Report Mods Framework

USER GUIDE

Monitoring Output Queues:

Once the RMF is loaded to your system and the RPTMOD subsystem is active, any output queue can be monitored by the RMF server by entering the following one-time command:

CHGOUTQ OUTQ(MYLIB/MYOUTQ) DTAQ(*server-library*/RPTMOD)

- Where *server-library* represents the library where the RMF server was installed (RPTMOD by default).
- It is recommended that only standalone output queues (no writer attached) are selected for use.
- Any number of output queues can be monitored concurrently.

Any spooled file that arrives on these queues in RDY status, or changes to RDY status from another status (such as releasing from HLD status) will trigger an entry to be sent to the RPTMOD data queue. The RMF server continuously polls this data queue for entries. When an entry arrives:

The spooled file's information is retrieved and compared against the rules defined in the RPTMOD table, which cross references spooled files and the associated actions (if any) that should be taken on it, in the order of the sequence number. Ties in the sequence number are broken by the identity column, which represents the order in which the rules were entered.

Rules Table (RPTMOD):

IDENTIFIERS		
RMSEQN	INTEGER	Sequence number. Determines order in which matching rules are executed.
RMFILE	CHAR(10)	Spooled file name (or blank or *ALL for wildcard).
RMAPGM	CHAR(10)	Application program that created the spooled file (or blank or *ALL)
RMSRCQ	CHAR(10)	Source output queue (or blank or *ALL).
RMUSRP	CHAR(10)	User profile that created the spooled file (or blank or *ALL).
RMFTYP	CHAR(10)	Form type of the original spooled file (or blank or *ALL).
RMENAB	CHAR(1)	Rule enabled (Y/N).
RMDELT	CHAR(1)	Delete original spooled file afterward (Y/N).
RMTYPE	CHAR(1)	Transform type: • S = Spooled file • P = PDF document • E = E-mail with PDF attachment • F = FTP directly to PDF capable printer
VALID FOR TYPES "S" AND "P"		
RMMPGM	CHAR(10)	Content modification program name.
RMMLIB	CHAR(10)	Content modification program library.
RMRECP	CHAR(30)	Recipient text to be printed at bottom of page.
RMPGLO	NUMERIC(3)	Page length override
RMPGWO	NUMERIC(3)	Page width override
RMLPIO	NUMERIC(2)	Lines per inch override
RMCPIO	NUMERIC(2)	Characters per inch override
RMOVFO	NUMERIC(2)	Overflow line override
RMOVNM	CHAR(10)	Front side overlay name (or *NONE to remove original overlay).
RMOVLB	CHAR(10)	Front side overlay library.
VALID FOR TYPE "S" ONLY		
RMTRGQ	CHAR(10)	Target output queue (Type "S" only)
RMCOPY	NUMERIC(2)	Number of copies (Type "S" only)
RMDRAW	CHAR(1)	Drawer number override (Type "S" only).
VALID FOR TYPES "P", "E" AND "F"		
RMPATH	VARCHAR(100)	IFS path for PDF files. The directory path must exist.
VALID FOR TYPE "E" ONLY		
RMMAIL	VARCHAR(64)	E-Mail address override. If not specified, SMTP alias (if exists) or combination of spooled file owner user profile and system domain is used (Type "E" only).
VALID FOR TYPE "F" ONLY		
RMFTPH	CHAR(50)	FTP host name or IP address of PDF capable printer (Type "F" only).
RMFTPS	CHAR(10)	FTP script member name in file QFTPSRC in server library (Type "F" only).
MISCELLANEOUS		
RMUNIQ	INTEGER	Auto-generated identity column. Used as Sequence Number tie-breaker.

The Identifiers section contains fields which determine whether a spooled file is selected for a specific action, based on its name (RMFILE), the program which created it (RMAPGM), the output queue in which it exists (RMSRCQ), the user who created it (RMUSRP) and its form type (RMFTYP). Any or all of these fields can be specified in a given rule. A value of blank or *ALL indicates a wild card. The Enabled flag (RMENAB) is a kill switch for the rule – only rules with a value of "Y" are honored.

If the Delete After flag (RMDELT) on the rule is set to "Y", the original spooled file is deleted. Use this setting with care because a spooled file may have multiple rule matches, especially when wild cards are used (blanks or *ALL) in the Identifiers section. If the spooled file is deleted prior to completion of subsequent matching rules, those rules will fail.

The Transform Type (RMTYPE) indicates what will be done with the spooled file:

- **S** = Another spooled file will be produced from the original, either as-is, or with overrides to its attributes such as Lines Per Inch (RMLPIO), Overflow Line (RMOVFO) or Front Side Overlay (RMOVNM), amongst others. If a Target Queue (RMTRGQ) is specified, the new spooled file will be created in that queue. Optionally, if alteration to the actual spooled file content is desired, a Modification Program (RMMPGM) may be specified. This program must conform to certain standards as outlined in the associated RPG code sample (MODSAMPLE), downloadable from the GitHub site.

- **P** = A Portable Document Format (PDF) file will be produced from the original, either as-is, or with overrides to its attributes such as Lines Per Inch (RMLPIO), Overflow Line (RMOVFO) or Front Side Overlay (RMOVNM), amongst others. If a target IFS directory path (RMPATH) is specified, the PDF document is placed in that location with a name that contains the original spooled file name, user profile and creation data. Optionally, if alteration to the actual spooled file content is desired, a Modification Program (RMMPGM) may be specified. This program must conform to certain standards as outlined in the associated RPG code sample (MODSAMPLE), downloadable from the GitHub site. Also, if custom naming of the PDF is desired, the Modification Program provides an avenue to construct a name (usually with content from the spooled file) which is passed back to the server program automatically.

- **E** = A portable Document Format (PDF) file will be produced from the original, in an as-is condition in the directory specified by RMPATH, then e-mailed to a specified address (RMMAIL) as an attachment. The naming convention of the attachment is generic based on spooled file, user and creation data. If RMMAIL is not specified, the server attempts to retrieve the user's SMTP alias from the System Distribution Directory. If no alias exists, a default e-mail address is constructed from the user profile and the system's domain name. Or, the e-mail address can be determined by a modification program and passed back to the server through the communications area. The IBM i Mail Server Framework and SMTP server must be active and functional. On most systems, the SMTP attributes include the IP address of a mail gateway. This gateway must allow mail relays from the IBM i system.

- **F** = A portable Document Format (PDF) file will be produced from the original, in an as-is condition in the directory specified by RMPATH, then transmitted directly to a PDF capable printer using the FTP protocol. Column RMFTPS must specify the name of a source physical file member in file QFTPSRC in the server library, with the necessary FTP commands to authenticate to the printer. The actual PUT command is dynamic and will be built by the server for each spooled file.

Modifying content:

Once upon a time, the RMF's original purpose was to allow programmers to modify spooled file content without altering 3rd party source code, or perhaps maybe they didn't even have the source code to begin with. It was part of a larger strategy known as "Mods Externalization" (or later "Decustomization") that we often spoke about at user groups and conferences. Avoiding mods to 3rd party software keeps the software eligible for vendor support and eliminates the risk of unintended consequences.

When a Modification Program is specified in an RPTMOD rule, the RMF server takes all character based content and loads it into memory, in the form of a two dimensional array of Pages (maximum 1500) by Lines (maximum 66). Each cell in the array is a 132 byte character field, representing all text characters in a printed line. The RMF uses *PRTCTL format in its extraction process so that lines that were ordinarily skipped over (and subsequently invisible to the DSPSPLF command), come to life in the array as a blank cell, which will allow us to move content to those lines if desired.

All modification programs should use the following (or similar) procedure interface:

```
dcl-pi *n;
  PagPtr pointer const;
  NumPgs int(10) const;
  CommArea like(RptModCom);
end-pr;
```

Likewise, the modification programs should define the RPTMODCOM communications data area:

```
dcl-ds RptModCom;
  FormType char(10);
  PDFName  varchar(64);
  EmailAdr varchar(64);
end-ds;
```

…and also the Page Grid structure.

```
dcl-ds Pages dim(1500) qualified based(PagPtr);
  Lines char(132) dim(66);
end-ds;
```

The above snippets can also be placed in /COPY members for easy redeployment. For simpler packaging and redistribution of the RMF, they were included in the server program source.

The **Pages** data structure is actually built by the server program (RPTMOD). Its location in memory is passed as the **PagPtr** parameter (hence the BASED keyword). Any character on any line on any page can be accessed and/or altered as you see fit. The second parameter, **NumPgs**, can be used as a looping limit when scanning through the entire spooled file. **CommArea** is a catchall container for various information being passed back and forth between the server program and the modification handler. This can be especially useful when the mod handler extracts an identifier from the spooled file content, such as a company number or customer number, and wishes to include that identifier in the PDF document name. Or it might make more sense to determine the recipient's e-mail address here in the mod handler, based on content in the spooled file instead of specifying an address in the rule record or letting the server build the address dynamically. You can add your own subfields to the **RptModCom** structure in both the server and handler programs.

Modifying actual spooled file content is nothing more than a search and replace mission, using ordinary string functions. It is predicated on two strategies:

- Knowing where content will be by its absolute line/column on the page.
- Searching for a unique text pattern on a line, or a common text pattern on multiple lines.

For example, say you wanted to add a column to your Open Order report to show the user ID who entered the order. This involves:

- Adding appropriate headers on each page, consistent with other columns.
    - Say lines 3 & 4 for the sake of argument.
- Determining which browsing lines are detail lines.
    - Inspection revealed that the order number lies in position 7-11, consumes the full five positions, always starts with a letter and is always preceded and followed by at least one blank space.
- Extracting pertinent information from the line to use as a database key to retrieve other information.
    - Chain to the Order Header using the order number as key.
- Inserting that information into the line.
    - RPG's %subst function can be used (on the left side of the expression) to overlay whitespace on the line and %replace can be used to shift content left or right.
- Optionally, moving other content on the line to make space for the new information.
    - Also best handled by %subst and %replace. If moving column detail, don't forget to move the column titles as well.

Assuming you have added the code snippets previously documented, the modification handler could be as simple as this. Remember the server program writes the memory grid back out to another spooled file.

Note – Assume that EntryUser is a column in the OrdHdr table.

```
dcl-f OrdHdr keyed;
dcl-s PagIdx int(5);
dcl-s LinIdx int(5);
dcl-s OrdKey like(OrdNum);

For PagIdx = 1 to NumPgs;
      %subst(Pages(PagIdx).Lines(3) : 123 : 5) = 'Entry';
      %subst(Pages(PagIdx).Lines(4) : 123 : 4) = 'User';
      For LinIdx = 1 to %elem(Pages.Lines);
            If %subst(Pages(PagIdx).Lines(LinIdx) : 6 : 1)  = ' ' and
               %subst(Pages(PagIdx).Lines(LinIdx) : 12 : 1) = ' ' and
               %subst(Pages(PagIdx).Lines(LinIdx) : 7 : 1) >= 'A' and
               %subst(Pages(PagIdx).Lines(LinIdx) : 7 : 1) <= 'Z' and
               %scan(' ' : %subst(Pages(PagIdx).Lines(LinIdx) : 7 : 5)) = 0;
                  OrdKey = %subst(Pages(PagIdx).Lines(LinIdx) : 7 : 5);
                  Chain Ordkey OrdHdrRec;
                  If %found(OrdHdr);
                        %subst(Pages(PagIdx).Lines(LinIdx) : 123 : 10) = EntryUser;
                  Endif;
            Endif;
      Endfor;
Endfor;

*inlr = *on;
```

Examples of RPTMOD rules:

The following rule would select all spooled files called QSYSPRT with form type ACK produced by program ORD115 that landed in output queue STORE15HLD, modify their content with handler program ORD115M01, which is always in the users' library list, and produce 2 hardcopies of the finished product on output queue STORE15P1. Afterwards it will delete the original spooled file:

RMFILE = QSYSPRT	RMMPGM = ORD115M01
RMAPGM = ORD115	RMMLIB = *LIBL
RMSRCQ = STORE15HLD	RMTRGQ = STORE15P1
RMFTYP = ACK	RMCOPY = 2
RMTYPE = S	
RMENAB = Y	
RMDELT = Y	

Of course, output queue STORE15HLD would need to specify data queue RPTMOD in the server library.

Now, say for example you wanted each copy to have a different text label on them for distribution…perhaps one for the customer and one for the merchant. This would require two separate rules:

- The first, similar to the above, but with RMCOPY = 1, RMDELT = N and RMRECP = "Customer Copy".
- The second, similar to the above, but with a higher sequence number than the first, plus RMCOPY = 1 and RMRECP = "Merchant Copy". This rule will have RMDELT = Y because it is executed last.

The following rule would select all spooled files for user FLINTSTONE that landed in output queue FREDF, turn them into a PDF and place them into a directory with a NetServer share that Fred has a drive letter mapped to. He is paperless, but being loyal to the company, he likes to have the Slate watermark on all his reports.

RMSRCQ = FREDF	RMOVNM = SLATEWMARK
RMUSRP = FLINTSTONE	RMOVLB = SLATEOVL
RMTYPE = P	RMPATH = /home/FLINTSTONE/Reports
RMENAB = Y	
RMDELT = Y	

Of course, output queue FREDF would need to specify data queue RPTMOD in the server library.

Rules can also be cascaded. For example, you wish to customize LPI and CPI settings on a report, then e-mail it automatically to a user. This would require two rules:

- A type "S" rule to apply the RMLPIO and RMCPIO values, with an RMTRGQ (target output queue) value that is also being monitored by the RMF (by virtue of its Data Queue setting).
- A type "E" rule to pick that spooled file from that output queue and specify an RMMAIL value (or blank for the SMTP alias or default user/domain combination).

When cascading rules, the following are VERY important:

- Only the last rule (by sequence number) should specify RMDELT = Y.
- Avoid matching RMSRCQ and RMTRGQ values in the same type "S" rule. This could cause an infinite loop if the rules are not properly differentiated.

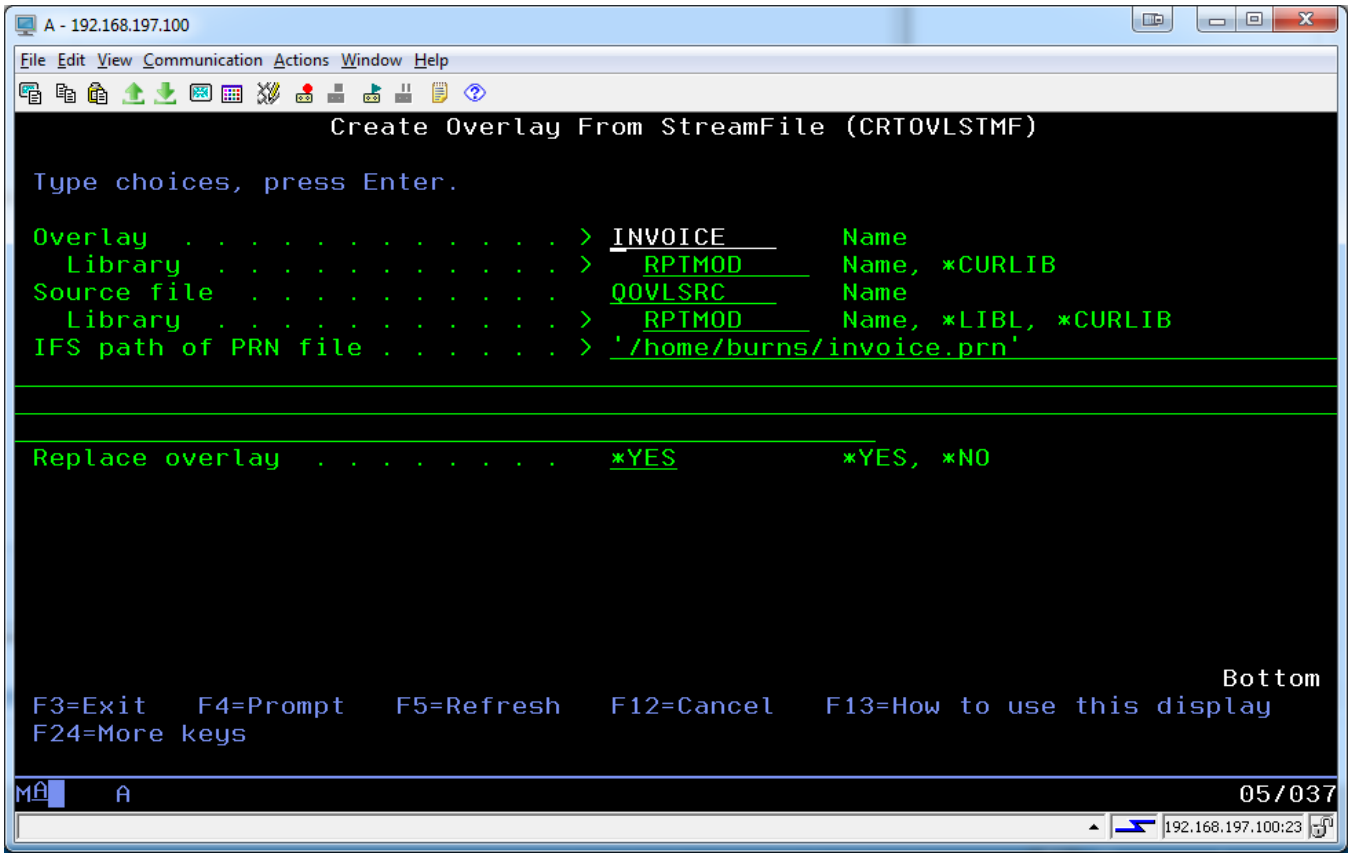
Dummy output queues (no associated writer) are never a bad thing, especially when the RMDELT value is used to clean up after ourselves. When deploying the RMF, consider using dummy queues as user defaults and letting the RMF rules handle report distribution.

Simple overlay creation:

Overlays are a great alternative to purchasing preprinted forms or ponying up for expensive commercial forms processing software. They allow spooled file text and graphical content to coexist on the same document. For years, this has been made possible through a series of printer drives provided by IBM with the IBM i Access for Windows product. These drivers allow users to create stylish headings, grids and backgrounds with popular Microsoft Windows desktop applications such as Word and Excel.

Instead of printing to a physical printer, a virtual AFP device driver redirects the output to a file containing source code for overlay object creation on IBM i. Unfortunately, this has proven to be a cumbersome multi-step process over the years. What's worse is, many of the examples available on the Web still utilize the now obsolete Document Library System (QDLS, also known as QDOC) in the transfer process. There is a more effective way, using the Root File System, although it is still multiple steps and still somewhat cumbersome. Plus, one wrong move and the overlay will fail to render properly.

Necessity being the mother of invention, introducing the Create Overlay from Stream File command (CRTOVLSTMF). This handy little gadget streamlines the process and reduces the number of failure points.



Specify:

- The qualified name of the overlay you wish to create.
- The qualified name of the physical file which will receive the overlay source in a member of the same name as the overlay. A popular standard name is QOVLSRC. A copy of this file is created in the RMF server library for your convenience.
- The full IFS path to a PRN file created by the IBM Generic AFP driver, from a Windows application. For detailed instruction on how to install/configure this driver, see the following document: http://www-01.ibm.com/support/docview.wss?uid=nas8N1018625
- Whether to replace the overlay if it exists. If *YES is specified, the overlay source is also replaced.

How the PRN file gets to the IFS is up to you. The document lists multiple possibilities, including FTP and NetServer. I recommend creating a NetServer share to a dedicated IFS directory created strictly for this purpose, and also creating a drive letter mapping in Windows to that share. It will make your life easier if you update your overlays regularly.

There are several validations performed on this command, including whether the source file exists and whether the Replace value is specified correctly when conflicting objects exist.