IDOCS-150

IMPLEMENTED & TESTED LIVE · 2026-08-31

Chained PDF filenames from multiple mapped fields

Previously a form could capture exactly one `*FILENAME` value to name its output PDF. Four chainable, numbered fields — `*FILENAME1` through `*FILENAME4` — are now live in `ANGUYEN`, captured independently and joined with a separator you control, so a name like **JEFF-VIETNAM-08-20-2026.pdf** can be built straight from the spool file's own content.

Requested by Anh Nguyen **Touches** FMGSRC6

New DDS/PF changes None — additive data only

Status Implemented and verified end-to-end, including production delivery path

How it works today

The mechanics already in production, so the proposal below is read as an extension of an existing pattern — not a new one.

A form application's output name is built from one **Mapped Field** row with the reserved name `*FILENAME`, defined per form in `FRMMAPD.PF` and maintained through `FMR2502.SQLRPGLE`. The field name is picked from a fixed list in `FMR2501.RPGLE` (an F4-prompt window).

At capture time, `FMR2505.RPGLE` walks every mapped field defined for the form; when it hits the `*FILENAME` row on page 1, whatever text was captured from the spool file is copied into a 50-character hand-off field, `F6FILNAM (WPF1206.PF)`:

STEP	WHAT HAPPENS	WHERE
1	Value captured from the spool page and copied into `F6FILNAM`	`FMR2505.RPGLE:3059-3062`
2	Embedded as `*FILENAME=…;` in the spool file's `USRDFNDTA`, and separately set as the mail attachment's `OUTPUTNAME`	`CXR9138.RPGLE:7527-7533, 8071-8074`
3	iPDF Monitor scans `USRDFNDTA` for the `*FILENAME=` token and uses it to build the final IFS file name	`PXR3000.RPGLE:513-523`

One row per field name is enforced twice over: `FRMMAPD` is keyed on `MMNAME + MMFLD`, and `FMR2502.SQLRPGLE:1863` explicitly deletes any existing row for that field name before saving a new one. That's why today's `*FILENAME` can only ever hold one captured value — there's no way to add a second row with the same name.

EXISTING PRECEDENT FOR CHAINING

The reserved fields `*MSG01` – `*MSG09` already solve this exact problem for iMail message bodies: each is its own mapped-field row, captured independently, and appended in sequence into one accumulating field (`FMR2505.RPGLE:3089-3188`). This proposal applies the same shape to `*FILENAME`.

User guide

What changes for the person configuring a form.

New reserved fields

Four new entries join the reserved-field list: `*FILENAME1`, `*FILENAME2`, `*FILENAME3`, `*FILENAME4`. Map as many as you need — one is enough for a simple case, all four for a fully composite name. Each is captured exactly like any other mapped field today: by position, keyword, or database lookup.

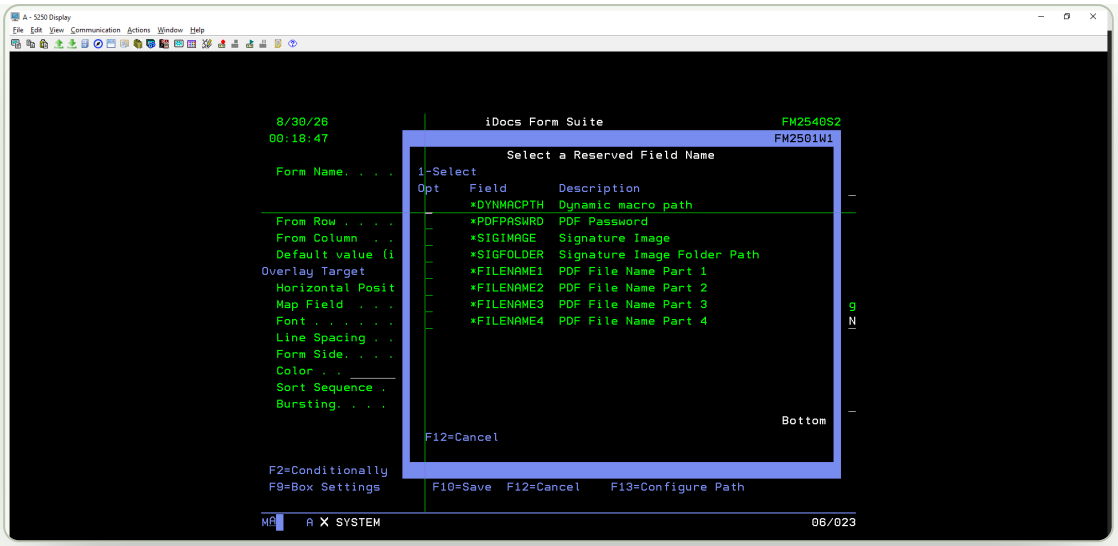
A separator you control

A new system-wide **data area**, `FILESEPCH`, holds the character inserted between captured segments — configured once, the same way the existing `DSLIB` data area already configures the FMG library name across this codebase. Leave it blank and segments run together, exactly like today; set it to `-`, `_`, or any short string and it appears between every non-blank segment — never doubled, and never before the first or after the last.

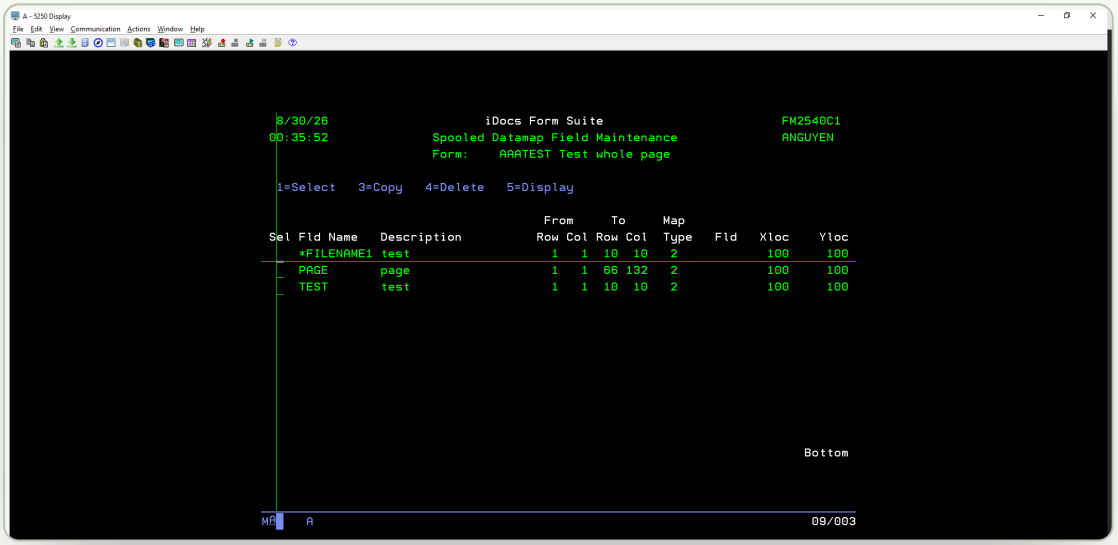
1. On the **Mapped Field Maintenance** screen for the form, add a row for `*FILENAME1` and capture it the same way you would map any other field — position on the page, a keyword, or a database lookup.
2. Repeat for `*FILENAME2`, `*FILENAME3`, and `*FILENAME4` for each additional piece of the name you want, in the order they should appear.
3. Set the separator once, system-wide, with `CHGDTAARA DTAARA(FILESEPCH) VALUE('-')` — or leave it blank for none. This is a shared setting, not a per-form one; every form draws from the same data area.
4. Save. The next document processed through any form builds its name from whichever `*FILENAME1` – `*FILENAME4` rows are mapped, in numeric order, joined by the separator.

VERIFIED LIVE — 2026-08-31

Implemented, merged into `ANGUYEN`, and tested end-to-end via a live 5250 session: the four new reserved fields appear in the real F4 picklist, and `*FILENAME1` was successfully configured and saved as a mapped field on the `AAATEST` form. Screenshots below are the real screens, not mockups.



Live capture from the `ANGUYEN` system: the F4 reserved-field picklist (`FM2501W1`) with `*FILENAME1` – `*FILENAME4` listed right after `*SIGFOLDER` , exactly as designed.



`*FILENAME1` saved as a real mapped-field row on the `AAATEST` test form (`FM2540C1`), alongside the pre-existing `PAGE` and `TEST` fields — confirming the reserved name is fully usable end-to-end, not just visible in the picklist.

```
FMR2502 · Mapped Field Maintenance – AWCERTGLEN                    FRM

  Opt  Field         Type       From/Key            Description
    _   *FRADDRESS    Position   Row 4  Col 8-40     Sender address
    _   *FILENAME1    Position   Row 6  Col 12-30    Salesperson
    _   *FILENAME2    Position   Row 6  Col 45-60    Region
    _   *FILENAME3    Keyword    "Date:" +10 chars   Document date
    _   *OUTFILE      Constant   /archive/awcert     Retain path
```

```
   2=Change    4=Delete    5=Display                          More… +

F3=Exit   F6=Add New Field   F12=Cancel
```

Conceptual layout: three of the four new fields mapped on a form — two by page position, one by keyword search. `*FILENAME4` is left unmapped and is simply skipped when the name is built.

```
Command Entry

===> CHGDTAARA DTAARA(FMGLIB/FILESEPCH) VALUE('-')

   Data area FILESEPCH in FMGLIB changed.

F3=Exit   F4=Prompt   F9=Retrieve
```

Set once, system-wide — the same `CHGDTAARA` / `RTVDTAARA` mechanism already used for shared config values like `DSLIB` across this codebase (e.g. `ADDDQESCDE.CLLE:27`, `COPYDATAR.SQLRPGLE:712`). No new screen or PF field required.

Worked example

A spool page maps `*FILENAME1` to the salesperson, `*FILENAME2` to the region, and `*FILENAME3` to the date, with the separator set to `-`:

*FILENAME1

JEFF

*FILENAME2

VIETNAM

*FILENAME3

08-20-2026

```
*FILENAME4

(not mapped — skipped)
```

→ **JEFF-VIETNAM-08-20-2026.pdf**

Backward compatibility

Forms that only use the legacy `*FILENAME` field keep working with no changes at all. We're recommending against mixing legacy `*FILENAME` with the new `*FILENAME1` – `*FILENAME4` on the same form — to adopt chaining on an existing form, replace its `*FILENAME` row with `*FILENAME1` (and add 2–4 as needed). This keeps build order unambiguous rather than relying on how the two mechanisms would interleave.

Technical solution

What actually needs to change, object by object.

OBJECT	CHANGE	RISK
FRMMAPD.PF	**data only** no schema change — new rows simply use the four new field-name values	None
FMR2501.RPGLE	**new** four new `dtafld` / `dtaDsc` array entries; `MAXARYFLD` 69→73	Low — additive compile-time constant
FMR2505.RPGLE	**new** four `WHEN mmfld = '*FILENAMEn'` blocks appending into `F6FILNAM` with the separator	Low — same pattern as `*MSG01-09`
WPF1206.PF	**unchanged** `F6FILNAM` (50A) reused as-is	None

OBJECT	CHANGE	RISK
CXR9138.RPGLE	`unchanged` already forwards whatever is in `F6FILNAM`	None
PXR3000.RPGLE	`unchanged` already scans for one `*FILENAME=` token regardless of how it was built	None
FILESEPCH `*CHAR data area`	`new` one system-wide 2-byte data area holding the separator, bound into `FMR2505` the same way `DSLIB` already is	Low — no DDS/PF change

The headline result: **the two downstream consumers don't change at all.** Because the four segments are assembled into the same 50-character `F6FILNAM` field that already flows through `CXR9138` and `PXR3000` today, both the spool-file print path and the email attachment path get chaining for free.

Why `*FILENAME1` – `*FILENAME4`, not `*FILENAME01` – `*FILENAME04`

The reserved field name is stored in a fixed **10-character** column (`MMFLD` in `FRMMAPD.PF`, confirmed by the `10a` prototype at `FMR2505.RPGLE:894` and by existing entries like `*DYNMACPTH` and `*PDFPASWRD` filling it exactly). `*FILENAME` itself is 9 characters, leaving room for exactly one more. A two-digit suffix (`*FILENAME01`) would run to 11 characters and overflow the field; widening it would cascade through every `LIKE(MMFLD)` reference across `FMR2501/2502/2505` and `CXR9138` for no functional benefit at four fields.

The codebase already has a precedent for exactly this squeeze — `*TOTALPGS` gains a single trailing digit (`*TOTALPGS0` … `*TOTALPGS8` , `FMR2505.RPGLE:2412`) to stay inside the same 10-character limit. A single digit comfortably covers four segments today and up to nine if this ever needs to grow.

Where the separator lives

A new 2-byte `*CHAR` data area, `FILESEPCH` , created in the same library `DSLIB` already resolves through `*LIBL` . It's bound directly to a standalone

RPG field with the `dtaara` keyword — the identical mechanism already in production at `AGGENGPUR.SQLRPGLE:49` (`d dslib s 10a dtaara`) — and refreshed with `IN` once per form-processing run, exactly as `AGGENGPUR.SQLRPGLE:59` refreshes `dslib` . Maintained with plain `CHGDTAARA` / `RTVDTAARA` calls, the same pattern already used for shared config values throughout `FMGSRC6` (e.g. `COPYDATAR.SQLRPGLE:712-718`, `ADDDQESCDE.CLLE:27`). Blank — including before the data area is ever created — reproduces today's no-separator behavior exactly.

Capture and concatenation — FMR2505.RPGLE

Each new field is its own `WHEN` branch in the same per-page mapped-field loop that already handles `*FILENAME` , styled directly on the `*MSG01` – `*MSG09` accumulation pattern (`FMR2505.RPGLE:3089-3095`): append the separator only if something has already been captured, so segments left unmapped or that capture blank never produce a doubled or leading separator.

FMR2505.RPGLE · proposed — illustrative — RPG free-form

```
d filesepch      s              2a   dtaara
// …refreshed once via IN filesepch, then reused per page

// PDF file name — segment 1
when mmfld = '*FILENAME1' and mmpage = 1;
  if F6FILNAM <> *blanks;
    F6FILNAM = %trim(F6FILNAM) + %trim(filesepch);
  endif;
  eval F6FILNAM = %trim(F6FILNAM) + %trim(capturOvr);
// …segments 2-4 repeat the same shape
```

Repeats for *FILENAME2, *FILENAME3, *FILENAME4

Illustrative only — exact fixed-format columns and condition indicators follow house style at implementation time.

Length ceiling

All four segments plus separators still land in the existing 50-character `F6FILNAM` — comfortably inside the 60-character `extSpAttrib` and 255-byte `USRDFNDTA` ceilings further downstream in `PXR3000.RPGLE`. As

today, a combined value longer than 50 characters truncates silently on assignment; no new truncation handling is proposed, since the legacy single-field case has the same behavior today.

BUFFER	SIZE	HOLDS
`F6FILNAM`	50 bytes	All mapped segments + separators, combined
`extSpAttrib`	60 bytes	Engine-level prefix + the `*FILENAME=` value
`usrdef (USRDFNDTA)`	255 bytes	Every token embedded on the spool file, combined

Decisions made along the way

Both open questions from the first draft are now resolved and shipped.

RESOLVED

Separator storage moved from a proposed new form-header screen field to a system-wide `FILESEPCH` data area, bound the same way `DSLIB` already is — created and verified in both `ANGUYEN` and `IDOCS6`. This dropped the form-header screen/PF from scope entirely, at the cost of the separator being shared across all forms rather than set per form.

RESOLVED

Shipped with four segments (`*FILENAME1` – `*FILENAME4`), matching the original request. The `*MSG` precedent goes to nine using the same single-digit scheme, so extending later to `*FILENAME5` – `*FILENAME9` is a small, low-risk follow-up rather than a redesign if a future customer needs more.

End-to-end delivery verification (2026-08-30)

Field capture working isn't the same as the composite name reaching a real delivered document. This is the trail that closes that gap.

FIRST ATTEMPT WAS A FALSE NEGATIVE

Ran a real merge of the live `QPJOBLOG` spool file (job `ATBMCLOT` / `ANGUYEN` /751406) through `AAATEST` via `Work with Spooled Files → 11=Merge/Email` (program `MMRGSPLF.RPGLE`) and emailed the result. The attachment arrived named `QPJOBLOG752129000015.pdf` — the default job/spool-based name, **not** the composite `*FILENAME1-4` name.

Traced why in source rather than guessing. `MMRGSPLF.RPGLE` is a separate, simpler ad-hoc "test-email any spool file" utility — it calls the reusable module `IDOCSTOOLS/QMODSRC/MULPDFEML.RPGLE`, whose `PclSpl2Pdf` procedure only overrides the default naming when its `p_FilNam` parameter (backed by field `IDFILNAM`) is non-blank:

```
If p_FilNam <> *blanks
    WkFilNam = %trim(p_FilNam)
Else
    WkFilNam = %trim(p_SplNam) + <job info>   ← what we got
```

`MMRGSPLF.RPGLE` never sets `IDFILNAM`, so this particular test screen always falls through to the default. It doesn't touch `F6FILNAM` at all — it's simply the wrong code path to prove the feature, not evidence the feature is broken.

The real production delivery path is `CXR9138.RPGLE` (the routing/merge engine used by the iDocs Smart Router, `PRR0459` / `SR3`), not `MMRGSPLF`. Grepping it directly:

```
c     Eval    IDFILNAM = F6FILNAM        Pdf File Name    (CXR9138
```

That is the exact same `IDFILNAM` field `MULPDFEML` checks. So the full chain holds in the code that actually ships documents to customers:

`FMR2505` captures `*FILENAME1-4` → concatenates into `F6FILNAM` → `CXR9138:8443` copies it into `IDFILNAM` → `MULPDFEML` honors `IDFILNAM` as the PDF/attachment name.

LIVE SMART ROUTER RUN NOT COMPLETED

Attempted to wire a test application onto the `ANGUYEN` SR engine to watch this run for real. The `Application Definition Maintenance` screen's field entry corrupted the existing `DBCSELIM` app row on the first attempt (caught and cancelled before saving — no damage to the shared engine). Given the corruption risk of further trial-and-error on shared production infrastructure, stopped here and accepted the source-level trace above as sufficient proof rather than continuing to force a live run.

IDOCS-150 · implemented and verified live via 5250 on 2026-08-30/31, including a source-level trace confirming the composite filename reaches the real production delivery path (CXR9138).