody>
</table>
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 =
</ AddLogo >
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo</td>
<td>The name of the logo you want to use. Store this logo in the FORMS directory of the master resource library.</td>
</tr>
<tr>
<td>Top</td>
<td>Contains the top coordinate (position) of the logo in FAP units (2400 units per inch)</td>
</tr>
<tr>
<td>Left</td>
<td>Contains the left coordinate (position) of the logo in FAP units (2400 units per inch)</td>
</tr>
<tr>
<td>Pages</td>
<td>(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.</td>
</tr>
<tr>
<td>Color</td>
<td>(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.</td>
</tr>
</tbody>
</table>
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:

```xml
< WIPData >
    Key1 = Company
    Key2 = Key2
    KeyID = KeyID
    TranCode = TranCode
    StatusCode = StatusCode
    Desc = Desc
</ WIPData >
```

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
<?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:

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

You need these input attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>Configuration</td>
</tr>
<tr>
<td>RECORDID</td>
<td>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.</td>
</tr>
<tr>
<td>PRTTYPE</td>
<td>XXX. If the INI option is not found, the system uses the default printer type of PDF.</td>
</tr>
<tr>
<td>PRINTPATH</td>
<td>The full path for the print file.</td>
</tr>
<tr>
<td>PRINTFILE</td>
<td>The output file name with or without a full path.</td>
</tr>
<tr>
<td></td>
<td>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).</td>
</tr>
<tr>
<td></td>
<td>If PRINTFILE does not include a path, the system checks PRINTPATH. If PRINTPATH does not exist, the system checks the FileDir option.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>If both PRINTFILE is omitted and the PrtType:XXX control group options are omitted, the system creates a unique name.</td>
</tr>
</tbody>
</table>
You need these INI options:

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Printer control group</strong></td>
<td></td>
</tr>
<tr>
<td>PrtType</td>
<td>Optional. Specify the printer type.</td>
</tr>
<tr>
<td><strong>PrtType:XXX control group</strong></td>
<td></td>
</tr>
</tbody>
</table>
| 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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>Configuration</td>
</tr>
<tr>
<td>RECORDID</td>
<td>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.</td>
</tr>
<tr>
<td>FILETYPE</td>
<td>The file type. The default is V2.</td>
</tr>
</tbody>
</table>
| 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    =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 |
| Ext     | (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:

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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: 

```ini
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:

```ini
File = ~KeyID
```
| Ext    | (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:

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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: 

```ini
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:

```ini
File = ~KeyID
```
| Ext    | (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 37  
DPRPrintFormset on page 194  
DPRRetrieveFormset on page 204  
DPRProcessTemplates on page 197  
DPRUnloadExportFile on page 249
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBUG</td>
<td>(Optional) If this rule argument is present, the rule will set the debug flag for DPWLIB.</td>
</tr>
<tr>
<td>ATTACHVARNAME or ATTACHVARNAME = VALUE</td>
<td>(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.</td>
</tr>
</tbody>
</table>

**Attachment variables**

None

**Attachment outputs**

<table>
<thead>
<tr>
<th>Attachment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>Returns Success or failure.</td>
</tr>
</tbody>
</table>

**Example**

Here is an example request type:

```xml
<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 DPWLIE 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 _DSI_API DPRPrintFormset ( DSIHANDLE hInstance,
                               char * pszParms,
                               unsigned long ulMsg,
                               unsigned long ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>ID of the requester.</td>
</tr>
<tr>
<td>ARCKEY</td>
<td>Documaker archive key value used to retrieve the data.</td>
</tr>
<tr>
<td>PRTTYPE</td>
<td>(Optional) Defaults to PDF, the name of printer in the PrtType control group.</td>
</tr>
<tr>
<td>PRINTPATH</td>
<td>(Optional) Location of the output print file, if this value is missing, the system uses PrintPath option in the Attachments control group.</td>
</tr>
<tr>
<td>RECIPIENT</td>
<td>The name of the Documaker recipient from POL file.</td>
</tr>
</tbody>
</table>

This rule creates these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTEPRINTFILE</td>
<td>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\0001AB0.pdf</td>
</tr>
<tr>
<td>RESULTS</td>
<td>SUCCESS or DPRXXXX error code</td>
</tr>
</tbody>
</table>

If the execution was successful, this rule copies input attachment into the output attachment.
INI options
This rule uses these INI options:

<table>
<thead>
<tr>
<th>Control Group</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachments</td>
<td>Debug</td>
<td>If set to Yes, the temporary NA and POL files are not removed. This is useful for debugging purposes. The default is No.</td>
</tr>
<tr>
<td>PrintPath</td>
<td>Location for the output PDF file, this option is ignored if attachment variable PRINTPATH exists.</td>
<td></td>
</tr>
<tr>
<td>MasterResource</td>
<td>DefLib</td>
<td>Location of the Documaker resources DefLib. Defaults to current directory.</td>
</tr>
<tr>
<td></td>
<td>XRFFile</td>
<td>Name of the FXR file, no default. If you omit this option, an error occurs.</td>
</tr>
<tr>
<td>Control</td>
<td>XRFEExt</td>
<td>Extension of the FXR file. Defaults to FXR</td>
</tr>
<tr>
<td></td>
<td>FormLib</td>
<td>Location of Documaker resources. Defaults to the current directory.</td>
</tr>
<tr>
<td></td>
<td>ImageExt</td>
<td>Extension of Documaker image files. Defaults to FAP</td>
</tr>
<tr>
<td></td>
<td>LogoExt</td>
<td>Extension of Documaker logo files. Defaults to LOG</td>
</tr>
<tr>
<td></td>
<td>TimeOut</td>
<td>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.</td>
</tr>
<tr>
<td>Recip_Names</td>
<td>(Optional) Use this INI control group to translate short recipient names from POL file into long names.</td>
<td></td>
</tr>
<tr>
<td>PrtType:PDF</td>
<td>See the chapter on using the PDF Converter in the Internet Document Server Guide for more information.</td>
<td></td>
</tr>
</tbody>
</table>

Returns
Success or failure

Errors
This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot find variable VARIABLE in the attachment list.</td>
</tr>
<tr>
<td>DPR0010</td>
<td>The system encountered an internal error of unknown type. The call by LOCATION to APINAME failed. Usually indicates improper configuration or DLL version mismatch.</td>
</tr>
<tr>
<td>DPR0011</td>
<td>The attachment field VARIABLE does not contain valid data. Means a required variable, such as ARCKEY, is blank.</td>
</tr>
<tr>
<td>DPR0014</td>
<td>Documaker platform error.</td>
</tr>
<tr>
<td>DPR0018</td>
<td>Cannot load font cross reference file.</td>
</tr>
<tr>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DPR0019</td>
<td>Cannot retrieve data into the file. ARCRetrieveDoc API failed. Means the archive is corrupt or Documaker internal error.</td>
</tr>
<tr>
<td>DPR0020</td>
<td>DSLoadFormList API failed on file FILE. The POLFILE.DAT file cannot be loaded.</td>
</tr>
<tr>
<td>DPR0021</td>
<td>DSLoadNAFormset API failed on file FILE. The NAFILE.DAT file cannot be loaded.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable VARIABLE to the attachment list.</td>
</tr>
</tbody>
</table>
**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:

```xml
<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
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>Use this option to specify the template to use for the main page.</td>
</tr>
<tr>
<td>DebugXML</td>
<td>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.</td>
</tr>
<tr>
<td>History</td>
<td>These options serve as examples of how you specify the templates to use for the remaining pages.</td>
</tr>
<tr>
<td>Details</td>
<td></td>
</tr>
<tr>
<td>PrintPath</td>
<td>Use this option to tell the system where to place the output files it creates.</td>
</tr>
</tbody>
</table>

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.

```xml
<TEMPLATES>
    <MAINPAGE>
        7C311063A8F2F811D3F0B6C600A028CC56DF6578.htm
    </MAINPAGE>
    <Details>
        B6313FD4CDF2F711D322B6C600A048CC56DF659A.htm
    </Details>
    <History>
        B6313FD6CFF2F711D326B6C600A050CC56DF659B.htm
    </History>
</TEMPLATES>
```
Syntax

```c
long _DSI_API DPRProcessTemplate ( DSIHANDLE hdsi,
                                   char * pszParms,
                                   unsigned long  ulMsg,
                                   unsigned long  ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>Pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td><code>DSI_MSG????</code> message, such as <code>DSI_MSGRUNF</code></td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrintFile</td>
<td>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.</td>
</tr>
<tr>
<td>PrintPath</td>
<td>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.</td>
</tr>
<tr>
<td>PRTType</td>
<td>The default extension for the output file names, if the specification of the templates in the INI file did not include an extension.</td>
</tr>
</tbody>
</table>

This rule creates these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemotePrintFile</td>
<td>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 <code>html\</code>, the result will be something like <code>html\00001AB0.pdf</code>.</td>
</tr>
<tr>
<td>Results</td>
<td>Success or a DPRXXXX error code</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0022</td>
<td>Cannot add variable VARIABLE to the attachment list.</td>
</tr>
<tr>
<td>DPR0037</td>
<td>Cannot find the template file #FILE,#.</td>
</tr>
</tbody>
</table>

See also

DPRPrint on page 184
DPRPrintFormset on page 194
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot find the variable VARIABLE in the attachment list.</td>
</tr>
</tbody>
</table>
**DPRRetFromUserDict**

Use this rule to retrieve words from a user dictionary.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>The pointer to the rule data.</td>
</tr>
<tr>
<td>char *pszParms</td>
<td>The pointer to the rule parameter string.</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>The DSI message.</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options.</td>
</tr>
</tbody>
</table>

**INI options**

You can use these INI options with this rule:

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageOpt</td>
<td>Enter the language option. The default is US English. You can choose from these languages and dictionaries:</td>
</tr>
<tr>
<td></td>
<td>Danish “ssceda.tlx,ssceda2.clx”</td>
</tr>
<tr>
<td></td>
<td>Dutch “sscedu.tlx,sscedu2.clx”</td>
</tr>
<tr>
<td></td>
<td>Finnish “sscefi.tlx,sscefi2.clx”</td>
</tr>
<tr>
<td></td>
<td>French “sscefr.tlx,sscefr2.clx”</td>
</tr>
<tr>
<td></td>
<td>German “sscege.tlx,sscege2.clx”</td>
</tr>
<tr>
<td></td>
<td>Italian “ssceit.tlx,ssceit1.clx”</td>
</tr>
<tr>
<td></td>
<td>Norwegian “sscenb.tlx,sscenb2.clx”</td>
</tr>
<tr>
<td></td>
<td>Portuguese_Brazil “sscep.th,sscep2.clx”</td>
</tr>
<tr>
<td></td>
<td>Portuguese “sscepo.tlx,sscepo2.clx”</td>
</tr>
<tr>
<td></td>
<td>Spanish “sscesp.tlx,sscesp2.clx”</td>
</tr>
<tr>
<td></td>
<td>Swedish “sscesw.tlx,sscesw2.clx”</td>
</tr>
<tr>
<td></td>
<td>UK English “sscebr.tlx,sscebr2.clx”</td>
</tr>
<tr>
<td></td>
<td>US English “ssceam.tlx,ssceam2.clx”</td>
</tr>
<tr>
<td>UserDict</td>
<td>Enter the name of the user dictionary. The default is user.tlx.</td>
</tr>
<tr>
<td>UserDictPath</td>
<td>Enter the path to the user dictionary. The default is the current working directory.</td>
</tr>
</tbody>
</table>
### Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETFILE</td>
<td>The output XML file name with full path for the new XML document tree.</td>
</tr>
<tr>
<td></td>
<td>This file will include all words retrieved from the user dictionary.</td>
</tr>
<tr>
<td></td>
<td>If you omit the name, the system generates a unique name for you.</td>
</tr>
<tr>
<td></td>
<td>If you omit the path, the system checks the UserDictPath INI option to get</td>
</tr>
<tr>
<td></td>
<td>the default path. If no path is specified there, the system exports the</td>
</tr>
<tr>
<td></td>
<td>file to the current working directory.</td>
</tr>
<tr>
<td>LanguageOpt</td>
<td>The language selection. The default is US English. You can choose from these</td>
</tr>
<tr>
<td></td>
<td>languages and dictionaries:</td>
</tr>
<tr>
<td>Danish</td>
<td>“ssceda.tlx,ssceda2.clx”</td>
</tr>
<tr>
<td>Dutch</td>
<td>“sscedu.tlx,sscedu2.clx”</td>
</tr>
<tr>
<td>Finnish</td>
<td>“sscefi.tlx,sscefi2.clx”</td>
</tr>
<tr>
<td>French</td>
<td>“sscefr.tlx,sscefr2.clx”</td>
</tr>
<tr>
<td>German</td>
<td>“sscege.tlx,sscege2.clx”</td>
</tr>
<tr>
<td>Italian</td>
<td>“ssceit.tlx,ssceit1.clx”</td>
</tr>
<tr>
<td>Norwegian</td>
<td>“sscenb.tlx,sscenb2.clx”</td>
</tr>
<tr>
<td>Portuguese_Brazil</td>
<td>“sscep.b.tlx,sscep.b.clx”</td>
</tr>
<tr>
<td>Portuguese</td>
<td>“sscepo.tlx,sscepo2.clx”</td>
</tr>
<tr>
<td>Spanish</td>
<td>“sscesp.tlx,sscesp2.clx”</td>
</tr>
<tr>
<td>Swedish</td>
<td>“sscesw.tlx,sscesw2.clx”</td>
</tr>
<tr>
<td>UK English</td>
<td>“sscebr.tlx,sscebr2.clx”</td>
</tr>
<tr>
<td>US English</td>
<td>“ssceam.tlx,ssceam2.clx”</td>
</tr>
<tr>
<td>UserDict</td>
<td>The name of the user dictionary. The default is user.tlx.</td>
</tr>
</tbody>
</table>

### Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETFILE</td>
<td>The name of the retrieved XML tree file with a full path. See the discussion</td>
</tr>
<tr>
<td></td>
<td>of this variable in the input attachment table.</td>
</tr>
</tbody>
</table>

### Returns

Success or failure

### Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE, # in the attachment list.</td>
</tr>
<tr>
<td>RPD0002</td>
<td>Cannot create #TAGNAME, # at #LOCATION, #.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE, # to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION, # to API #APINAME, # failed.</td>
</tr>
<tr>
<td>DPR0051</td>
<td>Failed to unload XML file #FILE, # in #LOCATION, #.</td>
</tr>
</tbody>
</table>
Here is an example of the retrieved file layout:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<SPELLER TYPE="IENTRY" VERSION="3.1">
<FIELDH>spelling</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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSARCFILE</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLDCONFIG</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 184
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>Pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

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:

```ini
< 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.
Attachment variables

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserID</td>
<td>The ID of the requester.</td>
</tr>
<tr>
<td>ARCKEY</td>
<td>The Documaker archive key value used to retrieve the data.</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot find variable VARIABLE in the attachment list.</td>
</tr>
<tr>
<td>DPR0010</td>
<td>Internal API failure, usually indicates something is configured incorrectly or there is a DLL version mismatch.</td>
</tr>
<tr>
<td>DPR0011</td>
<td>The attachment field VARIABLE does not contain valid data. Means the required variable, such as ARCKEY, is blank.</td>
</tr>
<tr>
<td>DPR0014</td>
<td>Documaker platform error.</td>
</tr>
<tr>
<td>DPR0018</td>
<td>Cannot load the font cross-reference file (FXR).</td>
</tr>
<tr>
<td>DPR0019</td>
<td>Cannot retrieve data into the file. ARCRetrieveDoc API failed. Means the archive is corrupt or there was a Documaker internal error.</td>
</tr>
<tr>
<td>DPR0020</td>
<td>DSLoadFormList API failed on the file named FILE. The POLFILE.DAT file cannot be loaded.</td>
</tr>
<tr>
<td>DPR0021</td>
<td>DSLoadNAFormset API failed on the file named FILE. The NAFILE.DAT file cannot be loaded.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable VARIABLE to the attachment list.</td>
</tr>
</tbody>
</table>

See also DPRPrint on page 184
DPRPrintFormset on page 194
**DPRRotateFormsetPages**

Use this rule 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 DPRLoadFAPIImages 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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0017</td>
<td>DSI variable VARIABLE cannot be located</td>
</tr>
</tbody>
</table>
**DPRSearch**

Use this rule to return a list of matching archive records.

**Syntax**

```c
long _DSI_API DPRSearch ( DSIHANDLE hInstance,
                          char * pszParms,
                          unsigned long ulMsg,
                          unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>The user ID of the requestor</td>
</tr>
<tr>
<td>FIELDS</td>
<td>The comma-delimited list of columns in application index file, to be used in the query</td>
</tr>
<tr>
<td>MAXRECORDS</td>
<td>(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.</td>
</tr>
<tr>
<td>CASESENSITIVE</td>
<td>If this attachment variable is present, the search is case sensitive.</td>
</tr>
<tr>
<td>RESTART</td>
<td>The condition for setting the start position before the search</td>
</tr>
<tr>
<td>PARTIALMATCH</td>
<td>If this value is present, the search condition uses partial match, so the value A matches the column value ABC.</td>
</tr>
<tr>
<td>TABLEINIGROUP</td>
<td>(Optional) The name of the INI control group to get the application index table name from, the default is the ArcRet control group</td>
</tr>
<tr>
<td>TABLEINIOPTION</td>
<td>(Optional) The name of the INI option to get the application index table name from, the default is Appldx.</td>
</tr>
</tbody>
</table>

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,KeyID`

...then Key1, Key2, and KeyID are required attachment variables.
This rule creates these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MORERECORDS</td>
<td>if there are more matches than was returned, this variable is set to Yes.</td>
</tr>
<tr>
<td>FIRSTRECORD</td>
<td>the number of the first record in the returned record set, 1 when this is</td>
</tr>
<tr>
<td></td>
<td>the first search</td>
</tr>
<tr>
<td>LASTRECORD</td>
<td>the number of the last record in the returned record set</td>
</tr>
<tr>
<td>RECORDS</td>
<td>the attachment record, stem variable with every column from the application</td>
</tr>
<tr>
<td></td>
<td>index table.</td>
</tr>
</tbody>
</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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNMSG</td>
<td>(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.</td>
</tr>
<tr>
<td>LDAP.PROPERTIES</td>
<td>(Optional) The name of a properties file that should be used to look up the options for the rule. The default file name is <code>ldap.properties</code>, which is looked up in the current directory for IDS. This option is only searched as a rule argument or input message variable.</td>
</tr>
<tr>
<td>LDAP.HOST</td>
<td>(Optional) The host name or IP address of the LDAP server. The default is localhost.</td>
</tr>
<tr>
<td>LDAP.PORT</td>
<td>(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).</td>
</tr>
<tr>
<td>LDAP.URL</td>
<td>(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.</td>
</tr>
<tr>
<td>LDAP.UID</td>
<td>(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 <a href="mailto:administrator@pd.com">administrator@pd.com</a>.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LDAP.USER</td>
<td>(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.</td>
</tr>
</tbody>
</table>
| 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=groupOfNames)).  
  Cn=? |
<p>| 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). |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP.MATCH.ATTRIBUTES</td>
<td>(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 <em>cn</em>.</td>
</tr>
<tr>
<td>LDAP.SEARCH.SCOPE</td>
<td>(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).</td>
</tr>
<tr>
<td>LDAP.DEREF.LINK</td>
<td>(Optional) Enter No if you do not want the system to reference links to other nodes during a search. The default is Yes.</td>
</tr>
<tr>
<td>LDAP.VERSION</td>
<td>(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).</td>
</tr>
<tr>
<td>LDAP.SEARCH.LEVEL</td>
<td>(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).</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LDAP.DN.IDENTIFIER</td>
<td>(Optional) The value for this property is used in the following ways:</td>
</tr>
<tr>
<td></td>
<td>- In cases where 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.</td>
</tr>
<tr>
<td></td>
<td>- In cases where 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 (<em>). This tells the system to match any value for the attributes specified. If LDAP.RDNDS property is also provided, the asterisk (</em>) is replaced with identifier=userID, followed by a comma and the LDAP.RDNDS 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: ((&amp;(objectClass=groupOfNames)(member=*) (CN=Administrator,CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM))) In a case where a value of 'CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM' is specified for LDAP.RDNDS and this property contains a value of CN, the search filter generated would look like this: ((&amp;(objectClass=groupOfNames)(member=CN=Administrator, CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM))). The default is CN.</td>
</tr>
</tbody>
</table>
## Chapter 2

### Documaker Bridge Rules

#### Attachment outputs

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. |

#### Example

Here is an example of a properties file:

```plaintext
ldap.uid=Administrator
ldap.pwd=-Encrypted 2XAUnkxUY1x7i5AnQ4m4E1m00
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 2XAUknkXYIxl715AnQ4m81m00
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.ssi=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 2yQgqaRIzRjd6m8L7WD1000
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 2yQgqaRIzRjd6m8L7WD1000
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->ATCLoadAttachment</entry>
<entry name="function">atcw32->ATCUnloadAttachment</entry>
<entry name="function">dprw32->DPRSetConfig</entry>
<entry name="function">dprw32->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
  </ Config:Example >
  ``

  **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
  </ LDAP >
  ``

  **The input message variable that is part of the request:**

  ```
  CONFIG=Example
  ``

  **The request type:**

  ```
  <section name="ReqType:SearchLDAP">
  <entry name="function">atcw32->ATCLoadAttachment</entry>
  <entry name="function">atcw32->ATCUnloadAttachment</entry>
  <entry name="function">dprw32->DPRSetConfig</entry>
  <entry name="function">dprw32->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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Search fields can include date field values which should be specified in one of these formats:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYMMDD</td>
<td>default; search any values equal to the given date.</td>
</tr>
<tr>
<td>&lt;.YYYYMMDD</td>
<td>search any values less than the given date.</td>
</tr>
<tr>
<td>&lt;=.YYYYMMDD</td>
<td>search any values less than or equal to the given date.</td>
</tr>
<tr>
<td>=.YYYYMMDD</td>
<td>search any values equal to the given date.</td>
</tr>
<tr>
<td>&gt;.YYYYMMDD</td>
<td>search any values greater than the given date.</td>
</tr>
<tr>
<td>&gt;=.YYYYMMDD</td>
<td>search any values greater or equal than the given date.</td>
</tr>
<tr>
<td>!=.YYYYMMDD</td>
<td>search any values not equal to the given date.</td>
</tr>
</tbody>
</table>

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 USERREPORTTOLIST 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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULTS</td>
<td>(Optional) If present, the rule checks that its value is SUCCESS. If the value is other than SUCCESS the rule will exit.</td>
</tr>
<tr>
<td>LOGINRESULT</td>
<td>(Optional) If present, the rule checks that its value is SUCCESS. If the value is other than SUCCESS the rule will exit.</td>
</tr>
<tr>
<td>USERID</td>
<td>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 USERREPORTTOLIST input attachment variables. While USERID is a required attachment variable, it is not checked against CURRUSER if the report to functionality is turned off.</td>
</tr>
<tr>
<td>PAGE</td>
<td>(Optional) The page number to display. The default is one (1).</td>
</tr>
<tr>
<td>PAGESIZE</td>
<td>(Optional) The number of records to display per page. The default is 20.</td>
</tr>
<tr>
<td>STARTRECORD</td>
<td>(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.</td>
</tr>
<tr>
<td>PARTIALMATCH</td>
<td>(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.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CASESENSITIVE</td>
<td>(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.</td>
</tr>
<tr>
<td>FIELDNAME</td>
<td>(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.</td>
</tr>
<tr>
<td>STATUS</td>
<td>(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.</td>
</tr>
<tr>
<td>STATUSCODELIST</td>
<td>(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.</td>
</tr>
<tr>
<td>USERLIST</td>
<td>(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.</td>
</tr>
<tr>
<td>USERREPORTTOLIST</td>
<td>(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.</td>
</tr>
</tbody>
</table>
| 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 USERREPORTTOLIST as before and it has no effect if you specify CURRUSER=~UNKNOWN~. |
| RECORDID           | (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. |
| RECNUM             |                                                                                                                                               |
| UNIQUE_ID          |                                                                                                                                               |
Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIP</td>
<td>The WIP status generated from the WIP option in the STATUS_CD control group. The default is WIP.</td>
</tr>
<tr>
<td>APPROVE</td>
<td>The Approve status generated from the Approve option in the STATUS_CD control group. The default is AP.</td>
</tr>
<tr>
<td>REJECT</td>
<td>The Reject status generated from the Reject option in the STATUS_CD control group. The default is RJ.</td>
</tr>
<tr>
<td>STATUS</td>
<td>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.</td>
</tr>
<tr>
<td>MORERECORDS</td>
<td>Indicates there are more records matching the search criteria. This variable is only present if there are more records.</td>
</tr>
<tr>
<td>NEXTPAGE</td>
<td>The next page number. This is only present if there are more records matching the search criteria.</td>
</tr>
<tr>
<td>PREVPAGE</td>
<td>The previous page number. This is only present if the current page is not the first page.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>This attachment variable contains a value of SUCCESS if the rule ran successfully; otherwise, FAILURE.</td>
</tr>
</tbody>
</table>

INI options

You can use these INI options with this rule:

```
< WIPSearchFormatKeys >
  FieldName = Format
</WIPSearchFormatKeys>
```

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
</WIPSearchFormatKeys>
```

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.
</STATUS_CD>
```
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:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Attachment variable not found.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>Virtual memory API error.</td>
</tr>
<tr>
<td>DPR0009</td>
<td>No matches found for the search criteria specified.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Error adding output attachment variable.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>API error - describes the caller name and name of the API that failed.</td>
</tr>
<tr>
<td>DPR0068</td>
<td>INI group not configured correctly.</td>
</tr>
<tr>
<td>DPR0069</td>
<td>No search criteria specified.</td>
</tr>
<tr>
<td>DPR0070</td>
<td>DFD variable not configured correctly.</td>
</tr>
<tr>
<td>DPR0071</td>
<td>Invalid data specified for input attachment variable.</td>
</tr>
</tbody>
</table>

See also

DPRApproveWipRecords on page 45
DPRCheckWipRecords on page 60
DPRGetWipList on page 123
DPRGetWipFormset on page 126
DPRGetWipRecipients on page 129
DPRSearchWip on page 217
DPRUpdateWipRecords on page 258
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long  ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long  ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

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 161
- DPRLoadedXML2Formset on page 157
- DPRUpdateFromMRL on page 253
- DPRFilterFormsetForms on page 94
- DPRSortFormsetForms on page 231
- DPRGetFormList on page 111
- DPRGetHTMLForms on page 114
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

```c
long __DSIAPI DPRSendMultiFiles ( DSIHANDLE hdsi,
       char * pszParms,
       ULONG ulMsg,
       ULONG long ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0007</td>
<td>The INI option #INIOPTION,# could not be located</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
<tr>
<td>DPR0056</td>
<td>The INI option #INIOPTION,# does not contain a valid value in the group #INIGROUP,#.</td>
</tr>
</tbody>
</table>

Example

Here is an example:

```c
[ 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:

```ini
< 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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule creates the attachment record LIBRARIES with these variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>name of the DLL (DPR or PDF)</td>
</tr>
<tr>
<td>VERSION</td>
<td>version of the DLL, string like 100.002.001</td>
</tr>
<tr>
<td>DATE</td>
<td>date the DLL was compiled as MMM DD YYYY</td>
</tr>
<tr>
<td>TIME</td>
<td>time the DLL was compiled as HH:MM:SS in 24-hour format</td>
</tr>
</tbody>
</table>

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**

```c
long _DSI_API DPRSet2ImageScope ( DSIHANDLE hInstance,
                                 char * pszParms,
                                 unsigned long ulMsg,
                                 unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

Here is an example of how you can use this rule to change the scope of the ACCTNUM1 and Service Address fields:

```c
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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.

```ini
< Config:CFG >
    INIFile =
</Config:CFG>
```

For each Config:CFG control group, you must make an entry in the Configurations control group, as shown below:

```ini
< Config:RPEX1 >
    INIFile = rpex1.ini
</Config:RPEX1>
< Configurations >
    Config = RPEX1
</Configurations>
```

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0013</td>
<td>The initialization file, FILENAME, could not be loaded.</td>
</tr>
<tr>
<td>DPR0024</td>
<td>Cannot create DSI variable VARIABLE.</td>
</tr>
</tbody>
</table>
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

\[
\text{long } _\text{DSI}_\text{API } \text{DPRSpellCheck}( \text{DSIHANDLE hdsi, char *pszParms, unsigned ulMsg, unsigned ulOptions } )
\]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>The pointer to the rule data.</td>
</tr>
<tr>
<td>char *pszParms</td>
<td>The pointer to the rule parameter string.</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>The DSI message.</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>Options.</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImportFile</td>
<td>The name of the input XML file to check spelling with a full path, such as:</td>
</tr>
<tr>
<td></td>
<td>d:\idos2.0\spell\spellXML_input.xml</td>
</tr>
</tbody>
</table>

Attachment outputs

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportFile</td>
<td>The name of the output XML file for new XML document tree that includes</td>
</tr>
<tr>
<td></td>
<td>spelling check information, such as</td>
</tr>
<tr>
<td></td>
<td>d:\idos2.0\spell\spellXML_output.xml</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0007</td>
<td>The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0039</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
<tr>
<td>DPR0051</td>
<td>Failed to unload XML file #FILE,# in #LOCATION,#.</td>
</tr>
<tr>
<td>DPR0063</td>
<td>Failed to load XML file #FILE,# in #LOCATION,#.</td>
</tr>
</tbody>
</table>
Here is an export file layout:

```xml
<?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</P>
    <WORD VALUE="timme" POS="11">time</WORD>
    <CHOICE MATCH="TRUE">time</CHOICE>
    <CHOICE>timed</CHOICE>
    <CHOICE>timer</CHOICE>
    <CHOICE>times</CHOICE>
    <CHOICE>timber</CHOICE>
    <CHOICE>timbre</CHOICE>
  </FIELD>
  <P>a new begning for</P>
  <WORD VALUE="begning" POS="6">begging</WORD>
  <CHOICE MATCH="TRUE">begging</CHOICE>
  <WORD>
  </WORD>
  <P>successs.</P>
  <WORD VALUE="successs" POS="0">success</WORD>
  <CHOICE MATCH="TRUE">success</CHOICE>
  <CHOICE>successes</CHOICE>
  <FIELD NAME="document.forms[0].elements[7].value">
    iPPS Livve
    <WORD VALUE="iPPS" POS="0">PP</WORD>
    <CHOICE MATCH="TRUE">PP</CHOICE>
    <CHOICE>PS</CHOICE>
    <CHOICE>its</CHOICE>
  </FIELD>
  <WORD>
  </WORD>
  <WORD VALUE="Livve" POS="5">Live</WORD>
  <CHOICE MATCH="TRUE">Live</CHOICE>
  <CHOICE>Five</CHOICE>
  <CHOICE>Give</CHOICE>
  <CHOICE>Life</CHOICE>
  <CHOICE>Like</CHOICE>
  <CHOICE>Line</CHOICE>
  <CHOICE>Love</CHOICE>
  <FIELD NAME="document.forms[0].elements[8].value">
    begning
    <WORD VALUE="begning" POS="0">begging</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[12].value">GA</FIELD>
  <FIELD NAME="document.forms[0].elements[13].value">30339</FIELD>
</SPELLER>
```
Here is an example of the request type in the docserve.xml file:

```xml
<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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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 161
- DPRLoadedXML2Formset on page 157
- DPRSendFormsetXML on page 222
- DPRUpdateFromMRL on page 253
- DPRFilterFormsetForms on page 94
- DPRGetFormList on page 111
- DPRGetHTMLForms on page 114
**DPRTemporaryXMLFile**

Use this rule to load and unload XML files into or from a temporary file.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY_NAME</td>
<td>The key.</td>
</tr>
<tr>
<td>DESCRIP</td>
<td>A description.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>The value returned. See the TABLERETURNS input attachment variable.</td>
</tr>
</tbody>
</table>

**Attachment variables**

<table>
<thead>
<tr>
<th>Attachment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>A configuration value for a master resource library in the DAP.INI file for IDS.</td>
</tr>
</tbody>
</table>
| 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 outputs

<table>
<thead>
<tr>
<th>Attachment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBLLKUPFILE</td>
<td>Only present if the KEEP input attachment variable or rule argument is present. It contains the path and file name of the output file.</td>
</tr>
</tbody>
</table>

| RESULTS      | Success or failure                                                         |

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEEP</td>
<td>(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.</td>
</tr>
</tbody>
</table>

### Example 1

Here is the request type for this example:

```xml
<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>
</section>
```
<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;transformation;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: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:Body>
        <DSIMSG VERSION="100.020.0">
            <CTLBLOCK>
                <REQTYPE>TBLLKUP</REQTYPE>
                <UNIQUE_ID>5533591529132872004-0-Thread-1</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-Bh5yuEiiOczVSVP9hIr7vVaIyXg0PoiSFO8Y.xml</VAR>
            </MSGVARS>
        </DSIMSG>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
Here is the Xslt template, which is used by the XsltTransformRule:

```xml
<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>
          <![CDATA[
            function setValue(obj) {
              if (obj.value != null)
                window.returnValue = obj.value;
              else
                window.returnValue = "";
            }
            window.close();
          ]]>
        </xsl:element>
      </head>
      <html>
        <head>
          <xsl:element name="script">
            <xsl:attribute name="language">JavaScript 1.2</xsl:attribute>
            <xsl:attribute name="type">text/javascript</xsl:attribute>
            <![CDATA[
              function setValue(obj) {
                if (obj.value != null)
                  window.returnValue = obj.value;
                else
                  window.returnValue = "";
                }
                window.close();
              ]]>
          </xsl:element>
        </head>
      </html>
    </html>
  </xsl:template>
</xsl:stylesheet>
```
Here is the request type for this example:

```
<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>
```

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: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:Body>
    <DSIMSG VERSION="100.020.0">
      <CTLBLOCK>
        <REQTYPE>TBLKUP2</REQTYPE>
        <UNIQUE_ID>4809681331132872004-0-Thread-2</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>
      </MSGVARS>
      <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>
      </ROWSET>
    </DSIMSG>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
Here is the HTML template used by the DPRTransform rule for this example:

```html
<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>
```
Example 3  Here is the request type for this example:

```xml
<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:

```xml
Content-Type: text/xml
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
  <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:

```xml
Content-Type: text/xml
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Body>
    <DSIMSG VERSION="100.020.0">
      <CTLBLOCK>
        <REQTYPE>TBLLKUP3</REQTYPE>
        <UNIQUE_ID>5060623132132872004-0-Thread-3</UNIQUE_ID>
      </CTLBLOCK>
      <MSGVARS>
        <VAR NAME="TABLEFILE">C:\rp\mstrres\insure\table\mktmsg.dbf</VAR>
      </MSGVARS>
    </DSIMSG>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
<REQTYPETBLLKUP3</REQTYPETBLLKUP3>
<UNIQUE_ID>5060623132132872004-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">0swwpsCtVzAQvwErFfyoeoIKkRN7wGG3_5cptomwGuKqlZ.xm
l</VAR>
  <VAR NAME="TEMPLATE">tbllkup.htm</VAR>
  <VAR NAME="TRANSFORMFILE">0pDp_S_-
UehF1YuqRukdoRo6pgqrTMle4AZxwguYrrj.htm</VAR>
</MSGVARS>
<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>
</DSIMSG>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Here is the HTML template used by the DPRTransform rule for this example:

```html
<html>
<head>
  <script language="javascript">
    function setValue(obj)
    {
      if (obj.value != null)
        window.returnValue = obj.value;
      else
        window.returnValue = "";
      window.close();
    }
  </script>
</head>
<body>
</body>
</html>
```
</head>
<body bgcolor="#f2eddb" onscroll="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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Attachment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>A configuration value for a master resource library in the DAP.INI for IDS.</td>
</tr>
<tr>
<td>SOURCE</td>
<td>(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).</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>A full path and file name of a template with embedded XSL to use for the transformation. It must be accessible to IDS.</td>
</tr>
<tr>
<td>FILETYPE</td>
<td>The extension of the output file.</td>
</tr>
<tr>
<td>PRINTPATH</td>
<td>(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.</td>
</tr>
<tr>
<td>TRANSFORMFILE</td>
<td>(Optional) The path and file name of the output file.</td>
</tr>
</tbody>
</table>

Attachment outputs

<table>
<thead>
<tr>
<th>Attachment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSFORMFILE</td>
<td>The path and file name of the output file.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>Success or failure.</td>
</tr>
<tr>
<td>DPRTRANSFORMFILE</td>
<td>The output template, sent as an output attachment, which is part of the output message.</td>
</tr>
</tbody>
</table>

Example 1

Here is the request type for Example 1:

```xml
<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>

Here is the input message for Example 1:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <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:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Body>
    <DSIMSG VERSION="100.020.0">
      <CTLBLOCK>
        <REQTYPE>TBLKUP2</REQTYPE>
        <UNIQUE_ID>2399062548162892004-0-Thread-2</UNIQUE_ID>
      </CTLBLOCK>
      <MSGVARS>
        <VAR NAME="FILETYPE">htm</VAR>
        <VAR NAME="RESULTS">SUCCESS</VAR>
      </MSGVARS>
    </DSIMSG>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Here is the HTML template for Example 1:

```html
<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;-->
```
```
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:Body>
    <DSIMSG VERSION="100.020.0">
      <CTLBLOCK>
        <REQTYPE>TBLLKUP3</REQTYPE>
        <UNIQUE_ID>5060623132132872004-0-Thread-3</UNIQUE_ID>
        <MSGVARS>
          <VAR NAME="CONFIG">AMERGEN</VAR>
          <VAR NAME="KEEP"></VAR>
        </MSGVARS>
      </CTLBLOCK>
    </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: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-ScKnfxexBzy0EyXmCPRMax2.xml</VAR>
        <VAR NAME="TEMPLATE">tbllkup.htm</VAR>
        <VAR NAME="TRANSFORMFILE">0FS7HpzYVxT33h_JxsFsQqV_p-UZmoUEn-0Zyu5jrBlok.htm</VAR>
      </MSGVARS>
      <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>
        </ROW>
        <ROW NUM="2">
          <VAR NAME="ENTRY_NAME">Greeting</VAR>
          <VAR NAME="DESCRIP">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>
        </ROW>
      </ROWSET>
    </DSIMSG>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
Here is the HTML Template for Example 2:

```html
<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 188 for more information.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

To use this rule you will need to specify the following rule name:

```
dprw32->DPRUnloadExportFile
```

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRFORMSET</td>
<td>The form set to export. This form set is created by some other rule, such as the DPRLoadImportFile rule.</td>
</tr>
<tr>
<td>EXPORT</td>
<td>The name of the output file.</td>
</tr>
<tr>
<td>APPENDEDEXPORT</td>
<td>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.</td>
</tr>
<tr>
<td>EXPORTRECIPS</td>
<td>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.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate the variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0017</td>
<td>Cannot locate the DSI variable #VARIABLE,#.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add the variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0035</td>
<td>Cannot open the export file #FILENAME,#. Error reported by OS #ERRORNO,#</td>
</tr>
<tr>
<td>DPR0036</td>
<td>The DSI variable #VARIABLE,# does not contain a valid form set (FAP file).</td>
</tr>
</tbody>
</table>

Returns
Success or failure

See also
DPRPrint on page 184
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP1</td>
<td>The Key1 for a form set.</td>
</tr>
<tr>
<td>GROUP2</td>
<td>The Key2 for a form set.</td>
</tr>
<tr>
<td>PRINTPATH</td>
<td>(Optional) Specifies the print path location of the XML form set.</td>
</tr>
<tr>
<td>XMLIMAGEOPTIONS</td>
<td>(Optional) Unloads all image options for a form set.</td>
</tr>
<tr>
<td>XMLALLFIELDS</td>
<td>(Optional) Unloads all empty field information for a form set.</td>
</tr>
<tr>
<td>XMLALLOBJECTS</td>
<td>(Optional) Unloads all objects for a form set.</td>
</tr>
</tbody>
</table>

Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLFORMSET</td>
<td>Contains the full path and file name of the unloaded XML form set.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>Success or failure</td>
</tr>
</tbody>
</table>

**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 162](#)
**DPRUnlockWip**

Use this rule to unlock a WIP record after it has been edited so other users can make changes to the record.

**Syntax**

```c
long _DSI_API DPRUnlockWip ( DSIHANDLE hInstance,
    char * pszParms,
    unsigned long  ulMsg,
    unsigned long  ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG?? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>The user ID you want to unlock.</td>
</tr>
<tr>
<td>RECNUM or UNIQUE_ID</td>
<td>Lets the rule find the correct WIP record.</td>
</tr>
</tbody>
</table>

**See also**

- DPRAddWipRecord on page 43
- DPRApproveWipRecords on page 45
- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRDelMultiWipRecords on page 81
- DPRDpw2Wip on page 84
- DPRFile2Dpw on page 92
- DPRGetOneWipRecord on page 116
- DPRIni2XML on page 132
- DPLockWip on page 164
- DPRModifyWipData on page 178
- DPRWip2Dpw on page 261
- DPRWipIndex2XML on page 267
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 _DSI_API DPRUpdateFromMRL ( DSIHANDLE hInstance,
      char * pszParms,
      unsigned long  ulMsg,
      unsigned long  ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDS</td>
<td>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.</td>
</tr>
<tr>
<td>XMLFORMID</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXRECORDS</td>
<td>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.</td>
</tr>
<tr>
<td>PAGE</td>
<td>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.</td>
</tr>
<tr>
<td>STARTRECORD</td>
<td>Enter the record you with which you want the rule to start.</td>
</tr>
</tbody>
</table>

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 161
- DPRLoadedXML2Formset on page 157
- DPRSendFormsetXML on page 222
- DPRFilterFormsetForms on page 94
- DPRSortFormsetForms on page 231
- DPRGetFormList on page 111
- DPRGetHTMLForms on page 114
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 than one with the same name. Updating fields that are embedded into text areas will force the reformating 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Returns

Success or failure

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0017</td>
<td>Cannot locate DSI variable #VARIABLE,#.</td>
</tr>
<tr>
<td>DPR0036</td>
<td>The DSI variable #VARIABLE,# does not contain a valid form set</td>
</tr>
</tbody>
</table>
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 DPRUpdateFormsetFromXML ( DSIHANDLE hInstance, char * pszParms, unsigned long  ulMsg, unsigned long  ulOptions )

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSGRUNF message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPRSETALLFIELDS</td>
<td>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.</td>
</tr>
</tbody>
</table>
NOTE: This is relevant only when you are using the WIP Edit plug-in. These attachment variables affect only DPRUpdateFormsetFromXML rule.

<table>
<thead>
<tr>
<th>DPRSETGLOBALFIELDS</th>
<th>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.</th>
</tr>
</thead>
</table>
| 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. |

See also
DPRGetWipFormset on page 126
DPRLoadXMLAttachment on page 161
DPRLoadImportFile on page 159
**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:

```
```

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 _DSI_API DPRUpdateWipRecords ( DSIHANDLE hdsi,
    char * pszParms,
    ULONG ulMsg,
    ULONG ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_ message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

This rule expects these attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIPS</td>
<td>The number of records to be updated.</td>
</tr>
<tr>
<td>WIPSX.FieldName</td>
<td>The value of the field to be updated in the original record.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Specifies the name of the WIP file.</td>
</tr>
<tr>
<td>Path</td>
<td>Specifies the path to the WIP file.</td>
</tr>
<tr>
<td>MaxWIPRecords</td>
<td>Specifies the maximum number of records to read into the processQ. This prevents it from slowing down because of a large volume of records.</td>
</tr>
</tbody>
</table>
Here is an example:

```
< WIPData >
  File    = WIP
  Path    = mstrres\sampco\wip
```

**Returns**  Success or failure

**Errors**  This rule can return these messages:

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

**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 45
DPRCheckWipRecords on page 60
DPRGetWipList on page 123
DPRGetWipFormset on page 126
DPRGetWipRecipients on page 129
DPRSearchWip on page 217
**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 _DSI_API DPRWip2Dpw ( DSIHANDLE hInstance,
                          char * pszParms,
                          unsigned long ulMsg,
                          unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0039</td>
<td>The API failed.</td>
</tr>
<tr>
<td>DPR0001</td>
<td>Failed to get the variable attachment.</td>
</tr>
<tr>
<td>DPR0056</td>
<td>Could not get the file name.</td>
</tr>
</tbody>
</table>

**See also**

- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRDpw2Wip on page 84
- DPRFile2Dpw on page 92
**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rule data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**Attachment variables**

This rule expects these input attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrtType</td>
<td>(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.</td>
</tr>
<tr>
<td>PrtDevice</td>
<td>(Optional) This is the name of the print device.</td>
</tr>
<tr>
<td>PrintFile</td>
<td>(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.</td>
</tr>
<tr>
<td>PrintPath</td>
<td>(Optional) This path points to the location of the print file.</td>
</tr>
<tr>
<td>DPRProofLogo</td>
<td>(Optional) Enter Yes if you want to include a logo. See DPRAddLogo on page 37 for information on setup options.</td>
</tr>
<tr>
<td>RecordIDs</td>
<td>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 RecordIDsX, as shown in the RecordIDsX discussion.</td>
</tr>
</tbody>
</table>
## Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| RecordIDsX | 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 RecordIDsX):

```
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 RecNumsX, as shown in the RecNumsX discussion.

| RecNumsX   | 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 RecNumsX):

```
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 =```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrtType</td>
<td>(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.</td>
</tr>
<tr>
<td>PrintPath</td>
<td>(Optional) The name of the print device. The system ignores this option if input attachment variable PrintPath is present.</td>
</tr>
</tbody>
</table>

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```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logo</td>
<td>The name of the logo you want to use. Store this logo in the FORMS directory of the master resource library.</td>
</tr>
<tr>
<td>Top</td>
<td>Contains the top coordinate (position) of the logo in FAP units (2400 units per inch)</td>
</tr>
<tr>
<td>Left</td>
<td>Contains the left coordinate (position) of the logo in FAP units (2400 units per inch)</td>
</tr>
<tr>
<td>Pages</td>
<td>(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.</td>
</tr>
</tbody>
</table>
Chapter 2
Documaker Bridge Rules

Returns
Success or failure

Errors
This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0001</td>
<td>Cannot locate the variable #VARIABLE,# in the attachment list.</td>
</tr>
<tr>
<td>DPR0022</td>
<td>Cannot add the variable #VARIABLE,# to the attachment list.</td>
</tr>
<tr>
<td>DPR0069</td>
<td>API #APINAME,# failed to print the form set at location #LOCATION,#.</td>
</tr>
<tr>
<td>DPR0084</td>
<td>Failed to get the current record #RECORDID,# in #LOCATION,#.</td>
</tr>
<tr>
<td>DPR0086</td>
<td>Failed to load the WIP form set.</td>
</tr>
</tbody>
</table>

Example
Here is an example request type:

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

Here are some example input attachments:

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

See also
DPRAddLogo on page 37
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:

```xml
< WIP2DPW >
    Menu = wipedit.res
```

Syntax

```c
long __DSI_API DPRWipIndex2XML ( DSIHANDLE hInstance,
                                char * pszParms,
                                unsigned long ulMsg,
                                unsigned long ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule expects this input attachment variable:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECNUM or UNIQUE_ID</td>
<td>Lets the rule find the correct WIP record.</td>
</tr>
</tbody>
</table>

The WIP record is broken into attachment variables.

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0039</td>
<td>The API failed.</td>
</tr>
<tr>
<td>DPR0001</td>
<td>The CONFIG variable was not found.</td>
</tr>
<tr>
<td>DPR0026</td>
<td>Could not open DWP file for creation.</td>
</tr>
<tr>
<td>DPR0010</td>
<td>Could not create WIPXMLVAR.</td>
</tr>
<tr>
<td>DPR0006</td>
<td>VMMMalloc failed.</td>
</tr>
</tbody>
</table>

Attachment outputs

This rule creates these DSI variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIPDATAPTR</td>
<td>An internal variable that contains the WIP buffer.</td>
</tr>
<tr>
<td>WIPXMLVAR</td>
<td>The XML version of the WIP record.</td>
</tr>
</tbody>
</table>
The rule writes out the WIP index portion of the DPW file on run-reverse.

See also

- DPRAddWipRecord on page 43
- DPRApproveWipRecords on page 45
- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRDelMultiWipRecords on page 81
- DPRDpw2Wip on page 84
- DPRFile2Dpw on page 92
- DPRGetOneWipRecord on page 116
- DPRIni2XML on page 132
- DPRLockWip on page 164
- DPRUnlockWip on page 252
- DPRModifyWipData on page 178
- DPRWip2Dpw on page 261
- DPRWipTableParms on page 269
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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>pointer to the rules data</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_message</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

Attachment variables

This rule expects this input attachment variable:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>The user configuration.</td>
</tr>
</tbody>
</table>

Attachment outputs

This rule expects these output attachments:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields</td>
<td>Specifies the WIP fields as defined in the WIP.DFD file.</td>
</tr>
<tr>
<td>Table</td>
<td>Specifies the WIP fields in the WIP table.</td>
</tr>
<tr>
<td>WIPKey</td>
<td>Specifies the WIP fields to fill in the WIP table.</td>
</tr>
<tr>
<td>OptKey</td>
<td>Specifies the action keys (status code) for the SELECTION options.</td>
</tr>
<tr>
<td>AppTxt</td>
<td>Specifies the application options.</td>
</tr>
<tr>
<td>ShwTxt</td>
<td>Specifies the show options.</td>
</tr>
<tr>
<td>EntryTbl</td>
<td>Specifies the entry table for searching table.</td>
</tr>
<tr>
<td>EntryKey</td>
<td>Specifies the entry keys for search records.</td>
</tr>
</tbody>
</table>

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:
Here is an example:

```xml
< 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
	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
</ PrtView_WIPTable >
```

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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR0022</td>
<td>Cannot add variable #VARIABLE,# to the attachment list.</td>
</tr>
</tbody>
</table>

See also

- DPRAddWipRecord on page 43
- DPRApproveWipRecords on page 45
- DPRAssignWipRecord on page 50
- DPRDeleteWipRecord on page 77
- DPRDelMultiWipRecords on page 81
DPRDpw2Wip on page 84
DPRFile2Dpw on page 92
DPRGetOneWipRecord on page 116
DPRIni2XML on page 132
DPRLockWip on page 164
DPRUnlockWip on page 252
DPRModifyWipData on page 178
DPRWip2Dpw on page 261
DPRWipIndex2XML on page 267
**DPRXMLDiff**

Use this rule after the DPRCompareXMLFiles rule to unload the XML file that rule created.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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.
MTCLoadFormset

Use this rule to load the Metacode or AFP print stream into a DAP form set. This rule creates a variable called MTCFORMSET with the value of the DAP form set handle. This rule expects the value METACODEFILE in the attachment with the name of the file to load. This rule destroys the DAP form set on the DSI_MSGRUNR message.

Syntax

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

NOTE: The DPRPrint rule also works with the Documanage Bridge as well as the Documaker 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNR</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

This rule uses these options in the MasterResource control group in the DAP.INI file:

```
< MasterResource >
    DefLib =
    XrfFile =
    FormLib =
```

You can also use the following INI option to tell the system where to look for your Metacode files:

```
< Metacode2PDF >
    MetacodePath =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetacodePath</td>
<td>This option specifies where the MET files are located.</td>
</tr>
</tbody>
</table>
Attachment variables

The required attachment variables are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>The user ID</td>
</tr>
</tbody>
</table>
| METACODEFILE or AFPFILE | The name of the Metacode file to load. If you omit the path, the MetacodePath option in the Metacode2PDF control group defines where the file is located.  
The name of the AFP file to load. If you omit the path, the AFPPath option in the AFP2PDF control group defines where the file is located. |
| PRTINPUTTYPE      | The name of the printer control group (PrtType:XXX) to use for your INI settings. The default printer group is XER for the Metacode printer group and AFP for the AFP printer group. |

This rule creates and destroys the MTCFORMSET DSI value.

Returns
Success or failure

Errors
This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTC0001</td>
<td>Cannot locate METACODEFILE in the attachment</td>
</tr>
<tr>
<td>MTC0010</td>
<td>DLL version mismatch, internal API failure</td>
</tr>
<tr>
<td>MTC0011</td>
<td>Attachment value METACODEFILE is empty</td>
</tr>
<tr>
<td>MTC0010</td>
<td>Print API failure, probably configured incorrectly in the INI file</td>
</tr>
</tbody>
</table>

For more information on error parameters see the HTML template ERRORS.HTM.
**MTCPrintFormset**

Use this rule to return a print output. This rule requires that the MTCFORMSET DSI variables created. Use the [MTCLoadFormset on page 273](#) rule to create this variable.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hInstance</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI.MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>Options</td>
</tr>
</tbody>
</table>

**Attachment variables**

There are no required attachment variables. If present, the system uses the following attachment variable:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrintPath</td>
<td>The full path for the output PDF file. If you omit this variable, the system uses the PrintPath option in the Attachments control group to determine the location of PDF file.</td>
</tr>
</tbody>
</table>

This rule generates a unique file name for the PDF file it creates and adds the name to the attachment as REMOTEPRINTFILE. The file name also includes path information.

This rule expects the MTCFORMSET variable be created with the DAP form set handle. It is similar to DPRPrint, but does not do any recipient filtering.

This rule can use the following control group and option in the DAP.INI file:

```ini
< Attachments >
   PrintPath =
</ Attachments >
```

**Returns**

Success or failure
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTC0017</td>
<td>Cannot locate DSI variable MTCFORMSET. Make sure you registered the MTCLoadFormset rule.</td>
</tr>
<tr>
<td>MTC0010</td>
<td>Cannot initialize the print subsystem. Check your INI file options</td>
</tr>
</tbody>
</table>

For the error parameters see the HTML template ERRORS.HTM.
Use this rule to check the required input attachment variables and INI options before starting the GenData program.

Syntax

```
_DSIXPORT DWORD _DSIAPI RPDCheckAttachments (DSIHANDLE hdsi,
    char * pszParms,
    ULONG ulMsg,
    ULONG ulOptions)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_MSG, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

```plaintext
Directory = d:\ProgIDS\RP\Mstrres\Validate\W32exe\..\..\fsiuser
UserINI = ..\..\fsiuser
```

So the full UserINI name becomes:

```plaintext
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:

```plaintext
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:

```plaintext
< RPDAttachments >
Variable = ReqType
Variable = Config
Variable = PrintBatches
Variable = ExtrFile
< IDSServer >
ExtrPath = d:\fap\mstrres\rpex1\extract\n< RPDRunRP >
Executable = d:\rel120\rps100\shippw32\gendaw32.exe
Directory = d:\fap\mstrres\rpex1\nUserINI = fsiuser
< Debug >
RPDCheckAttachments =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RPDAttachments control group</strong></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Enter the name of the variable.</td>
</tr>
<tr>
<td><strong>IDSServer control group</strong></td>
<td></td>
</tr>
<tr>
<td>ExtrPath</td>
<td>Enter the default path for the ExtrFile option.</td>
</tr>
<tr>
<td><strong>RPDRunRP control group</strong></td>
<td></td>
</tr>
<tr>
<td>Executable</td>
<td>Enter the name and path of the program you want to execute, such as d:\rpsetup\gendaw32.exe.</td>
</tr>
<tr>
<td>Directory</td>
<td>Enter the path to the master resource library, where you want to run Documaker.</td>
</tr>
<tr>
<td>UserINI</td>
<td>(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.</td>
</tr>
<tr>
<td><strong>Debug control group</strong></td>
<td></td>
</tr>
<tr>
<td>RPDCheckAttachments</td>
<td>Enter Yes to append errors to the ErrFile.</td>
</tr>
</tbody>
</table>
Returns  Success or failure

Errors  This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPD0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list at #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0004</td>
<td>Cannot add variable #VARIABLE,# to attachment at #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0007</td>
<td>File #FILENAME,# does not exists. Failed to #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0009</td>
<td>The INI option #INIOPTION,# could not be located in the group #INIGROUP,#.</td>
</tr>
</tbody>
</table>

See also  RPDCheckRPRun on page 280
RPDCreateJob on page 283
RPDDeleteFiles on page 288
RPDProcessJob on page 290
RPDRunRP on page 293
RPDSetPDFAttachmentVariables on page 298
RPDStopRPRun on page 300
**RPDCheckRPRun**

Use this rule to make sure Documaker Server is running. If Documaker Server is not running, this rule starts it.

**Syntax**

```
_DSIEXPORT DWORD _DSIAPI RPDCheckRPRun (DSIHANDLE hdsi,
    char * pszParms,
    ULONG ulMsg,
    ULONG ulOptions)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_MSG, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

To determine if Documaker Server 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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>The configuration for the master resource library (MRL). See also the DPRSetConfig rule and the setup with multiple master resource directories.</td>
</tr>
</tbody>
</table>

**Attachment outputs**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDRunProcess</td>
<td>This value is the process ID for the Documaker process.</td>
</tr>
</tbody>
</table>
INI options

You can use these INI options:

```
< RPDRunRP >
   Executable =
   Directory =
   UserINI =
< IDSServer >
   GENSEmaphoreName =
   RPDSemaphoreName =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDRunRP control group</td>
<td></td>
</tr>
<tr>
<td>Executable</td>
<td>The name and path of the program you want to execute, such as d:\rpsetup\gendaw32.exe.</td>
</tr>
<tr>
<td>Directory</td>
<td>The path to the master resource library, where you want to run Documaker.</td>
</tr>
<tr>
<td>UserINI</td>
<td>(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.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>IDSServer control group</td>
<td></td>
</tr>
<tr>
<td>GENSEmaphoreName</td>
<td>The name of the semaphore. The default is gendata.</td>
</tr>
<tr>
<td>RPDSemaphoreName</td>
<td>The name of the semaphore. The default is rpdrunrp.</td>
</tr>
<tr>
<td>MaxConfigAllowed</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug control group</td>
<td></td>
</tr>
<tr>
<td>RPDCheckRPRun</td>
<td>Enter Yes if you want errors appended to the ErrFile and the LogTrace file to record the trace.</td>
</tr>
<tr>
<td>RPDErrFile</td>
<td>Specify a name for the RPDErrfile. Include the full path</td>
</tr>
</tbody>
</table>

Returns  
Success or failure
Chapter 2
Documaker Bridge Rules

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPD0001</td>
<td>Cannot locate variable #VARIABLE,# in the attachment list at #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0004</td>
<td>Cannot add variable #VARIABLE,# to attachment at #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0008</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
<tr>
<td>RPD0009</td>
<td>The INI option #INIOPTION,# can not be located in the group #INIGROUP,#.</td>
</tr>
<tr>
<td>RPD0010</td>
<td>Cannot create DSI variable #VARIABLE,#. #LOCATION,# failed.</td>
</tr>
<tr>
<td>RPD0013</td>
<td>Cannot open the RPD error file #VARIABLE,# at #LOCATION,#.</td>
</tr>
</tbody>
</table>

See also
RPDCheckAttachments on page 277
RPDCreateJob on page 283
RPDDeleteFiles on page 288
RPDProcessJob on page 290
RPDRunRP on page 293
RPDSetPDFAttachmentVariables on page 298
RPDStopRPRun on page 300
**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**

```c
_DSIEXPORT DWORD _DSIAPI RPDCreateJob (DSIHANDLE hdsi,
    char * pszParms,
    ULONG ulMsg,
    ULONG ulOptions)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ulMsg</td>
<td>DSI_MSG, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtrFile</td>
<td>Extract file name and path. This is a required input file.</td>
</tr>
<tr>
<td>MsgFile</td>
<td>(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.</td>
</tr>
<tr>
<td>ErrFile</td>
<td>(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.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LogFile</td>
<td>(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.</td>
</tr>
<tr>
<td>DBLogFile</td>
<td>(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.</td>
</tr>
<tr>
<td>NAFile</td>
<td>(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.</td>
</tr>
<tr>
<td>POLFile</td>
<td>(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.</td>
</tr>
<tr>
<td>NewTrn</td>
<td>(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.</td>
</tr>
<tr>
<td>PrintBatchPath</td>
<td>The default path for print batches.</td>
</tr>
</tbody>
</table>
| 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):  
```xml
<PrinterInfo>
    <Printer>
        ...
    </Printer>
</PrinterInfo>
```
| 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. |
### Output DSI variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDJOB8TICKET</td>
<td>Job ticket variable. Its value is a XML document handle for the job ticket.</td>
</tr>
</tbody>
</table>

### Input DSI variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDJOB8LOG</td>
<td>Job log variable. Returns an XML document handle for the job log.</td>
</tr>
</tbody>
</table>

### Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchX</td>
<td>The name and path of the batch files. X denotes the number of batch files, from one to the maximum.</td>
</tr>
<tr>
<td>BatchX</td>
<td>The name and path of the batch files. X denotes the number of batch files, from one to the maximum.</td>
</tr>
<tr>
<td>PrinterX</td>
<td>Name and path of print batches. X denotes the number of the print batches from one to the maximum.</td>
</tr>
<tr>
<td>Results</td>
<td>Success or an error code from the IDS rules.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt; IDSServer &gt;</code></td>
<td></td>
</tr>
<tr>
<td>PrintPath</td>
<td>Used as a default path for print batches and the rest of the output files.</td>
</tr>
<tr>
<td>PrintFileCacheTime</td>
<td>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.</td>
</tr>
<tr>
<td>TextFileCacheTime</td>
<td>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.</td>
</tr>
<tr>
<td>FileExt</td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt; Printer &gt;</code></td>
<td></td>
</tr>
<tr>
<td>PrtType</td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt; RPDRunRP &gt;</code></td>
<td></td>
</tr>
<tr>
<td>BaseLocation</td>
<td>The URL to the output data directory. Your entry must be consistent with the PrintPath or other defined data path.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug control group</td>
<td>Enter Yes if you want errors appended to the ErrFile and the LogTrace file to record the trace.</td>
</tr>
</tbody>
</table>
Returns

Success or failure

Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPD0002</td>
<td>Cannot create #TAGNAME at #LOCATION.</td>
</tr>
<tr>
<td>RPD0003</td>
<td>Cannot create DSI variable #VARIABLE at #LOCATION.</td>
</tr>
<tr>
<td>RPD0004</td>
<td>Cannot add variable #VARIABLE to attachment at #LOCATION.</td>
</tr>
<tr>
<td>RPD0005</td>
<td>Cannot locate DSI variable #VARIABLE at #LOCATION.</td>
</tr>
<tr>
<td>RPD0006</td>
<td>DSI variable #VARIABLE does not contain valid data. Failed to #LOCATION.</td>
</tr>
<tr>
<td>RPD0013</td>
<td>Cannot open the RPD error file #VARIABLE at #LOCATION.</td>
</tr>
<tr>
<td>RPD0020</td>
<td>A GenData program error that is passed back via the JobLog file.</td>
</tr>
</tbody>
</table>

See also
RPDCheckAttachments on page 277
RPDCheckRPRun on page 280
RPDDeleteFiles on page 288
RPDProcessJob on page 290
RPDRunRP on page 293
RPDSetPDFAttachmentVariables on page 298
RPDStopRPRun on page 300
**RPDDDeleteFiles**

Use this rule to delete files created by the RPDRunRP rule.

**Syntax**

```c
_DSIXPORT DWORD _DSIEXPORT RPDDeleteFiles (DSIHANDLE hdsi,
    char * pszParms,
    ULONG ulMsg,
    ULONG ulOptions)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ulMsg</td>
<td>DSI_MSG, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETCODE</td>
<td>Returned code from a prior RPD rule.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>Success or failure from a prior RPD rule.</td>
</tr>
</tbody>
</table>

You have these output attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPNAME</td>
<td>A 46-byte unique name for creating temporary output files, such as .usr, .sys, and so on.</td>
</tr>
</tbody>
</table>

**INI options**

You can use these INI options with this rule:

```ini
< RPRun >
    SaveOnErrors=
    KeepAll =
```
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):<br><br>```<RPRun>
    SaveOnErrors =
</RPRun>```

Use these settings in the request INI to determine if all files should be kept:<br><br>```<RPRun>
    KeepAll =
</RPRun>```

To trigger this rule, add this line in the DOCSERV.INI file:<br><br>`function = RPDW32->RPDDeleteFiles`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| **SaveOnErrors** | When the returned error code is greater than 4 or RESULTS returns a FAILURE and if SaveOnErrors is set to True, the Delete flag is set to False 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.                                                                                                          |
**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**

```plaintext
_DSIEXPORT DWORD _DSIAPI RPDProcessJob (DSIHANDLE hdsi,
    char * pszParms,
    ULONG ulMsg,
    ULONG ulOptions)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string unsigned long</td>
</tr>
<tr>
<td>ulMsg</td>
<td>DSI_MSG, such as DSI_MSGRUNF unsigned long</td>
</tr>
<tr>
<td>ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDJobTicket</td>
<td>A job ticket variable. It returns the XML document handle for the job ticket.</td>
</tr>
</tbody>
</table>
Output files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBTICKET.XML</td>
<td>A job ticket, which is a trigger for the RP process. It contains request information and information used to update INI options.</td>
</tr>
</tbody>
</table>

Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDJobLog</td>
<td>The job log variable. Its value is an XML document handle for the job log.</td>
</tr>
</tbody>
</table>

INI options

You can use these INI options:

```xml
< RPDRunRP>
  Directory =
  < IDSServer >
    MaxWaitTime =
    SleepingTime =
    WaitForStart =
  < Debug >
    RPDProcessJob =
</ RPDRunRP>
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDRunRP control group</td>
<td></td>
</tr>
<tr>
<td>Directory</td>
<td>Enter the path where you want to load and unload the JOBTICKET.XML and JOBLOG.XML files.</td>
</tr>
<tr>
<td>IDSServer control group</td>
<td></td>
</tr>
<tr>
<td>MaxWaitTime</td>
<td>Enter, in seconds, the maximum length of time you want IDS to wait for the JOBLOG.XML file. The default is 60 seconds.</td>
</tr>
<tr>
<td>SleepingTime</td>
<td>Enter the time, in milliseconds, to specify how often IDS should check for a job ticket. The default is 1000 (1 second).</td>
</tr>
<tr>
<td>WaitForStart</td>
<td>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.</td>
</tr>
<tr>
<td>Debug control group</td>
<td></td>
</tr>
<tr>
<td>RPDProcessJob</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns

Success or failure
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPD0003</td>
<td>Cannot create the DSI variable #VARIABLE,# at #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0004</td>
<td>Cannot add the variable #VARIABLE,# to attachment at #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0005</td>
<td>Cannot locate the DSI variable #VARIABLE,# at #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0006</td>
<td>The DSI variable #VARIABLE,# does not contain valid data. Failed to #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0007</td>
<td>The file #FILENAME,# does not exist. Failed to #LOCATION,#.</td>
</tr>
<tr>
<td>RPD0008</td>
<td>The call by #LOCATION,# to API #APINAME,# failed.</td>
</tr>
<tr>
<td>RPD0009</td>
<td>The INI option #INIOPTION,# cannot be located in the group #INIGROUP,#.</td>
</tr>
<tr>
<td>RPD0111</td>
<td>Records the unexpected termination of the GenData program.</td>
</tr>
<tr>
<td>RPD0013</td>
<td>Cannot open the RPD error file #VARIABLE,# at #LOCATION,#.</td>
</tr>
</tbody>
</table>

See also

RPDCheckAttachments on page 277
RPDCheckRPRun on page 280
RPDCreateJob on page 283
RPDDeleteFiles on page 288
RPDRunRP on page 293
RPDSetPDFAttachmentVariables on page 298
RPDStopRPRun on page 300
RPDRunRP

Use this rule to run Documaker Server. 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string unsigned long</td>
</tr>
<tr>
<td>ulMsg</td>
<td>DSI_MSG, such as DSI_MSGRUNF unsigned long</td>
</tr>
<tr>
<td>ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORFILE</td>
<td>The URL of the error file created by Documaker Server. This variable only exists if there is an error or a warning. If no error file has been created, this variable will be blank.</td>
</tr>
<tr>
<td>ERRORMSG</td>
<td>This variable only exists if there was an error and will contain the error message.</td>
</tr>
<tr>
<td>RESULTS</td>
<td>Success or failure</td>
</tr>
<tr>
<td>RETCODE</td>
<td>The code returned from Documaker Server. If the error occurred before Documaker Server was called, the return code contains FAIL.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseDirectory</td>
<td>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/</td>
</tr>
<tr>
<td>BaseLocation</td>
<td>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: <a href="http://205.176.142.5./doc-data/">http://205.176.142.5./doc-data/</a> (where doc-data is the alias for d:\docserv\tempdata)</td>
</tr>
<tr>
<td>CacheTime</td>
<td>If an error file is created by the rule, it is cached for this length of time in minutes. The default is 60.</td>
</tr>
<tr>
<td>Debug</td>
<td>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.</td>
</tr>
<tr>
<td>SingleStepGenData</td>
<td>Set this to Yes to run GenData only. The default is No.</td>
</tr>
<tr>
<td>Startup</td>
<td>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\</td>
</tr>
<tr>
<td>TempRetries</td>
<td>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.</td>
</tr>
<tr>
<td>UserINI</td>
<td>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</td>
</tr>
<tr>
<td>Debug</td>
<td>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.</td>
</tr>
</tbody>
</table>
The INI file loaded by IDS 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 = db1
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:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseDirectory not specified in RPRUN section of INI</td>
<td>You need to specify the BaseDirectory in the RPRun control group.</td>
</tr>
<tr>
<td>BaseLocation not specified in RPRUN section of INI</td>
<td>You need to specify the BaseLocation in the RPRun control group.</td>
</tr>
<tr>
<td>UserINI not specified in RPRun section of INI</td>
<td>The UserINI file was not specified in RPRun section of the INI file</td>
</tr>
<tr>
<td>BaseDirectory does not exist</td>
<td>Appears if the BaseDirectory listed in the INI does not exist. (The actual BaseDirectory is displayed)</td>
</tr>
<tr>
<td>UserINI does not exist</td>
<td>Appears if the UserINI listed in the INI does not exist. (The actual UserINI file name is displayed)</td>
</tr>
<tr>
<td>Unable to locate 'Extract' Attachment variable</td>
<td>Appears when unable to locate extract attachment variable</td>
</tr>
<tr>
<td>Empty extract file specification in 'Extract' attachment variable</td>
<td>Appears when there is an empty extract file specification in the extract attachment variable</td>
</tr>
<tr>
<td>Extract file does not exist</td>
<td>Appears if the extract file specified in the attachment variable does not exist (the actual extract file name is displayed)</td>
</tr>
<tr>
<td>Unable to create temporary file</td>
<td>Appears when the attempt to create a temporary file with the new unique name was unsuccessful.</td>
</tr>
<tr>
<td>Call to CopyFiles() failed</td>
<td>Appears when there was an error copying the extract file or INI files to the BaseDirectory using the new unique name.</td>
</tr>
<tr>
<td>Call to RPDGetFsisys failed. Check The &lt;Environment&gt; FSISYSINI entry in FSIUSER</td>
<td>Appears when the FSISYS.INI file listed in the FSISYS.INI section of the FSIUSER.INI file does not exist.</td>
</tr>
<tr>
<td>GENTRAN step failed</td>
<td>Appears when the GenTrn program completes with a return code that is greater than four (4).</td>
</tr>
<tr>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GENDATA step failed</td>
<td>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.</td>
</tr>
<tr>
<td>GENPrint step failed</td>
<td>Appears when the GenPrint program completes with a return code that is greater than four (4).</td>
</tr>
<tr>
<td>Call to CopyFiles() failed.</td>
<td>Appears when the call to CopyFiles fails.</td>
</tr>
<tr>
<td>Call to RPDGetFsisys failed.</td>
<td>Check the FSISYINI option in the Environment control group in your FSIUSER.INI file.</td>
</tr>
<tr>
<td>GENTRAN step failed.</td>
<td>The GenTrn processing step failed.</td>
</tr>
<tr>
<td>GENDATA step failed.</td>
<td>The GenData processing step failed.</td>
</tr>
<tr>
<td>GENPRINT step failed.</td>
<td>The GenPrint processing step failed.</td>
</tr>
<tr>
<td>Startup path does not exist.</td>
<td>The startup path is incorrect.</td>
</tr>
<tr>
<td>Unable to create temporary file.</td>
<td>The system cannot create a temporary file.</td>
</tr>
<tr>
<td>Unknown critical error occurred.</td>
<td>Appears when there was a failure for an unknown reason.</td>
</tr>
</tbody>
</table>

See also
- RPDCheckAttachments on page 277
- RPDCheckRPRun on page 280
- RPDCreateJob on page 283
- RPDDeleteFiles on page 288
- RPDProcessJob on page 290
- RPDSetPDFAttachmentVariables on page 298
- RPDStopRPRun on page 300
**RPDSetPDFAttachmentVariables**

Use this rule to create PDF file name and URL attachment variables. This rule is run after the RPDRunRP rule.

**Syntax**

```
_DSIEXPORT DWORD _DSI_API RPDSetPDFAttachmentVariables (DSIHANDLE hdsi,
    char * pszParms,
    ULONG ulMsg,
    ULONG ulOptions)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ULONG ulMsg</td>
<td>DSI_MSG, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ULONG ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

This rule creates the FILE and URL attachment variables in the DSI_OUTPUTQUEUE for PDF files generated by Documaker Server (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:


Use this option in the request INI to specify the base location to use:

```
< RPRun >
    BaseLocation =
</ RPRun >
```

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 =
</ RPRun >
```

You can trigger this rule by adding the following line in the DOCSERV.INI file:

```
function = RPDW32->RPDSetPDFAttachmentVariables
```

**Attachment variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMPNAME</td>
<td>A unique 4-character hex name used as an output file. In PDF format.</td>
</tr>
</tbody>
</table>
Attachment outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFS</td>
<td>The number of PDF files.</td>
</tr>
<tr>
<td>PDFSX.FILE</td>
<td>The output PDF file.</td>
</tr>
<tr>
<td>PDFSX.URL</td>
<td>A complete URL.</td>
</tr>
</tbody>
</table>

Returns  Success or failure

Errors  This rule can return these messages:
- 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 277
          RPDCheckRPRun on page 280
          RPDCreateJob on page 283
          RPDDeleteFiles on page 288
          RPDProcessJob on page 290
          RPDRunRP on page 293
          RPDStopRPRun on page 300
**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

```c
_DSIXPORT DWORD _DSIAPI RPDStopRPRun (DSIHANDLE hdsi, char * pszParms, ULONG ulMsg, ULONG ulOptions)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>Pointer to rule parameter string</td>
</tr>
<tr>
<td>ulMsg</td>
<td>unsigned long, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

This rule receives the current process ID from the DSI variable RPDRunProcess and then terminates Documaker.

Attachment outputs

```
< Debug >
RPDStopRPRun =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDStopRPRun</td>
<td>Enter Yes to append errors to the ErrFile and have the LogTrace file record the trace.</td>
</tr>
</tbody>
</table>

Returns

Success or failure

See also

- RPDCheckAttachments on page 277
- RPDCheckRPRun on page 280
- RPDCreateJob on page 283
- RPDDeleteFiles on page 288
- RPDProcessJob on page 290
- RPDRunRP on page 293
- RPDSetPDFAttachmentVariables on page 298
TPDCreateFormset

Use this rule to create a PDF file from a TIFF, BMP, or JPEG file. On RUNF, this rule creates a DSIValue named TPDFORMSET and locates the stem attachment variable named TIFFNAME. For each of the stem values called NAME, the rule creates a page in PDF format.

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

### Syntax

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

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>the DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>a pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

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

Here is an example of the request type setup:

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

### Attachment variables

Here are the input attachment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG</td>
<td>This is the name of the configuration, such as <em>SAMPCO</em>.</td>
</tr>
<tr>
<td>TIFFNAME</td>
<td>This is the number of input bitmap files.</td>
</tr>
<tr>
<td>SRCTYPE</td>
<td>This is the source bitmap file type, such as <em>TIF</em>.</td>
</tr>
</tbody>
</table>
NOTE: You can have as many TIFFNAME#.NAME/TYPE variables as necessary.

The rule also tries to locate PDFNAME in the input attachment as the name of the output file. If one cannot be located, the rule generates a unique name and adds it to the output attachment as REMOTEFILENAME.

This rule uses the TIFFPATH option to locate TIFF, BMP, or JPEG files if the name of the TIFF, BMP, or JPEG file in the TIFFNAME variable does not already have a path.

```< TIFF2PDF >
   TIFFPath =
```

On RUNR message, this rule locates the TPDFORMSET value, destroys the form set and deletes the value.

This rule depends on the TPDInitRule rule being registered on the INI request.

**Returns**  Success or failure

**Errors**  This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPD0001</td>
<td>The system cannot locate DSILocateValue &amp;NAME&amp;. Make sure the TPDInit rule was executed.</td>
</tr>
<tr>
<td>TPD0002</td>
<td>The call to API &amp;APINAME&amp; failed.</td>
</tr>
<tr>
<td>TPD0003</td>
<td>The system cannot load file &amp;TIFFNAME&amp;.</td>
</tr>
<tr>
<td>TPD0004</td>
<td>The system cannot locate the stem variable &amp;NAME&amp;.</td>
</tr>
<tr>
<td>TPD0005</td>
<td>The stem variable &amp;NAME&amp; has an invalid value (0).</td>
</tr>
</tbody>
</table>
**TPDCreateOutput**

Use this rule to create a PDF output file. This rule uses the TPDFORMSET value created by the TPDLoadFormset rule. The rule tries to locate PDFNAME in the input attachment as the name of the output file. If it cannot locate PDFNAME, the rule generates a unique name and adds it to the output attachment as REMOTEFILENAME.

**Syntax**

```c
long _DSI_API TPDCreateOutput (DSIHANDLE hdsi,
     char * pszParms,
     unsigned long ulMsg,
     unsigned long ulOptions )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>the DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>a pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG???? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

**NOTE:** This rule depends on the TPDInit rule being registered on the INI request.

**Errors**

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPD0001</td>
<td>Cannot DSILocateValue &amp;NAME&amp;. Make sure the TPDInit rule was executed.</td>
</tr>
<tr>
<td>TPD0002</td>
<td>The call to API &amp;APINAME&amp; failed.</td>
</tr>
</tbody>
</table>

See also

TPDInitRule on page 306
TPDLoadFormset on page 304
TPDLoadFormset

Use this rule to load bitmap files. On the RUNF message, the rule creates a DSIValue named TPDFORMSET to hold the form set handle created by the TPDStartFormset API, locates the stem attachment variable with the name TIFFNAME, and for each of the stem values NAME, calls TPDAddPage.

Syntax

```c
long _DSI_API TPDLoadFormset (DSIHANDLE hdsi, 
    char * pszParms, 
    unsigned long ulMsg, 
    unsigned long ulOptions )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>the DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>a pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSGRUNF message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

This rule uses this INI option:

```ini
<TIFF2PDF>
    TIFFPath =
</TIFF2PDF>
```

to locate TIFF files if the name of the TIFF file in TIFFNAME variable does not have a path. The rule creates a DSI value called TPDFORMSETH that holds the form set in memory.

On the RUNR message, the rule locates the TPDFORMSET value, destroys the form set by calling TPDStopFormset, and deletes the value. It also deletes the TPDFORMSETH value.

**NOTE:** This rule depends on the TPDInit rule being registered on the INI request.

Here is an example of the server INI configuration:

```ini
[ReqType:TPD]
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCUploadAttachment
    function = tpdw32->TPDLoadFormset
    function = dprw32->DPRAddLogo,TPDFORMSETH
    function = dprw32->DPRAddText,TPDFORMSETH
    function = tpdw32->TPDCreateOutput
```
Errors

This rule can return these messages:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPD0001</td>
<td>Cannot DSILocateValue &amp;NAME&amp;. Make sure the TPDInit rule was executed.</td>
</tr>
<tr>
<td>TPD0002</td>
<td>The call to API &amp;APINAME&amp; failed.</td>
</tr>
<tr>
<td>TPD0003</td>
<td>The file named &amp;TIFFNAME&amp; cannot be loaded.</td>
</tr>
<tr>
<td>TPD0004</td>
<td>The stem variable &amp;NAME&amp; cannot be located.</td>
</tr>
<tr>
<td>TPD0005</td>
<td>The stem variable &amp;NAME&amp; has invalid value (0).</td>
</tr>
</tbody>
</table>

See also TPDInitRule on page 306
**TPDInitRule**

Use this rule to initialize the TIFF2PDF Bridge. On INIT message, this rule creates a DSIValue named TPDHANDLE which holds the handle to the TIFF2PDF Bridge.

On TERM message, this rule terminates the TIFF2PDF Bridge and deletes DSIValue TPDHANDLE.

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

Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIHANDLE hdsi</td>
<td>the DSI instance handle</td>
</tr>
<tr>
<td>char * pszParms</td>
<td>a pointer to the rule parameter string</td>
</tr>
<tr>
<td>unsigned long ulMsg</td>
<td>DSI_MSG??? message, such as DSI_MSGRUNF</td>
</tr>
<tr>
<td>unsigned long ulOptions</td>
<td>options</td>
</tr>
</tbody>
</table>

You should use this rule as an INIT rule, only on the INI request type.

Returns

Success or failure

See also

TPDLoadFormset on page 304
Appendix A

Reading Print Stream Files

Documaker Bridge provides a way to read print-ready files, such as those produced by Documerge. This appendix discusses the following topics:

- Getting AFP Resources on page 308
- Getting Metacode Resources on page 310
- Building AFP System Resources on page 312
- Building Metacode System Resources on page 314
- Creating Font Cross-reference Files on page 319
- Creating Documaker Graphics Files on page 325
- Limitations on page 327
GETTING AFP RESOURCES

The first step to prepare IDS for producing Adobe™ Portable Document Format (PDF) output from AFP archives is to get a copy of the AFP resources used to produce these archives. You will need these AFP resources:

- AFP fonts (coded font, character set, and code page files)
- AFP overlays
- AFP page segments
- Archived AFP print streams

AFP Fonts

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

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

Code page
In IBM AFP terminology, a code page file maps code points to an AFP character name in a character set file. Code page files always begin with the letter \textit{T}, such as \textit{T1DOC037}.

Character set
A character set file contains the bitmap image of each character in the character set. Character set files always begin with the letter \textit{C}, such as \textit{C0FATIN8.240} or \textit{C0FATIN8.300}. The character set file name extension (240 or 300) indicates whether the bitmap images are drawn at 240 or 300 dots per inch.

You will use these AFP fonts to create a font cross-reference (FXR) file. You must have these system resources available to produce PDF files from the archived AFP print streams.

Having these AFP fonts also lets you print the AFP archives so you can establish baselines for testing. If the AFP fonts are not installed for the AFP printer you are testing with, you will need to install the AFP fonts for that printer.

\textbf{NOTE:} Sampling the AFP fonts will also help you fine-tune the font cross-reference (FXR) file.

AFP Overlays

You must have a copy of the AFP overlays used by the archived AFP print streams. You will use these AFP overlays to produce PDF files from the archived AFP print streams.

If the AFP overlays are not installed for the AFP printer you are testing with, you will need to install the AFP overlays for that printer.
**AFP Page Segments**

You must have a copy of the AFP page segments used by the archived AFP print streams. Page segments are graphics files. You will use these AFP page segments to produce PDF files from the archived AFP print streams.

If the AFP page segments are not installed for the AFP printer you are testing with, you will need to install the AFP page segments for that printer.

**Archived AFP Print Streams**

You must also have a copy of the archived AFP print streams to produce PDF files using IDS. You can print the archived AFP print streams beforehand to establish baselines as you test.

Your next step is to build system resources, turn to Building AFP System Resources on page 312.
The first step necessary to prepare IDS to produce Adobe Portable Document Format (PDF) output from Metacode archives is to get a copy of the Metacode resources used to produce these archives. You will need these Metacode resources:

- Xerox JSL
- Metacode fonts and images
- Archived Metacode print streams

**Xerox JSL**

First get a copy of the Xerox JSLs used to print the Metacode print streams before they were archived. You will use these Xerox JSLs as you configure IDS's INI settings to read Metacode archives. If the JSL is not installed on the Metacode printer you are testing with, install the JSL on the printer and compile the JSL into a JDL on that printer.

You will also need to know which JDE was used within the JSL file to produce the archived Metacode print stream. Metacode print streams can switch to a different JDL/JDE than the JDL/JDE the printer was started with. If the archived Metacode print streams switch to a different JDL/JDE, IDS's INI settings will be based on the JDL/JDE which is switched to by the archive Metacode print streams.

Once the Xerox JSL files are installed and compiled on the printer, you can print the Metacode archives to establish baselines for testing.

**Metacode Fonts and Images**

Next, get a copy of the Xerox fonts (FNT files) and images (IMG files) used by the archived Metacode print streams. You will use these Xerox fonts and images to create a font cross-reference (FXR) file and Documaker graphics (LOG) files. You must have these system resources available to produce PDF files from the archived Metacode print streams.

The Documaker Bridge and the MRG2FAP utility let you load FRM files and IMG files referenced in the Metacode print stream being converted. The system looks for the FRM and IMG files in the directory specified by the FormLib option in the MasterResource control group. If you omit this option, the system looks in the current directory.

Having these Xerox fonts and images also lets you print the Metacode archives so you can establish baselines for testing. If the Xerox fonts and images are not installed on the Metacode printer you are testing with, you will need to install these files on that printer.

**NOTE:** Sampling the Xerox fonts and images will also help you fine-tune the font cross-reference (FXR) file.
Loading fonts directly

To handle Xerox fonts that contain multiple signatures or characters that must be printed vertically for the bitmap image to print correctly, the Documeerge Metacode loader lets you load Xerox fonts directly.

The system loads a Xerox font when the print stream references a font that is not listed in the FXR and it cannot find a logo with the same name as the Xerox font. The system loads the Xerox font from the master resource’s FontLib directory, as specified in the INI file. In addition to signature fonts, you can also use this feature to include bar code, MICR, or symbol fonts.

Using this capability slows performance and increases the size of PDF files. Do not use this capability to load all fonts if you are making PDF files—doing so causes the PDF driver to crash.

**ARCHIVED METACODE PRINT STREAMS**

You must also have a copy of the archived Metacode print streams to produce PDF files using IDS. You can print the archived Metacode print streams beforehand to establish baselines as you test.

Your next step is to build Metacode system resources, turn to Building Metacode System Resources on page 314.
To build system resources, you must modify the system initialization (INI) files used by the various Documaker applications. The INI files you will modify are listed below:

- FSISYS.INI
- FAPCOMP.INI

**SYSTEM INITIALIZATION (INI) FILES**

You must add a PrtType:AFP control group to these INI files. This control group contains the AFP options used for the archived AFP print streams.

**PrtType Control Group**

Below is an example of the PrtType:AFP control group, which contains these INI options:

```
< PrtType:AFP >
  OverlayExt = .ovr
  PageSegExt = .psg
  PaperSize = 0
</PrtType:AFP>
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OverlayExt</td>
<td>Use this option to tell the system what file extension is used by the AFP overlay file names. The default is OVL.</td>
</tr>
<tr>
<td>PageSegExt</td>
<td>Use this option to tell the system what file extension is used by the AFP page segment file names. The default is PSG.</td>
</tr>
<tr>
<td>PaperSize</td>
<td>Use this option to specify the size of the paper. Here are the most commonly-used sizes: zero (0) for US letter size (default) 1 for US legal size 2 for A4 size 3 for US executive size 4 for US ledger 98 for a custom size</td>
</tr>
</tbody>
</table>

To see a listing of all sizes, see Choosing a Paper Size on page 337.

**AFP2PDF Control Group**

Below is an example of the AFP2PDF control group which contains INI options used by the system

```
< AFP2PDF >
  AFPPath =
</AFP2PDF>
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFPPath</td>
<td>Defines the location of input AFP files.</td>
</tr>
</tbody>
</table>
**Master Resource Control Group**

Below is an example of the MasterResource control group that contains the INI options used by the system:

```
< CONFIG:AFPFiles >
  FormLib= .\n
< Configurations >
  Config = AFPFiles
< MasterResource >
  FormLib= [CONFIG:AFPFiles] FormLib =
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormLib</td>
<td>Use this option to tell the system where the resource files (AFP overlays, AFP page segments, and the IBMXREF.TBL file) are stored. If not found, the system looks in the location specified in the DEFLIB option</td>
</tr>
</tbody>
</table>
The FSISYS.INI and FAPCOMP.INI files are system initialization files used by various Documaker applications. You must add a PrtType:XER control group to these INI files. This control group will contain the Xerox Metacode options used for the archived Metacode print streams.

**PrtType Control Group**

Below is an example of the PrtType:XER control group, which contains these options:

```xml
< PrtType:XER >
  DJDEIden = A'@@@DJDE'
  DJDEOffset = 0
  DJDESkip = 8
  OutMode = BARR
  ImageOpt = No
  JDEName = DFLT
  JDLCode = NONE
  JDLData = 0,255
  JDLHost = IBMONL
  JDLName = CBA
  JDLRStack = 0,10,EQ,X'13131313131313131313' (optional)
  JDLRPage = 1,5,EQ,X'FFFF26FFFF' (optional)
  PrinterInk = Blue
  PaperSize = 0
  DefaultFont= 11010
</ PrtType:XER >
```

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

An excerpt of a JDL follows, along with an explanation of each of the PrtType:XER control group options.

**JDL example**

Here is an excerpt of a JDL. This excerpt is referenced in the control group options discussion.

```plaintext
CBA:   JDL;
T1:    TABLE    CONSTANT=X'12121212121212121212';
T2:    TABLE    CONSTANT=X'13131313131313131313131313131313';
T3:    TABLE    CONSTANT=X'FFFF26FFFF';
C1:    CRITERIA CONSTANT=(0,9,EQ,T1);
C2:    CRITERIA CONSTANT=(0,10,EQ,T2);
C3:    CRITERIA CONSTANT=(1,5,EQ,T3);
VOLUME            HOST=IBMONL;
LINE             DATA=(0,255);
IDEN             PRE=A'@@@DJDE',
                 OFF=0,
                 SKIP=8;
OFFSET           TEST=C1;
ROFFSET          TEST=C2,DELIMITER=YES,PRINT=NONE;
RPAGETEST=C3,SIDE=NUFront;
/* 8.5 x 11 job */
USA1: JDE;       /* JOB can be used in place of JDE */
OUTPUT            PAPERSIZE=USLETTER;
```
These options represent the IDEN statement of the JDL. The value of the DJDEIden setting is a string constant. The types of supported string constants are ASCII (A'string'), EBCDIC (E'string'), Character ('string'), and Hex (X'string'). These types of strings are not supported: Octal, H2, and H6. Strings containing repeat counts, embedded hex values, and upper/lower case toggles are not supported. Using the JDL sample listed earlier, the INI options should be:

- DJDEIden = A'@@@DJDE'
- DJDEOffset = 0
- DJDESkip = 8

**OutMode**

This option indicates the output format for the Metacode data stream generated by Documerge. You have these options:

Enter **BARR**, if you generate output using a Windows system and then transmit that output to a Xerox printer using BARR SPOOL hardware and software. If you choose BARR, a length byte is placed at the start and end of each Metacode record.

Enter **BARRWORD** only if records longer than 255 characters can be handled by your Xerox printer.

Enter **ELIXIR** to convert Elixir-formatted Metacode print files into FAP files.

For normalized Metacode, the system supports the standard Documerge 4-byte ISI format and the 2-byte variable (ISI 2-byte) format. Enter **MRG4** to use the Documerge 4-byte ISI format. Enter **MRG2** to indicate you want to use the 2-byte variable (ISI 2-byte) format.

Enter **PCO** if you generate output using a Windows system and then transmit that output to a Xerox printer using PCO hardware and software (from Prism). When you select PCO, a 4-byte length field is placed at the start of each Metacode record.

**NOTE**: Oracle Insurance has not completely tested the PCO interface.

Enter **JES2** for MVS environments. If you will upload output generated on a Windows system to an MVS system and then transmit the output to your printer via JES2, use OutMode = JES2.

Enter **ENTIRE** if you will transmit output generated by a Windows or UNIX system to a Xerox printer via a Sun workstation using ENTIRE/FIBER GATEWAY hardware and software (from Entire, Inc.). When you choose ENTIRE, a 2-byte length field is placed at the start of each Metacode record.
Enter **LAN4235**, if you generate output for a Xerox 4235 printer attached to a network.

Here is an example of this INI option:

```
OutMode = BARR
```

**NOTE:** This version assumes Metacode output produced by Documerge which does not correspond to any of the outmodes listed above. You must, however, still choose an outmode from those options listed previously.

### ImageOpt

Use this option to specify if the logos are saved on the Xerox printer as IMG files or as FNT files. To use IMG files, your printer must have GVG or GHO hardware installed. Also, in the JSL, you must set the Graphics option to Yes.

If you are using IMG files, set this option to Yes; otherwise set it to No. Metacode printers have a limit of 16 images on a page. Here is an example of this option:

```
ImageOpt = No
```

### JDEName

Use this option to represent the name of the job. A JDL may contain many jobs (JDEs) from which to choose. Using the JDL sample listed earlier, the Metacode job is selected using this INI setting: (This JDE must contain VOLUME CODE=NONE)

```
JDEName = META
```

### JDLCode

Use this option to represent the type of input format expected by the Xerox printer during normal operation (that is, the JDL/JDE setting used to start the printer). Character translation is performed as necessary.

The system supports EBCDIC, ASCII, or NONE, which is the same as ASCII. These formats are not supported: BCD, H2BCD, H6BCD, IBMBCD, PEBCDIC, and user-defined code translation.

Referring to the sample JSL, if the printer is normally started with STA DLFT,CBA then the JDLCode parameter must be set to CODE = EBCDIC. The INI setting must contain the value of the CODE= statement for the printer's normal operation. Here is an example of this INI option:

```
JDLCode = EBCDIC
```

### JDLData

Use this option to represent the starting position and length of the print line data within an input data record. The LINE statement contains a DATA entry which holds these values. Here is an example of this INI setting:

```
JDLData = 0,255
```

### JDLHost

Use this option to tell the system whether the printer is normally on-line or off-line. You can choose from **IBMONL** (on-line) and **IBMOS** (off-line). Using the JDL example listed earlier, this INI option should be set to:

```
JDLHost = IBMONL
```
For a Xerox print driver, specify these functions in the PrtType:XER control group:

- `OutputFunc = XEROOutput`
- `OutMetFunc = XEROutMet`
- `InitFunc = XERInit`
- `TermFunc = XERTerm`
- `Module = XERW32`

### JDLName

Use this option to represent the name of the JDL to use. Using the JDL sample listed earlier, this option should be set to:

```
JDLName = CBA
```

### JDLRStack

Use this optional INI option to represent criteria which tells the system to send an end of report condition to the printer. In the JDL example, the RSTACK statement performed a criteria test named C2. The C2 test checks a specific part of each input line against the string named T2. If the string T2 matches an input data record at position 0 for length of 10 bytes, an end of report condition is signaled. Only CONSTANT criteria using an EQ operator is supported.

**NOTE:** If the printer is alternately used for Metacode and text file print jobs, you must include the JDLRStack option. Always use JDLRStack.

Using the JDL sample listed earlier, this option should be set to:

```
JDLRStack = 0,10,EQ,X'13131313131313131313'
```

### JDLRPage

Use this optional INI option to represent the criteria which signals a jump to the front side of a new sheet to the printer. In the JDL sample listed earlier, the RPAGE statement performed a criteria test named C3. The C3 test checks a specific part of each input line against the string named T3. If the string T3 matches an input data record at position zero (0) for a length of 5 bytes, a jump to new sheet condition is signaled because of the SIDE=NUFRONT setting. Only CONSTANT criteria using an EQ operator is supported. The SIDE=NUFRONT setting in the JSL is required for JDLRPage to work properly.

**NOTE:** If the print job is likely to contain duplex pages alternating with simplex (one sided) pages, JDLRPage provides a way to leave the back sides of certain pages blank.

Using the JDL sample listed earlier, this option should be set to:

```
JDLRPage = 1,5,EQ,X'FFFF26FFFF'
```

### PrinterInk

Use this option to specify the color of ink loaded on a Xerox highlight color printer. You can set the PrinterInk option to either Blue, Red, or Green (blue is the default). This option is used with the SendColor INI option. If you set the SendColor option to Yes, you should also set the PrinterInk option. Here is an example of this INI option:

```
PrinterInk = Blue
```
PaperSize

Use this option to specify the size of the paper. Here is a list of the most commonly-chosen options. For a complete listing of all options, see Choosing a Paper Size in the Printers Reference.

<table>
<thead>
<tr>
<th>For</th>
<th>Enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>letter</td>
<td>zero (0). This is the default.</td>
</tr>
<tr>
<td>legal</td>
<td>1</td>
</tr>
<tr>
<td>A4</td>
<td>2</td>
</tr>
<tr>
<td>executive</td>
<td>3</td>
</tr>
<tr>
<td>custom</td>
<td>98</td>
</tr>
</tbody>
</table>

DefaultFont

Use this option when displaying the names of fonts which are not found in the font cross-reference (FXR) or LOGO.DAT files. The value for the DefaultFont option is a font ID which is contained in the font cross-reference (FXR) file being used.

```
< PrtType:XER >
  DefaultFont = 11010
```
Creating Font Cross-reference Files

You will need to create a font cross-reference (FXR) file using the AFP fonts referenced by the archived AFP or Metacode print streams. You can create and update font cross-reference (FXR) files using the FNTEDW32 utility.

Adding Fonts to the Font Cross-reference File

First, start the FNTEDW32 utility by entering this command, in the directory in which you installed IDS:

```
fntedw32
```

The FNTEDW32 utility’s Insert option lets you add font information to your font set (FXR file). Follow these steps to add font information:

1. When the FNTEDW32 utility starts, you see the following window:

   ![Font Cross-reference File Selection Window](image)

   Select the font cross-reference file to which you want to add fonts and click Ok. The Font List window for the font cross-reference file appears.

2. To insert fonts, click Insert in the Font List window. The Insert Fonts window appears.

   ![Insert Fonts Window](image)

   **NOTE:** The Insert button is active only if no font is selected. If the Insert button is not active, click the Deselect All button. This activates the Insert button.

3. Select AFP font or Xerox Metacode fonts as the font type you want to insert and click Ok. The Load AFP Font File window appears.
4 Select the font files you want to add. You can insert multiple fonts. If the file is in a different directory or folder, use the Drives and Folders fields to find the file. Once you select the file you want, click Ok. The selected font set is inserted in your font set.

**NOTE:** Remember that AFP coded font files begin with the letter X, such as X0DATIN8.FNT. A coded font file contains references to specific character set and specific code page. The corresponding character set and code page files should be in the same directory as the coded font file to import it. The font information imported from the AFP font will be assigned a font ID which is one greater than the largest font ID contained in the font cross-reference file (FXR).

If you get an error while inserting AFP fonts, the error may have occurred because your AFP fonts are corrupt or the files do not contain AFP fonts.

If you get an error while inserting Xerox fonts, the error may have occurred because...

- Xerox fonts are encrypted. The Documaker Bridge cannot use encrypted fonts. Xerox can take an encrypted font and provide a non-encrypted equivalent.
- Xerox fonts are corrupt or the files do not contain Xerox fonts.
CUSTOMIZING A FONT CROSS-REFERENCE FILE

While using the FNTEDW32 utility, you can use the Edit option to edit the information about fonts and printers. Highlight the font you for which you want to edit information and click Edit. This window appears:

On the Description tab, make sure the settings for the Stroke wt., Style, and Spacing fields are correct. Italic fonts should have a style of *Italic*. Bold fonts should have a stroke weight that’s greater than zero. Fixed pitch fonts should have a spacing of *fixed*. And, proportional fonts should have a spacing of *proportional*.

When you click the Printers tab, the following window appears:
The Printers tab lets you enter printer-specific information for PCL, AFP, Metacode, and PostScript printers. In the AFP (or Metacode) section of the Printers page, the Font File field should contain the name of the font file.

### AFP font file names

For AFP font files, the names begin with an X. AFP font file names are limited to eight characters. AFP font file names must be in uppercase letters and should not include an extension.

### Metacode font file names

Xerox Metacode font file names are limited to six characters. Do not enter an extension. The Rotated Font Files field should include the 90, 180, and 270 degree versions of the Portrait Xerox font file separated by semicolons, such as:

```
FNT90;FNT180;FNT270
```

### Acrobat fonts

The Adobe Acrobat Reader uses PostScript fonts instead of AFP or Metacode fonts. To make the PDF look as much like the original printed output, the AFP or Metacode fonts must be mapped to one of the standard base fonts which are always available to the Adobe Acrobat Reader. The system uses the PostScript Font Name (also called Setup Data) setting in the font cross-reference file (FXR) to specify which base font to use. The standard base fonts for Acrobat Reader are:

- Courier, Courier-Bold, Courier-Oblique, Courier-BoldOblique
- Helvetica, Helvetica-Bold, Helvetica-Oblique, Helvetica-BoldOblique
- Times-Roman, Times-Bold, Times-Italic, Times-BoldItalic
- Symbol, ZapfDingbats

### CHECKING YOUR FONT CROSS-REFERENCE FILE

Once you finish making changes to the font cross-reference file, you can use the FXRVALID utility to check a font cross-reference (FXR) file for settings which would cause problems when creating PDF files.

**NOTE:** For more information on FXRVALID and other utilities, see the Docutoolbox Reference.

The FXRVALID utility performs several checks on font IDs in the font cross-reference (FXR) file, including the following.

### Checking typefaces

This check makes sure all font IDs contain one of the following PostScript font names in the Setup Data field for PostScript printing:

<table>
<thead>
<tr>
<th>Font Name</th>
<th>Font Name</th>
<th>Font Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier</td>
<td>Helvetica-Bold</td>
<td>Symbol</td>
</tr>
<tr>
<td>Courier-Bold</td>
<td>Helvetica-Oblique</td>
<td>Univers-Medium</td>
</tr>
<tr>
<td>Courier-BoldItalic</td>
<td>Helvetica-BoldOblique</td>
<td>Univers-Bold</td>
</tr>
<tr>
<td>Courier-Oblique</td>
<td>Times-Roman</td>
<td>Univers-MediumItalic</td>
</tr>
</tbody>
</table>
Creating Font Cross-reference Files

The FXRVALID utility tells you via an error message if the FXR file contains an invalid PostScript font name or does not contain a PostScript font. The message also tells you whether a fixed or proportional font will be used in place of the invalid typeface. The PDF printer driver will make the font substitution.

**NOTE:** Font IDs have either a fixed pitch or a proportional spacing value. If font substitution is required for fixed pitch fonts, Courier is typically used. If font substitution is required for proportional fonts, Helvetica is typically used. In addition, the stroke weight and style settings of the font ID are checked to see if bold and/or italic versions of these fonts should be used.

<table>
<thead>
<tr>
<th>Checking point sizes</th>
<th>Checking the code page</th>
<th>Checking spacing</th>
<th>Checking the style</th>
</tr>
</thead>
<tbody>
<tr>
<td>This check compares the font height to the point size for each PostScript font in the FXR. A warning message appears for every font ID whose font height differs from the point size by a factor of 1/3 or greater. The utility uses the font height to determine the point size. A warning also appears if the font height equals zero.</td>
<td>A warning appears for any font IDs whose code page field is not empty or is not set to 1004.</td>
<td>This check makes sure the spacing value (fixed or proportional) of the font ID matches a PostScript font with an equivalent spacing style. If the spacing value does not match, a warning appears.</td>
<td>This check makes sure the font style (upright or italic) of the font ID matches a PostScript font with an equivalent font style. If a font ID specifies an italic style, a warning appears if the Setup Data field does not contain a PostScript font name containing the word <em>Italic</em> or <em>Oblique</em>. If a font ID specifies an upright style, a warning appears if the Setup Data field contains <em>Italic</em> or <em>Oblique</em>.</td>
</tr>
</tbody>
</table>

The FXRVALID utility tells you via an error message if the FXR file contains an invalid PostScript font name or does not contain a PostScript font. The message also tells you whether a fixed or proportional font will be used in place of the invalid typeface. The PDF printer driver will make the font substitution.

**NOTE:** Font IDs have either a fixed pitch or a proportional spacing value. If font substitution is required for fixed pitch fonts, Courier is typically used. If font substitution is required for proportional fonts, Helvetica is typically used. In addition, the stroke weight and style settings of the font ID are checked to see if bold and/or italic versions of these fonts should be used.

<table>
<thead>
<tr>
<th>Checking point sizes</th>
<th>Checking the code page</th>
<th>Checking spacing</th>
<th>Checking the style</th>
</tr>
</thead>
<tbody>
<tr>
<td>This check compares the font height to the point size for each PostScript font in the FXR. A warning message appears for every font ID whose font height differs from the point size by a factor of 1/3 or greater. The utility uses the font height to determine the point size. A warning also appears if the font height equals zero.</td>
<td>A warning appears for any font IDs whose code page field is not empty or is not set to 1004.</td>
<td>This check makes sure the spacing value (fixed or proportional) of the font ID matches a PostScript font with an equivalent spacing style. If the spacing value does not match, a warning appears.</td>
<td>This check makes sure the font style (upright or italic) of the font ID matches a PostScript font with an equivalent font style. If a font ID specifies an italic style, a warning appears if the Setup Data field does not contain a PostScript font name containing the word <em>Italic</em> or <em>Oblique</em>. If a font ID specifies an upright style, a warning appears if the Setup Data field contains <em>Italic</em> or <em>Oblique</em>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Courier-Bold Oblique</th>
<th>Times-Bold</th>
<th>Univers-Bold Italic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier-Italic</td>
<td>Times-Italic</td>
<td>ZapfDingbats</td>
</tr>
<tr>
<td>Helvetica</td>
<td>Times-Bold Italic</td>
<td></td>
</tr>
</tbody>
</table>
Checking the weight

This check makes sure the font weight (bold or normal) of the font ID matches a PostScript font with an equivalent font weight. If a font ID specifies a bold style, a message appears if the Setup Data field does not contain a PostScript font name which includes the word Bold and vice versa.

Using the FXRVALID Utility

To use this utility, enter this command:

```bash
fxrvaldw32 /I /E /G /O /R /D?
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/I</td>
<td>The name of the font cross-reference (FXR) file, omit the extension.</td>
</tr>
<tr>
<td>/E</td>
<td>(Optional) An error file name, omit the extension.</td>
</tr>
<tr>
<td>/G</td>
<td>Turns on the adding of “OTH” entry and the grouping of fonts. You can specify the grouping threshold as an error percentage. The default is zero (0). The default range is 32,127.</td>
</tr>
<tr>
<td>/O</td>
<td>(Optional) An output file name. The new FXR file contains “OTH” entries and grouping. If you omit the file name, the utility uses the input file name with an FXX extension. If you include a file name without an extension, the utility defaults to FXX.</td>
</tr>
<tr>
<td>/R</td>
<td>(Optional) Use this parameter (startchar,endchar) to specify the range of characters in width table to be checked for grouping. You can enter any integer from 0 to 255. The default value for startchar is 32 and the default value for endchar is 127. If endchar is less than startchar, the value of endchar is set to that for startchar.</td>
</tr>
<tr>
<td>/D?</td>
<td>Turns on the DownloadFont option, known as the Option field in the “OTH” entry, in every “OTH” entry. The DownloadFont option in every “OTH” entry is turned off if you omit this parameter.</td>
</tr>
</tbody>
</table>

For example, if you enter:

```bash
fxrvaldw /I=rel113sm
```

The utility checks the font cross-reference file named REL113SM.FXR and creates an error file named REL113SM.ERR which you can open in any ASCII text editor.
To optimize performance, you should create graphics (LOG) files for signature fonts, images, and logos referenced by the archived Metacode print streams. While the system can load Xerox fonts directly for signature fonts, doing so makes processing slower than if you had used logos.

You can convert a Xerox font, image, or logo into a logo using the XER2LOGW utility. For example, if you are running on a Windows computer, you would enter...

```
xer2logw /I=xfont.fnt
```

to create a logo named XFONT.LOG. You could also enter...

```
xer2logw /I=ximage.img
```

to create a logo named XIMAGE.LOG. You could also enter...

```
xer2logw /I=xlogo.lgo
```

to create a logo named XLOGO.LOG.

**CREATING A LOGO.DAT FILE**

You do not need to create logos for Xerox fonts and images which are rotated versions of the Xerox fonts and images you previously converted into system logos. Instead, you will need to create a LOGO.DAT text file for these non-portrait signature fonts or images referenced by the archived Metacode print streams.

The LOGO.DAT file should be placed in the FormLib directory. The LOGO.DAT file, which is a semicolon-delimited file, should look similar to...

```
FNT0;FNT90;FNT180;FNT270;
```

...where FNT0 is the file name for zero (0°) rotation, FNT90 is the file name for 90° rotation, FNT180 is the file name for 180° rotation, and FNT270 is the file name for 270° rotation.

**REMOVING UNWANTED TEXT AND LOGOS**

If you see text or logos when viewing a form using the Acrobat Reader that do not appear in the printed Metacode output, it is because your Metacode output contains characters not defined in the Xerox font.

To prevent this, add an INI control group whose name is the Xerox font name and specify the first and last characters defined in that font. Typically, this only affects signature and other non-text fonts.

For example, if your signature font is named QFLOGO.FNT, you would set up an INI control group with the following options:

```
< QFLogo >
FirstChar = 65
LastChar = 68
```

In this example, the first and last character code points are 65 (A) and 68 (D) respectively. Do not include the file extension (.FNT) in the group name. Do not use letters like A and D or hexadecimal numbers for the FirstChar and LastChar option settings. You must use decimal numbers.
**NOTE:** To determine the first and last characters used in a font, sample the font on the Xerox printer.

These settings only affect signature and other non-text fonts that have been converted into Documaker LOG files.

**USING THE MRG2FAP UTILITY**

You can use the MRG2FAP utility to convert a Documerge AFP or Metacode file into a FAP file. This utility also converts AFP print files created by the Documaker system into FAP files. You can then view and edit the FAP file using Documaker Studio.

The MRG2FAP utility lets you load Xerox FRM files and IMG files that are referenced in the Metacode print stream being converted. In addition, the MRG2FAP utility can produce a BPSD/Field cross-reference listing.

The system looks for the FRM and IMG files in the directory specified by the FormLib option in the MasterResource control group. If you omit this option, the system looks in the current directory.

Use the KeepBlankPages option when you are converting AFP and Xerox Documerge files into FAP files (MRG2FAP) to retain blank pages. Here is an example:

```
< PrtType:AFP > or < PrtType:XER >
KeepBlankPages = Yes
```

Normally blank pages are removed because the system assumes they are duplex back pages that are not needed. If, however, you want to retain these pages, add this option and set it to Yes. The default is No which indicates you do want to remove blank pages during a conversion.

For more information, see the Docutoolbox Reference.

**OVERLAYS AND PAGE SEGMENTS**

If the AFP print file contains references to overlays or page segments, copy the overlay or page segment files into the directory in which the AFP print file resides. Add the following options in the PrtType:AFP control group in the FSISYS.INI file to specify the file extension for overlay and page segment files.

```
< PrtType:AFP >
OverlayExt =
PageSegExt =
```
Here is a summary of the AFP, Xerox Metacode, and PDF limitations you should keep in mind:

**AFP Loader Limitations**

- The AFP loader works with AFP output produced by Docmaker applications. It assumes records are delimited by a blocking scheme similar to files produced under MVS. At the beginning of the file is a four-byte block length. A four-byte record data length follows this.

  The record data length indicates the length of the next piece of data, in this case, an AFP command. Additional record data lengths and associated data follow this until the block length is exhausted. At this point, a new block length is expected and the process repeats itself. The AFP page segment and overlay files must use the same format as the AFP print stream being converted. The AFP loader should also work on AFP print streams without the logical block and record data lengths.

- All fonts used by an AFP print stream must be found in the font cross-reference (FXR) file.

- Large print-ready files (more than 100 pages) will process slowly.

- The AFP loader cannot display charts and inline graphics. Inline graphic support only applies to the Metacode loader.

**Metacode Loader Limitations**

- The PrtType settings must match the settings used to produce the print-ready Metacode file.

- All fonts used by a Metacode print stream must be found in the font cross-reference (FXR) file or in a Docmaker graphics (LOG) file.

- Rotated text may not display properly. Short bind back pages will display upside-down. Landscape pages display sideways.

- Large print-ready files (more than 100 pages) will process slowly.

**PDF Limitations**

The system does not currently support the full set of Adobe Acrobat PDF capabilities. Here are some of the limitations.

- If the PostScript Font Name/Setup Data setting in the FXR does not match a PDF base font, the system maps these PostScript font names to PDF base font names:
  - Courier-Italic maps to Courier-Oblique
  - Courier-BoldItalic maps to Courier-BoldOblique
  - Univers-Medium maps to Helvetica
  - Univers-Bold maps to Helvetica-Bold
  - Univers-MediumItalic maps to Helvetica-Oblique
Univers-BoldItalic maps to Helvetica-BoldOblique

Finally, if the PostScript font name fails to map to a PDF base font name using the preceding rules, then fixed pitch fonts will map to Courier and proportional fonts will map to Helvetica. If a font has bold, italic, or bold and italic attributes, the Courier or Helvetica PDF base font with corresponding attributes will be used.

- Only the ANSI code page (also known as code page 1004) is supported for PDF files. Normally, this will only be an issue if your documents include international characters. If you have used AGFA fonts for printing, this should not be an issue.

- The system currently supports four standard page size in the PDF file:
  - Letter (8.5 x 11 inches)
  - Legal (8.5 x 14 inches)
  - A-4 (8.26x 11.69 inches)
  - Executive (7.25 x 10.5 inches)

Portait and landscape page orientations are available for these standard page size. The customized page size will be converted into Letter size with corresponding orientation.

- Page-at-a-Time downloading of PDF files is supported. To take advantage of this you must have an Acrobat 3.0 or higher viewer with an appropriate web browser and web server. The web browser would have to support a proposed HTTP extension for specifying byte ranges of a file to be downloaded, and the web server must support byte range downloading.

- Although Acrobat Reader supports variable fields, radio buttons, push buttons, list boxes, and hypertext links, the system does not support creation of these objects within a PDF file.

- The Metacode loader can load fonts directly. Using this capability slows performance and increases the size of PDF files. Do not use this capability to load all fonts if you are making PDF files—doing so causes the PDF Print Driver to crash.
Index

A

A4 page size
  PaperSize option 312, 318
  PDF files 328
Acrobat Reader
  logos 37
  logos and text 39
Address option 171
AFP
  AFP2PDF control group 312
  limitations 327
  location of overlays 313
  location of page segments 313
  system resources 308, 312
AFP2PDF control group 274
AFPPath INI option 274
AGFA fonts 328
ANSI code page 328
Approve option 62, 124
archive keys 205
archive module
  configuring INI control group options 16
  Documaker Bridge 1
archived Metacode print streams 311
ArchiveMem option 47
ATTACH.MSG file
  DPREmCompareXMLFiles 64
attachment variables
  finding 283
Attachments control group 195
  DPRCompareXMLFiles 64
authenticating users 23
AutorunInterval option 18

B
barcode fonts
  loading directly 311
BARR SPOOL 315
BARRWORD 315
BCD 316
bind operation 6
blank pages 35, 74
Bookmark option
  Documaker Bridge 15
bridges
  Documaker Bridge rules 30
browsers
  authenticating users 23
building system resources
  Metacode 314

cache
  DPRInit 134
CAD request type 21
CaseSensitiveKeys option 62
checking
  code pages 323
  font cross-reference files 322
  font styles 323
  font weight 324
  point sizes 323
  spacing 323
  typefaces 322
code pages
  1004 328
  Documaker Bridge 323
  support for PDF files 328
colors
  logos 38, 266
  text 40
compressing
  PDF files 15, 16
confidential data 23
Config:XXX control group
  DAP.INI file 16
  dynamic configuration 15
Control control group 195
cookies
  authenticating users 23
creating
  logo files 325

D
DAL scripts 88
DAP.INI file
  Documaker Bridge 15
  modifying 5
  PDF compression option 15
  TimeOut option 16
Data option 171
DB2 8
  communication errors 10
  retrieving form sets from 6
DBGetLastError function 10
DBTable control group 8
DDTFile option 10
Debug option 71, 163, 195, 204
decryption 23, 71
DefaultFont 318
DefaultTimeout option 22
DefLib option 195, 204
DeleteFiles option 47
dictionary
adding words 41
DIFCompareXMLFiles 65
Directory Information Tree 210
DJDEIden 315
DJDESkip 315
DLLs
version information 224
DOCCLNT.INI file
setting up 20
DOCSERV.INI file
and the DAP.INI file 15
docserv.xml file 18
Documaker
bridge 1
configuring INI control group options 16
Documaker Bridge
DAP.INI file 15
debugging rule 68
debugging rules 68
illustration 2
modifying resources 4
overview 2
rules 30
setting up the DOCCLNT.INI file 20
using 1
Documaker RP
checking 280
stopping 300
Documaker Server
Documaker Bridge 1
Documerge
converting Metacode files 326
DPA files
DPRA LoadDPA 155
DPRAAddBlankPages 35
DPRAAddLogo 37
DPRAAddText 39
DPRAAddToUserDict 41
DPRAAddWipRecord 43
DPRAApproveWipRecords 43, 45, 81
DPRArchiveFormset 47, 50, 52, 77
DPRAssignWipRecord 50
DPRAUpdateFormsetFromXML 256
DPRBatchArchive 52
DPRBuildFormset 54
DPRCheck 56
DPRCheckLogin 59
DPRCheckWipRecords 60
DPRCompareXMLFiles 64
DPRConvertible 66
DPRCreatEmailAttachment 67
and DPRA ParseRecord 181
DPRDebug 68
DPRD ecryptLogin 69
DPRDecryptValue 71
DPRDecryptValue control group 71
DPRDefaultLogin 72
DPRD elBlankPages 74
DPRDeleteFiles 76
DPRDeleteWipRecord 77
DPR DelFromUserDict 79
DPRD elMultiWipRecords 81
DPRD pw2Wip 84
DPREditUserDict 86
DPRF ap2Html 88, 89
DPRFile2Dpw 92
DPRF ilterFormsetForms 94
DPRF indTemplate 95
DPRF indWipRecordsByUser 97
DPR GenerateDefinitionFile 100
DPR GenerateSeedValue 102
DPRGetConfigList 104
DPRGetDFDInfo 106
DPRGetFormList 111
DPRGetFormsetRecips 112
DPRGetHTMLForms 114
DPRGetInitValue 115
DPRGetOneWipRecord 116
DPRGetRecipients 118
DPRGetUserList 120
DPRGetWipFormset 126
DPRGetWipList 123
Index

DPRGetWipRecipients 129
DPRIni2XML 132
DPRInit 134
DPRInitLib 135
DPRLbyCopy 136
DPRLbyDelete 138
DPRLbyGet 140
DPRLbyLock 143
DPRLbyMKCol 145
DPRLbyOptions 146
DPRLbyPropFind 147
DPRLbyPropPatch 150
DPRLbyPut 151
DPRLbyUnlock 153
DPRLoadDPA 155
DPRLoadedXML2Formset 157
DPRLoadFAPImages 158
DPRLoadImportFile 159, 184
DPRLoadXMLAttachment 161
DPRLoadXMLFormset 162
DPRLocateOneRecord 163
DPRLockWip 164
DPRLog 166
DPRLog control group 166
DPRLogin 167
DPRLoginUser 169
DPRLogVar control group 166
DPRLMail 170
    and DPRCreateAttachment 67
    and DPRLog 166
DPRLMapRecipData 172
DPRLModifyUser 174
DPRLModifyWipData 178
DPRLParseRecord 181
    and DPRLog 166
DPRLPatchLevel 180
DPRLPostDMProcess 182
DPRLPrint 184
DPRLPrintDpw 192
DPRLPrintFormset 184, 194
    and DPRRetrieveFormset 204
        DPRProcessTemplate 198
        DPRProcessTemplate 197
        DPRRenameVars 199
        DPRRetFromUserDict 200
        DPRRetrieveDPA 203
        and DPRLoadDPA 155
        DPRRetrieveFormset 204
        DPRRotateFormsetPages 207
        DPRSearch 208
        DPRSearchLDAP 210
        DPRSearchWip 217
        DPRSendFormsetXML 222
        DPRSendMultiFiles 223
        DPRSendVersion 224
        DPRSet2ImageScope 225
        DPRSetConfig 226
        DPRSetConfig rule 25
        dynamic configuration 15
        DPRSortFormsetForms 231
        DPRSpellCheck 228
        DPRTblLookUp 233
        DPRTemporaryXMLFile 232
        defined 232
        DPRTransform 243
        DPRTRC.LOG file 71, 163
        DPRUnloadExportFile 249
        DPRUnloadXMLFormset 251
        DPRUnlockWip 252
        DPRUpdateFormsetFields 255
        DPRUpdateFromMRL 253
        DPRUpdateWipRecords 258
        DPRWip2Dpw 261
        DPRWipBatchPrint 263
        DPRWipIndex2XML 267
        DPRWipTableParms 269
        DPRXMLDiff 272
        defined 272
        dummy pages 74
E

Elixir 315
email
   and DPRLog 166
   and DPRMail 170
   creating an attachment 67
Email2IDS control group 170
EmailDFD control group 170, 181
Enable_Debug_Options option 10
encrypted fonts 320
encryption 23, 71
Entire, Inc. 315
ERR request type
   rules to run 20
error messages
   Documaker Server 7
executive page size 328
export files
   unloading 249

F

FAPAddBlankPages 36, 75
FAPCOMP.INI file 314
fields
   scope 225
File option 62, 124, 127, 259
   DPRApproveWipRecords 45
filler pages 74
FirstChar option 325
FNT files 316
font cross-reference files
   checking 322
font IDs 322
FontID option 40
fonts
   AGFA 328
   in added text 40
   loading Xerox fonts directly 311
   Metacode 310
   Postscript 328
   removing unwanted characters 326
   signature fonts 325
   Xerox 320
FormFile option 10
FormLib option 61, 127, 195, 310
FRM files 310
FSISYS.INI file
   building Metacode resources 314
FSIVER utility 180
FXR file
   checking fonts 322
FXRVALID utility 322

G

GenArc program
   Documaker Bridge 1
GenData program
   checking attachments 277
   running 293
   generating unique IDs
      DOCCLNT.INI file 21
GenPrint program
   running 293
GenTrn program
   running 293
   getting Metacode resources 310

H

H2 315
H2BCD 316
H6 315
Index

H6BCD 316
hash values 23
highlight color printer 317
HTML files
creating email 67
DPRMail 170
HTML templates
DPRProcessTemplate 197

IBMBCD 316
IBMXREF.TBL file
location of 313
IDEN statement 315
ImageExt option 61, 127, 195, 204
ImageOpt option 316
images
for Metacode 310
IMG files 310, 316
import file
loading 159
INI files
DAP.INI 15
DOCCLNT.INI file 20
FAPCOMP.INI 314
FSISYS.INI 314
loading 15
modifying 5
InitFunc option 317

JD 314
JDE 310
JDEName 316
JDLCODE 316
JDLDATA 316
JDLHOST 316

JDLName 317
JDLRPage 317
JDLRStack 317
JOBLOG.XML file 7
JSL, Xerox 310, 314

K

KeepAll option 288, 289
KeepBlankPages option 326

L

landscape orientation 328
LastChar option 325
LbyLib option 10
LDAP
DPRSearchLDAP rule 210
legal page size
supported sizes 328
letter page size
supported sizes 328
Library Manager 135
using 10
limitations 327
linking to Documerge 327
log files
DPRLog 166
LOGIN.HTM
editing 4
LogoExt option 61, 127, 195, 204
LogoFile option 10
logos
DPRAddLogo 37
linking to Documerge 325
master resources
  modifying 4
MasterResource control group 195, 310
MaxTimeout option 22
MaxWaitTime option 290
MaxWIPRecords option 62, 124, 127, 259
memory
  debugging 68
Message option 171
Metacode 327
  archived print streams 311
  building system resources 314
  converting to FAP files 326
  creating graphics files 325
  file locations 273
  fonts and images 310
  getting resources 310
  loader limitations 327
  MRG2FAP utility 326
  normalized 315
  PDF limitations 328
  resources 310
  rotating pages 207
Metacode2PDF control group 273, 274
MetacodePath option 273, 274
MICR fonts
  loading directly 311
MinTimeout option 22
Module option 317
MRG2FAP utility 310, 326
MTCLoadFormset 273
  and DPRPrint 184
MTCPrintFormset 275

non-text fonts 325

Octal 315
ODBC
  connection errors 9
  disconnecting 9
OMR marks 74
Oracle DBMS 8
orientations
  supported 328
OutMetFunc option 317
OutMode 315
OutputFunc option 317
overlays
  MRG2FAP utility 326

page segments
  MRG2FAP utility 326
page sizes 328
pages
  blank 35, 74
PaperSize option 312, 318
passwords
  authenticating users 23
Path option 62, 124, 127, 171, 181, 259
  DPRApproveWipRecords 45
PCO hardware and software 315
PDF 327
Index

PDF files
- and the Metacode loader 311
- compression 16
- creation options 15
- deleting 195
- INI options 15
- limitations 327
- registering 194
- removing (TimeOut option) 16
- TIFF files 34
- TPDCreatetateFormset 301
- PDFFileCache control group 16, 195
- PEBCDIC 316
- PersistOutput option 10
- portrait orientation 328
- PrinterInk option 317
- PrintPath option 195, 204
- PRT request
  - rules to run 20
- PrtType control group
  - Metacode resources 314
  - PDF compression option 15
  - PrintFormset 195
- PrtView_WIPTable control group 269

R

Recip_Names control group 195
- DPRPrint 184
- recipient filtering 184
- Reject option 62, 124
- renaming
  - attachment variables 199
- REQTYPE
  - Documaker Bridge 20
- ReqType control group
  - DPRCompareXMLFiles 64
- request
  - queue INI settings 20
- resources
  - modifying 4
  - rotating
    - text 40
- RPDCreatetateAttachments 277
  - defined 277
- RPDCreatetateRPRun 280
  - defined 280
- RPDCreatetateJob 283
  - defined 283
- RPDCreatetateJob rule 7
- RPDeleteFiles 288
  - defined 288
- RPDPersoateJob 290
  - defined 290
- RPDRunRP 293
  - defined 293
- RPDSetaatePDFAttachmentVariables 298
  - defined 298
- RPDS stopRP Run 300
  - defined 300
- RPEX1.INI file
  - modifying 5
  - rules
    - bridge 30
    - to run on a request 20
  - RULServerJobProc rule 7

S

SaveOnErrors option 288, 289
- SCS request type 21
- security
  - authenticating users 23
- SENDBACKPAGE 232
- SendColor option
  - Metacode printers 317
- Setting 22
- setting up
  - the DAP INI file, Documaker Bridge 15
  - the DOCCLNT.INI file, Documaker Bridge 20
  - time-out values in the DOCCLNT.INI file 22
ShowErrors attachment variable 7
signature fonts 325
signatures
  fonts 325
  in Metacode print streams 311
SingleStepGenData option 293
SQL return code -805 6
Subject option 171
Summary Patch Report 180
symbol fonts 311

T

T1 option 171
T2 option 171
templates
  DPRFindTemplate 95
TermFunc option 317
TerSub paragraph 89
TIFF2PDF bridge
  initializing 306
  TIFF2PDF control group 302
  TPDInitRule 306
TIFFNAME variable 301
TIFFPath option 302
TimeOut option 195
PDF files 16
timeout settings 22
time-out values
  DOCCLNT.INI file 22
  global 22
TPDCreateFormset 301
TPDCreateOutput 303
TPDFORMSET 301
TPDInitRule 306
  and TPDCreateFormset 302
TPDLoadFormset 304
trace files 10
transaction-based INI loading 15

U

UserDictPath option 201
using
  Documaker Bridge 1

V

V2 files
  DPRUnloadExportFile 249
version information
  DLLs 224

W

WIP
  assigning users 50
  DPRApproveWipRecords 45
  DPRCheckWipRecords 60
  DPRGetWipFormset 126
  DPRGetWipList 123
  DPRGetWipRecipients 129
  DPRUpdateWipRecords 258
WIP option 62, 124
WIPEdit plug-in 116
Word
  creating a WDF file 100
  workspace definition file 100

X

xBase libraries
  using 10
XER2LOGW utility 325
Index

Xerox
  4235 printer 316
  additional printer settings 317
  fonts 311, 320
  highlight color printer 317
  JSL 310
  signature fonts 325
XML
  DPRProcessTemplate 197
XML files
  DPRCompareXMLFiles 64
  DPRTemporaryFile 232
  DPRUnloadExportFile 249
  DPRXMLDiff 272
XML2Attach control group 95, 170
XML2Body control group 95, 170
XRFExt option 195
XRFFile option 195