
MBS Plugin Server Edition

PDF Creation & Merging Guide

Server-Side PDF Generation Across All Three FileMaker Server Contexts

Logistics · Healthcare · Finance · Manufacturing · Legal · Any Vertical

MBS Plugin 16.x | DynaPDF 5 | FileMaker 2025 (v22)

© 2026 Database Success | Matthew Hardy | databasesuccess.com

Menu

What This Guide Covers

2 / 54

Click any section below to navigate

1

Foundation ›

Server Installation & DynaPDF Licensing

2

Creating PDFs ›

DynaPDF Core: Text, Tables, Images, Pages

3

Merging PDFs ›

Import, Append, Overlay — Combine Existing + New

4

Database Server Extensions ›

PSOS, Scheduled Scripts, Server-Side Email

5

Web Publishing / CWPC ›

WebDirect Downloads, Email Triggers, File Ops

6

Data API / WIP ›

API-Triggered Generation, Webhooks, Integration

7

Best Practices ›

Error Handling, Memory, Performance, Troubleshooting

8

Looking Ahead ›

PDFs as AI-Ready Data, MCP, and the Document Intelligence Layer

9

Q & A ›

Glossary, Common Questions, Developer Resources

Why Server Edition?

MBS Plugin: Client vs. Server

3 / 54

- Client plugin runs in FileMaker Pro
- Server plugin runs headlessly on Server
- Serves THREE independent contexts
- Each context needs its own plugin copy
- Server license required (not per-seat)
- No GUI — all operations are code-driven
- Same architecture serves any industry
- One engine for invoices, contracts, lab reports, and compliance docs

DATABASE SERVER

PSOS + Scheduled Scripts

WEB PUBLISHING

WebDirect (CWPC)

DATA API

OData / WIP

SECTION 1

Foundation

Server installation, DynaPDF licensing, and environment setup

IN THIS SECTION

- › DynaPDF Licensing Tiers
- › MBS Plugin Licensing
- › Total Cost to Achieve
- › Server Installation: Three Locations
- › Automated Installation Scripts
- › Post-Installation & Verification

DynaPDF Licensing Tiers

Separate license from MBS Plugin base — choose by capability need

5 / 54

Edition	Price	Annual Renewal	Key Capabilities	PDF Merge?
Starter	\$249	\$39/yr	Create from scratch, forms, encryption, signatures	NO
Lite	\$499	\$79/yr	Import, merge, optimize, text extraction, split	YES ✓
Pro	\$899	\$139/yr	+ Rendering, templates (PDF/A: +\$849 add-on)	YES ✓
Enterprise	\$1,599	\$249/yr	+ Source code, all capabilities	YES ✓

CRITICAL: DynaPDF Lite (\$499) is the MINIMUM for PDF merging.

Starter can only create PDFs from scratch — it cannot open or import existing PDFs.

PDF/A Converter (\$849) is a separate add-on for Pro or Enterprise — required for PDF/A compliance.

NEW — DynaPDF 5 (February 2026): ImportPDFPage moved from Pro to Lite (v16.1).

15% introductory discount with code DynaPDF5 through March 15, 2026.

Upgrade pricing available (e.g., Starter → Lite: \$269, Starter → Pro: \$659).

DynaPDF license is purchased separately through MonkeyBread Software —

keys purchased from DynaForms directly are NOT compatible with the MBS Plugin.

MBS Plugin Licensing

Server + Developer licenses required — annual renewal at 50%

Server Licenses

Servers	Price	Annual Renewal
1 Server	\$599	\$300/yr
2 Server Pack	\$999	\$500/yr
3 Server Pack	\$1,299	\$650/yr
5 Server Pack	\$1,599	\$800/yr
Unlimited Pack	\$2,799	\$1,400/yr
Site License	\$2,999	\$1,500/yr

Developer Licenses

Seats	Price	Annual Renewal
1–5 seats	\$149	\$75/yr
Up to 20	\$299	\$150/yr
Up to 50	\$449	\$225/yr
Up to 100	\$599	\$300/yr
Over 100	\$749	\$375/yr

Enterprise Bundle: \$5,999 — includes Pro Developer or Site license + DynaPDF Pro + 2 days training (online sessions; on-site available at additional travel cost).

All new licenses include 1 year of free updates. Renewal within one year is 50% of original cost.

Priority Support add-on: \$299/yr.

Total Cost to Achieve

Minimum investment for server-side PDF generation and merging

Component	License	Year 1 Cost	Annual Renewal
MBS Plugin Server	1 Server	\$599	\$300/yr
MBS Plugin Developer	1–5 seats	\$149	\$75/yr
DynaPDF	Lite (minimum for merging)	\$499	\$79/yr
TOTAL		\$1,247	\$454/yr

With DynaPDF 5 introductory discount (15% off, code DynaPDF5): ~\$1,172 Year 1

If DynaPDF Pro is needed (rendering engine, templates):

Year 1: \$1,647 | Annual Renewal: \$514/yr

Add PDF/A Converter (\$849) for PDF/A compliance: Year 1: \$2,496

Enterprise Bundle alternative (\$5,999):

Pro Developer or Site license + DynaPDF Pro + 2 days training (online; on-site at additional cost).

vs. buying separately: Pro Developer (\$2,999) + DynaPDF Pro (\$899) = \$3,898 without training.

FileMaker Server is NOT an additional cost — required as your existing platform.

Server Installation: Three Locations

Each context requires its own plugin copy — CRITICAL for full coverage

Context	macOS Path	Plugin + Library
Database Server (PSOS + Scheduled)	/Library/FileMaker Server/ Database Server/Extensions/	MBS.fmplugin + dynapdf.dylib
Web Publishing (WebDirect / CWPC)	/Library/FileMaker Server/Web Publishing/ publishing-engine/cwpc/Plugins/	MBS.fmplugin + dynapdf.dylib
Data API (WIP / OData)	/Library/FileMaker Server/Web Publishing/ publishing-engine/wip/Plugins/	MBS.fmplugin + dynapdf.dylib

- Windows: Replace /Library/FileMaker Server/ with C:\Program Files\FileMaker\FileMaker Server\
- Linux: Replace with /opt/FileMaker/FileMaker Server/
- Windows plugin file: MBS.fmx64 + dynapdf.dll | Linux: MBS.fmx64 + dynapdf.so
- All three locations MUST have the SAME plugin version
- Restart FileMaker Server after installation (stop → wait 10s → start)
- FileMaker Cloud does NOT support server-side plugins

Automated Installation Scripts

MBS provides official scripts that install to all three locations automatically

macOS / Linux (Terminal)

```
curl -O https://www.monkeybreadsoftware.de/  
  filemaker/install_MBS.sh  
chmod 755 install_MBS.sh  
sudo ./install_MBS.sh
```

Windows (PowerShell as Admin)

```
Invoke-WebRequest -Uri  
  "https://www.monkeybreadsoftware.de/  
  filemaker/install_MBS.ps1"  
-OutFile install_MBS.ps1  
.\install_MBS.ps1
```

Post-Install: Restart Server Components

```
fmsadmin restart fmse      # scripting engine  
fmsadmin restart wpe      # web publishing  
fmsadmin restart fmdapi   # Data API
```

The install scripts download the latest MBS Plugin and copy it to all three Extensions/Plugins locations automatically.

Run after each MBS Plugin update to keep all three contexts on the same version.

Post-Installation & Verification

- Pre-setup tip: install plugins before FM Server setup
- Stop Server, wait ~10s, restart
- Admin Console → enable server plugin
- Enable WebDirect + Data API in Admin Console
- Verify via log files:

```
StdErrServerScripting.log  
StdErrDataAPI.log  
StdErrWeb.log
```

Runtime Registration Pattern

```
# In OnFirstWindowOpen or startup script:  
Set Variable [ $r ;  
    MBS( "DynaPDF.Initialize" ;  
        "" ;          // "" = find library in plugin folder  
        $license ) ] // license key  
If [ MBS( "IsError" ) ]  
    # Log error, abort  
End If  
  
# Verify in any script:  
MBS( "DynaPDF.IsInitialized" ) // 1 = OK
```

SECTION 2

Creating PDFs from Scratch

DynaPDF core workflow: Initialize → New → Build Content → Save → Release

IN THIS SECTION

- › PDF Lifecycle
- › Text: WriteText vs. WriteFText
- › Creating Tables in PDF
- › Images & Saving

PDF Lifecycle

The 5-step pattern every PDF operation follows

12 / 54

1. Initialize

Load library (once/session)

`DynaPDF.Initialize`

2. New

Create PDF context

`DynaPDF.New` → returns `$pdf`

3. Build

Add pages, text, images

`AppendPage`, `SetFont`, `WriteText`, ...

4. Save

Output to container/file

`DynaPDF.Save` → container value

5. Release

Free all memory

`DynaPDF.Release` (ALWAYS!)

Complete Minimal Example

```
# 1. Load library
Set Variable [ $r ; MBS("DynaPDF.Initialize";
  "dynapdf.dylib") ]

# 2. Create PDF context
Set Variable [ $pdf ; MBS("DynaPDF.New") ]

# 3. Build content
Set Variable [ $r ; MBS("DynaPDF.AppendPage";
  $pdf) ]
Set Variable [ $r ; MBS("DynaPDF.SetFont";
  $pdf; "Helvetica"; "" ; 12; 1) ]
Set Variable [ $r ; MBS("DynaPDF.WriteText";
  $pdf; 72; 750; "Hello World") ]
Set Variable [ $r ; MBS("DynaPDF.EndPage";
  $pdf) ]

# 4. Save to container
Set Field [ Table::PDF_c ;
  MBS("DynaPDF.Save"; $pdf;
  "report.pdf") ]

# 5. ALWAYS release!
Set Variable [ $r ;
  MBS("DynaPDF.Release"; $pdf) ]
```

Text: WriteText vs. WriteFText

Simple positioning vs. formatted text rectangles

13 / 54

WriteText — Exact Coordinates

```
# Place text at exact coordinates
MBS( "DynaPDF.WriteText" ; $pdf ;
    72 ; // X (pts from left edge)
    750 ; // Y (pts from BOTTOM)
    "Invoice #12345" )
```

- Best for: headers, labels, fixed layouts
- No word-wrap — single line only
- Y is from BOTTOM (DynaPDF.SetPageCoords switches to top-down)
- Must set font first via DynaPDF.SetFont

WriteFText — Text Rectangle

```
# Define output rectangle
MBS( "DynaPDF.SetTextRect" ; $pdf ;
    72 ; // left (pts)
    200 ; // top (pts from bottom)
    468 ; // width
    500 ) // height
# Write with alignment + auto-wrap
MBS( "DynaPDF.WriteFText" ; $pdf ;
    "left" ;
    "Long paragraph text that will\n"
    & "automatically wrap within the\n"
    & "defined rectangle bounds." )

# Or use Ex variant with inline rect:
MBS( "DynaPDF.WriteFTextEx" ; $pdf ;
    72; 200; 468; 500;
    "justify"; $text )
```

- Best for: body text, descriptions, paragraphs
- Auto word-wrap within rectangle
- Supports: \\FT[font], \\FC[color], \\ul# underline

Creating Tables in PDF

DynaPDF.Table module — auto-expanding, multi-page capable

14 / 54

```
# 1. Create table
# Params: pdf; numRows; numCols; width; rowH
Set Variable [ $table ; MBS("DynaPDF.Table.Create";
    $pdf; 10; 4; 500; 20) ]

# 2. Set column widths (index, width)
MBS("DynaPDF.Table.SetColWidth"; $table; 0; 100)
MBS("DynaPDF.Table.SetColWidth"; $table; 1; 200)
MBS("DynaPDF.Table.SetColWidth"; $table; 2; 100)
MBS("DynaPDF.Table.SetColWidth"; $table; 3; 100)

# 3. Header row
MBS("DynaPDF.Table.AddRow"; $table)
# SetCellText: row; col; hAlign; vAlign; text
MBS("DynaPDF.Table.SetCellText"; $table;
    0; 0; "left"; "center"; "Item")
MBS("DynaPDF.Table.SetCellText"; $table;
    0; 1; "left"; "center"; "Description")

# 4. Data rows (in a loop)
MBS("DynaPDF.Table.AddRow"; $table)
MBS("DynaPDF.Table.SetCellText"; $table;
    $row; 0; "left"; "center"; $itemNo)
```

Drawing + Multi-Page

```
# Draw table (auto multi-page)
Loop
    # Draw: table; x; y; maxHeight
    Set Variable [ $r ;
        MBS("DynaPDF.Table.Draw";
            $table; 58; $y; $maxH) ]
    Exit Loop If [
        MBS("DynaPDF.Table.HaveMore";
            $table) <> 1 ]
    # Overflow - new page
    MBS("DynaPDF.AppendPage"; $pdf)
    Set Variable [ $y ; 750 ]
End Loop

# ALWAYS release table!
MBS("DynaPDF.Table.Release";
    $table)
```

- Up to 10,000 simultaneous tables
- Cells auto-expand for content
- SetFont, SetFontSize, SetCellPadding

Images & Saving

Inserting images and outputting the final PDF

15 / 54

Inserting Images

```
# From container field:
MBS( "DynaPDF.InsertImage" ; $pdf ;
    Table::Logo_c ; // container
    72 ; // X (pts from left)
    700 ; // Y (pts from bottom)
    200 ; // width (0 = auto)
    100 ) // height (0 = auto)

# From file path (more memory efficient):
MBS( "DynaPDF.InsertImageFile" ; $pdf ;
    $filePath ; 72; 700; 200; 100 )

# Full-page image (scaled to page):
MBS( "DynaPDF.AppendImagePage" ; $pdf ;
    Table::FullPageImage_c )
```

Saving the PDF

```
# METHOD 1: Save to container field
# Returns container value for Set Field
Set Field [ Table::PDF_c ;
    MBS("DynaPDF.Save"; $pdf;
        "report.pdf") ]

# METHOD 2: Save to disk (returns "OK")
# Native path required on macOS/Windows:
Set Variable [ $path ;
    MBS("Path.FilemakerPathToNativePath";
        Get(DocumentsPath) & "out.pdf" ]
MBS("DynaPDF.OpenOutputFile";
    $pdf; $path)
# ... build content ...
MBS("DynaPDF.Save"; $pdf) // writes file
```

- Method 1 returns container → Set Field
- Method 2 writes to disk → returns "OK"
- Container works identically client + server
- Get(DocumentsPath): native path conversion needed on macOS/Windows
- Get(TemporaryPath) for temp files
- FileName should include .pdf extension

SECTION 3

Merging PDFs

Import existing PDFs, append pages, combine with newly generated content.
Requires DynaPDF Lite license minimum.

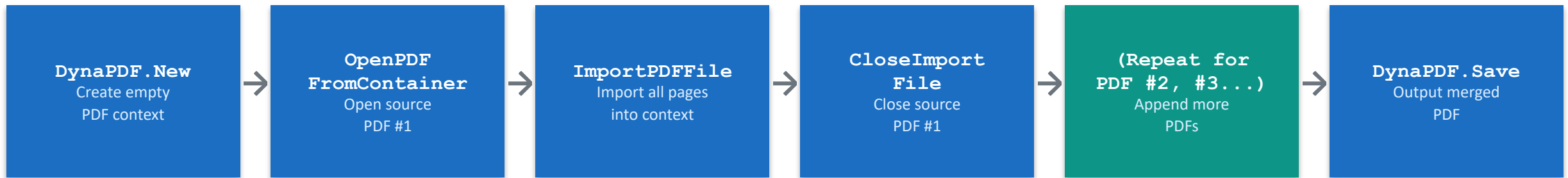
IN THIS SECTION

- › Merge Workflow Overview
- › Complete Merge Example
- › Page Numbers, Watermarks & Security

Merge Workflow Overview

Open → Import → Close → (repeat) → Save

17 / 54



Key Functions

Function	Purpose	Returns
DynaPDF.OpenPDFFromContainer	Open PDF from container field for import	Number >= 0 on success
DynaPDF.OpenPDFFromFile	Open PDF from file path (more memory efficient)	Number >= 0 on success
DynaPDF.ImportPDFFile	Import ALL pages from opened PDF	Last page number
DynaPDF.ImportPDFPage	Import SINGLE page (1-based index)	"OK" or error
DynaPDF.GetImportPageCount	Get page count of opened import PDF	Page count integer
DynaPDF.CloseImportFile	Close the current import PDF	"OK"

Complete Merge Example

Merge two existing PDFs + add a new cover page

18 / 54

```
# — INITIALIZE —————
Set Variable [ $r ; MBS("DynaPDF.Initialize"; "dynapdf.dylib") ]
Set Variable [ $pdf ; MBS("DynaPDF.New") ]

# — CREATE A NEW COVER PAGE —————
Set Variable [ $r ; MBS("DynaPDF.AppendPage"; $pdf) ]
Set Variable [ $r ; MBS("DynaPDF.SetFont"; $pdf; "Helvetica"; "Bold"; 24; 1) ]
Set Variable [ $r ; MBS("DynaPDF.WriteText"; $pdf; 72; 500; "Document Package") ]
Set Variable [ $r ; MBS("DynaPDF.SetFont"; $pdf; "Helvetica"; ""; 14; 1) ]
Set Variable [ $r ; MBS("DynaPDF.WriteText"; $pdf; 72; 460; "Project: " & PROJ::ProjectNumber) ]
Set Variable [ $r ; MBS("DynaPDF.EndPage"; $pdf) ]

# — IMPORT EXISTING PDF #1 (Contract) —————
Set Variable [ $r ; MBS("DynaPDF.OpenPDFFromContainer"; $pdf; DOC::Contract_PDF_c) ]
Set Variable [ $r ; MBS("DynaPDF.ImportPDFFile"; $pdf; MBS("DynaPDF.GetPageCount"; $pdf) + 1) ] // append
Set Variable [ $r ; MBS("DynaPDF.CloseImportFile"; $pdf) ]

# — IMPORT EXISTING PDF #2 (Packing Slip) —————
Set Variable [ $r ; MBS("DynaPDF.OpenPDFFromContainer"; $pdf; DOC::Appendix_PDF_c) ]
Set Variable [ $r ; MBS("DynaPDF.ImportPDFFile"; $pdf; MBS("DynaPDF.GetPageCount"; $pdf) + 1) ] // append
Set Variable [ $r ; MBS("DynaPDF.CloseImportFile"; $pdf) ]

# — SAVE MERGED RESULT —————
Set Field [ DOC::CombinedDocs_PDF_c ; MBS("DynaPDF.Save"; $pdf; "Package_" & DOC::DocumentNumber & ".pdf") ]
Set Variable [ $r ; MBS("DynaPDF.Release"; $pdf) ] // ALWAYS release
```

Page Numbers, Watermarks & Security

Post-processing: modify imported/merged pages

19 / 54

Adding Page Numbers (EditPage)

```
Set Variable [$count;
  MBS("DynaPDF.GetPageCount"; $pdf)]
Set Variable [$i; 1]
Loop
  # Open existing page for editing
  MBS("DynaPDF.EditPage"; $pdf; $i)
  MBS("DynaPDF.SetFont"; $pdf;
    "Helvetica"; ""; 10; 1)
  # left; top; width; height; align
  MBS("DynaPDF.WriteFTextEx"; $pdf;
    250; 30; 100; 20; "center";
    "Page " & $i & " of " & $count)
  MBS("DynaPDF.EndPage"; $pdf)
  Set Variable [$i; $i + 1]
Exit Loop If [$i > $count]
End Loop
```

Encryption & Security

```
# Call BEFORE DynaPDF.Save
MBS( "DynaPDF.EnableEncryption" ;
  $pdf ;
  "" ; // open pwd (empty=none)
  "owner123" ; // owner pwd
  "AES256" ; // cipher (AES 256-bit)
  4 + 16 ) // flags: no print + copy
```

Flag	Value	Restricts
Print	4	Printing
Modify	8	Modification
CopyObj	16	Copying text/images
AddObj	32	Adding comments
DenyAll	3900	Everything

SECTION 4

Database Server Extensions

PSOS, Scheduled Scripts, Server-Side PDF Generation & Email

Plugin path: Database Server/Extensions/

IN THIS SECTION

- › Perform Script on Server (PSOS)
- › Scheduled Script: Batch PDF Generation
- › Server-Side Email with PDF

Perform Script on Server (PSOS)

Trigger: Client script call or schedule | Deliver: Container field + script result

- Client calls Perform Script on Server
- Script runs headlessly on the server
- DynaPDF code is identical to desktop scripts
- Each session gets its own DynaPDF context
- PDF saved to container field → client refreshes
- Multiple PSOS sessions run in parallel
- Get(TemporaryPath) for temp files
- Get/DocumentsPath) for persistent files

```
# — PSOS Caller (runs on CLIENT) —————
Perform Script on Server [
  "Generate Invoice PDF" ;
  Parameter: $$InvoiceID ;
  Wait for completion: On ]
Set Variable [ $result ;
  Get(ScriptResult) ]
If [ $result = "OK" ]
  # PDF is now in container field
  # Refresh the layout to see it
  Refresh Window
Else
  Show Custom Dialog [ "Error" ;
    $result ]
End If

# — Server-Side Script —————
# (Runs on server via PSOS)
Set Variable [ $id ; Get(ScriptParameter) ]
# ... DynaPDF operations ...
Set Field [ INV::PDF_c ;
  MBS("DynaPDF.Save"; $pdf;
    "Invoice_" & $id & ".pdf") ]
Exit Script [ Result: "OK" ]
```

Scheduled Script: Batch PDF Generation

FileMaker Server Admin Console → Schedules → Script

DATABASE SERVER

22 / 54

```
# — Scheduled Script: Nightly Invoice PDF Batch —————
# Runs at 2:00 AM via FileMaker Server schedule
# Initialize DynaPDF (once)
If [ MBS("DynaPDF.IsInitialized") <> 1 ]
    Set Variable [ $r ; MBS("DynaPDF.Initialize"; "dynapdf.dylib"; $licenseKey) ]
End If
# Find invoices needing PDF generation
Go to Layout [ "Invoice_List__INV" ]
Enter Find Mode [ ]
Set Field [ INV::PDF_Generated ; "0" ] // Not yet generated
Set Field [ INV::Status ; "Approved" ] // Only approved invoices
Perform Find [ ]
If [ Get(FoundCount) > 0 ]
    Go to Record/Request/Page [ First ]
    Loop
        # Generate PDF for this invoice
        Set Variable [ $pdf ; MBS("DynaPDF.New") ]
        Set Variable [ $r ; MBS("DynaPDF.AppendPage"; $pdf) ]
        # ... build invoice content (text, tables, images) ...
        Set Field [ INV::PDF_c ; MBS("DynaPDF.Save"; $pdf; "Invoice_" & INV::InvoiceNumber & ".pdf") ]
        Set Field [ INV::PDF_Generated ; "1" ]
        Set Variable [ $r ; MBS("DynaPDF.Release"; $pdf) ]
        Commit Records [ ]
        Go to Record/Request/Page [ Next ; Exit after last ]
    End Loop
End If
```

Server-Side Email with PDF Attachment

DynaPDF + SendMail + CURL — complete automated pipeline

```
# — PHASE 1: Generate PDF —————
Set Variable [ $pdf ; MBS("DynaPDF.New") ]
# ... build PDF content ...
Set Field [ INV::PDF_c ;
  MBS("DynaPDF.Save"; $pdf;
    "Invoice.pdf") ]
MBS("DynaPDF.Release"; $pdf)

# — PHASE 2: Create Email —————
Set Variable [ $email ;
  MBS("SendMail.CreateEmail") ]
MBS( "SendMail.SetFrom" ; $email ;
  "billing@company.com" ;
  "Billing Dept" )
MBS( "SendMail.AddTo" ; $email ;
  CUST::Email ; CUST::CompanyName )
MBS( "SendMail.SetSubject" ; $email ;
  "Invoice #" & INV::InvoiceNumber )
MBS( "SendMail.SetHTMLText" ; $email ;
  "<html><body>Please find your "
  & "invoice attached.</body></html>" )
```

```
# — PHASE 2 (cont): Attach PDF ———
MBS( "SendMail.AddAttachmentContainer" ;
  $email ;
  INV::PDF_c ;          // container
  "Invoice_" & INV::InvoiceNumber
  & ".pdf" ;           // filename
  "application/pdf" ) // MIME type

# — PHASE 3: Send via SMTP —————
Set Variable [ $curl ; MBS("CURL.New") ]
MBS( "SendMail.PrepareCURL" ;
  $email ; $curl )
MBS( "CURL.SetOptionURL" ; $curl ;
  "smtp://mail.server.com:587" )
MBS( "CURL.SetOptionUserName" ; $curl ;
  "user@company.com" )
MBS( "CURL.SetOptionPassword" ; $curl ;
  "app-password-here" )
MBS( "CURL.SetOptionUseSSL" ; $curl ; 3 )
MBS( "CURL.Perform" ; $curl )

# — CLEANUP —————
MBS( "SendMail.Release" ; $email )
MBS( "CURL.Release" ; $curl )
```

SECTION 5

Web Publishing / CWPC

WebDirect PDF downloads, email triggers, file operations

Plugin path: publishing-engine/cwpc/Plugins/

IN THIS SECTION

- › WebDirect PDF Generation
- › Generate + Download Pattern
- › Email Trigger with PDF

WebDirect PDF Generation

Trigger: Browser button click | Deliver: Container download in browser

- User clicks button in browser → script runs on server
- DynaPDF code is identical to PSOS and desktop
- PDF saved to container → user downloads from browser
- No client-side plugin needed (server does all work)
- Script runs in CWPC process (separate from PSOS)
- Ideal for self-service document generation

WebDirect PDF Workflow

- 1 User clicks "Generate PDF" in WebDirect
- 2 Script triggers (runs server-side in CWPC)
- 3 DynaPDF creates/merges PDF on server
- 4 PDF saved to container field
- 5 Layout refreshes → container shows PDF
- 6 User clicks container to download

WebDirect: Generate + Download Pattern

Script runs in CWPC context — same DynaPDF API as Database Server

```
# — WebDirect PDF Generation Script ———
# Triggered by button click in WebDirect

# Initialize (if not already)
If [ MBS("DynaPDF.IsInitialized") <> 1 ]
  Set Variable [ $r ; MBS("DynaPDF.Initialize";
    "dynapdf.dylib"; $license) ]
End If

Set Variable [ $pdf ; MBS("DynaPDF.New") ]

# Import existing template PDF
Set Variable [ $r ; MBS("DynaPDF.OpenPDFFromContainer";
  $pdf; Settings::LetterheadTemplate_c) ]
Set Variable [ $r ; MBS("DynaPDF.ImportPDFFile";
  $pdf; 1) ]
Set Variable [ $r ; MBS("DynaPDF.CloseImportFile";
  $pdf) ]

# Edit imported page — add dynamic content
Set Variable [ $r ; MBS("DynaPDF.EditPage";
  $pdf; 1) ]
Set Variable [ $r ; MBS("DynaPDF.SetFont";
  $pdf; "Helvetica"; ""; 12; 1) ]
Set Variable [ $r ; MBS("DynaPDF.WriteText";
  $pdf; 72; 600; "To: " & CUST::Name) ]
Set Variable [ $r ; MBS("DynaPDF.EndPage"; $pdf) ]
```

```
# Save to container field
Set Field [ DOC::Quote_PDF_c ;
  MBS("DynaPDF.Save"; $pdf;
    "Quote_" & DOC::DocumentNumber
    & ".pdf") ]
MBS("DynaPDF.Release"; $pdf)
Commit Records []

# Refresh for WebDirect user
Refresh Window [ Flush cached
  join results ]

# User can now click container
# field to download the PDF
```

- Container field = download link
- WebDirect streams PDF to browser
- User gets native browser PDF viewer
- No filesystem access from WebDirect

WebDirect: Email Trigger with PDF

User triggers email from WebDirect → server sends with PDF attachment

- User clicks "Email Invoice" button
- Script generates PDF via DynaPDF
- SendMail + CURL sends email server-side
- No client email app needed (Outlook, etc.)
- SMTP configured in script, not on client
- Identical to Database Server email pattern
- Same SendMail pattern from Section 4

WebDirect-Specific Considerations

- ⚠ No Export Field Contents to desktop
- ⚠ Container field is the ONLY download path
- ⚠ Script triggers are synchronous — user waits
- ⚠ Large PDFs: use PSOS to avoid timeout
- ⚠ No progress indicators during generation
- ⚠ File system limited to server Documents
- ⚠ Users share the same CWPC plugin instance
- ⚠ Memory management critical for multi-user

SECTION 6

Data API / WIP

API-triggered PDF generation, integration workflows, webhook handlers

Plugin path: publishing-engine/wip/Plugins/

IN THIS SECTION

- › Conceptual Overview
- › Invoice Example: Complete Round Trip
- › Quick Reference
- › Authentication & Sessions
- › Running Scripts via API
- › Container Retrieval & Errors
- › OData Integration & Security
- › Webhook Handlers
- › Integration Patterns

Data API: A Different Way to Think About PDFs

The DynaPDF code is identical — what changes is who triggers it and how the PDF gets delivered

DATA API / WIP

29 / 54

What You Already Know

In FileMaker Pro or PSOS, creating a PDF is straightforward:

```
User clicks button → Script runs → DynaPDF creates PDF  
→ Saved to container field → User sees it on layout
```

A FileMaker user triggers it and sees the result. Simple.

What's Different with Data API

There is no FileMaker user. An external system does everything:

```
REST API call → FM script runs in WIP → DynaPDF  
creates PDF  
→ Saved to container → API call downloads the PDF
```

No one is logged into FileMaker. The API triggers and delivers.

Your FileMaker Solution Becomes a Document API

Your DynaPDF code does not change. The exact same script that generates a PDF when a user clicks a button can generate that same PDF when an outside system makes a REST API call. The script doesn't know or care who called it — it receives a parameter, builds the PDF, and saves it.

Only the plumbing changes: how the script gets invoked (REST endpoint vs. button click) and how the PDF gets delivered (HTTP download vs. layout refresh).

This pattern works identically whether you're building for logistics, healthcare, finance, legal, manufacturing, or any other vertical.

Any outside system can request documents without a FileMaker license. What that looks like across industries:

- Logistics — generate 200 bills of lading overnight, zero clicks
- Healthcare — patient portal downloads lab reports on demand
- Finance — deal closes, compliance packet auto-generated and filed
- Manufacturing — inspection triggers certificate PDF to client
- Legal — court filing system pulls contracts via REST

One engine, any document, every industry.

Example: Invoice PDF via Data API

Three API calls: authenticate, trigger the script, download the PDF

The External System (3 API Calls)

```
# — CALL 1: Authenticate —————
curl -X POST \
  https://server/fmi/data/v2/databases/\
    MyDatabase/sessions \
  -H "Content-Type: application/json" \
  -H "Authorization: Basic $CREDS"
# Returns: { "response": { "token": "abc..." } }

# — CALL 2: Trigger PDF Generation —————
# script.param must be URL-encoded JSON
curl -X GET \
  https://server/fmi/data/v2/databases/\
    MyDatabase/layouts/Invoice_Detail/\
    script/Generate_Invoice_PDF\
    ?script.param=%7B%22id%22%3A%2250123%22%7D \
  -H "Authorization: Bearer $TOKEN"
# Returns: { "scriptResult": "OK",
#   "scriptError": "0" }

# — CALL 3: Download the PDF —————
curl -X GET "$CONTAINER_URL" \
  -H "Authorization: Bearer $TOKEN" \
  -o Invoice_50123.pdf
```

The FileMaker Script (You Know This)

```
# "Generate_Invoice_PDF" script
# Same script whether called via PSOS,
# WebDirect, or Data API.

Set Variable [ $id ; JSONGetElement (
  Get (ScriptParameter); "id" ) ]
Enter Find Mode [ ]
Set Field [ INV::InvoiceNumber ; $id ]
Perform Find [ ]

Set Variable [ $pdf ; MBS("DynaPDF.New") ]
MBS("DynaPDF.AppendPage"; $pdf)
# ... text, tables, images, merge ...

Set Field [ DOC::Contract_PDF_c ;
  MBS("DynaPDF.Save"; $pdf; "Invoice.pdf") ]
MBS("DynaPDF.Release"; $pdf)
Exit Script [ Result: "OK" ]
```

The script receives a parameter, builds a PDF, and saves it. It has no idea whether a user clicked a button or an outside system made a REST call.

Data API Quick Reference

Endpoints, authentication, and session management at a glance

Property	Value
Base URL	/fmi/data/v2/databases/{db}/...
Auth	Bearer token (15-min idle expiry)
Run script	GET .../layouts/{l}/script/{s}
Script param	?script.param={JSON}
Container GET	URL returned in fieldData response
Extended priv	fmrest (must be enabled)
Max sessions	Configurable in Admin Console
Usage limits	None on FM Server (Cloud has annual cap)

- Data API v2 (current) or use vLatest
- Plugin runs in WIP process (separate from PSOS)
- Same DynaPDF API as all other contexts
- HTTPS required for all API calls
- Script errors return HTTP 200 (check scriptError)
- Container URLs are session-scoped (use same token)
- JSON parameters: always URL-encode in GET requests
- Claris ID auth: 1-hour token (vs 15-min for Basic)

Authentication & Session Management

Token-based auth with 15-minute idle expiry

```
# — LOGIN: Get Bearer Token —————
# v2 or vLatest both work
curl -X POST \
  https://server/fmi/data/v2/\
    databases/MyDatabase/sessions \
  -H "Content-Type: application/json" \
  -H "Authorization: Basic YWRtaW46cGFzcw==" \
  -d "{}" # empty body required

# Response:
# { "response": {
#   "token": "c4d2e429122e9cde..." },
#   "messages": [
#     { "code": "0", "message": "OK" }] }
```

```
# — LOGOUT: Release Session —————
curl -X DELETE \
  https://server/fmi/data/v2/\
    databases/MyDatabase/sessions/$TOKEN \
  -H "Content-Type: application/json"

# Always logout to free session slots!
```

Authentication Methods

Method	Header	Used With
FileMaker acct	Authorization: Basic {b64}	FM Server
Claris ID	Authorization: FMID {token}	FM Cloud
OAuth / IdP	X-FM-Data-OAuth-*	FM Server

Session Rules

- Token expires after 15 min idle
- Each API call resets the timer
- Claris ID tokens: 1-hour validity
- Validate: GET /fmi/data/v2/validateSession
- Returns isSessionInUse status
- Error 956 = max sessions exceeded
- Always logout when done

Running Scripts via Data API

Standalone execution or attached to CRUD operations

Standalone Script Execution

```
# Run script directly (GET)
curl -X GET \
  https://server/fmi/data/v2/databases/\
    MyDatabase/layouts/Invoice_List/\
    script/Generate_Invoice_PDF\
    ?script.param=%7B%22id%22%3A%2212345%22%7D \
  -H "Authorization: Bearer $TOKEN"

# Response includes scriptResult + scriptError:
# { "response": { "scriptError": "0",
#   "scriptResult": "OK" } }
```

Execution Order (Attached Scripts)

```
# Server runs in this order:
1. Navigate to layout
2. script.prerequisite
3. API action (CRUD/Find)
4. script.presort → sorting
5. script (main) → return result
```

Attached Scripts (on CRUD)

```
# Scripts attached to a Find request
curl -X POST \
  https://server/fmi/data/v2/databases/\
    MyDatabase/layouts/Invoice/_find \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "query": [{"Status": "Approved"}],
    "script": "Generate_Batch_PDFs",
    "script.param": "all_found",
    "script.prerequisite": "Init_DynaPDF"
  }'
```

- Param is single text (use JSON)
- prerequisite → CRUD → presort → script
- HTTP 200 even on script error!
- Always check scriptError in response

Container Retrieval & Error Handling

How external systems download PDFs and handle failures

Retrieving PDF from Container

```
# Step 1: Get record (container returns URL)
curl -X GET \
  https://server/fmi/data/v2/databases/\
    MyDatabase/layouts/Invoice/records/42 \
  -H "Authorization: Bearer $TOKEN"

# Response fieldData includes:
# "PDF_c": "https://server/Streaming_SSL/\
#   MyDatabase/58F59F18526716ED...?RCType=
#   EmbeddedRCFileProcessor"

# Step 2: Download the binary PDF
# Use same session token as the API call
curl -X GET \
  "$CONTAINER_URL" \
  -H "Authorization: Bearer $TOKEN" \
  -o invoice_12345.pdf

# Returns raw PDF binary
# Content-Type: application/pdf
```

API Error Response Format

```
// Every response has a messages array
// Note: these are FM error codes, not HTTP
{ "response": {},
  "messages": [{
    "code": "401",
    "message": "No records match"
  ] }
```

Key Error Codes

Code	Meaning
0	Success
212	Invalid credentials
401	No records match
952	Invalid/expired token
956	Max sessions exceeded

OData Integration & Security

OData 4.01 protocol + security requirements for API access

DATA API / WIP

35 / 54

OData API (Separate from Data API)

```
# OData base URL (different path!)
https://server/fmi/odata/v4/MyDatabase/

# Run script via OData
POST /fmi/odata/v4/MyDatabase/
  Script.Generate_Invoice_PDF
Body: { "scriptParameterValue": "12345" }
```

- Full CRUD + script execution
- Requires fmodata extended privilege
- Webhooks fire on record/schema changes
- Scripts run server-side (like PSOS)
- Script names: no special chars, no leading digits
- Separate from Data API (fmrest)

Security Requirements

Requirement	Detail
Transport	HTTPS mandatory (TLS 1.2+)
SSL cert	Custom CA cert for production
Auth type	Session token (no static API keys)
Extended priv	fmrest (Data API), fmodata (OData)
Container URLs	Require Bearer token to download
Encoding	UTF-8 for all request bodies

URL case Database names are case-sensitive

- No rate limiting (server resources only)
- Max URI length: 2,000 chars
- Concurrent PDF ops share session pool
- Error 956 when pool exhausted

Webhook Handlers

MBS WebHook component — HTTP listener triggers FM scripts

DATA API / WIP

36 / 54

```
# — Setup Webhook Listener —————
# (In a startup/scheduled script)
# Step 1: Create webhook object
Set Variable [ $hook ;
  MBS("WebHook.Create") ]
# Step 2: Assign script to trigger
Set Variable [ $r ;
  MBS("WebHook.SetScript"; $hook;
    Get(FileName);
    "Handle_PDF_Request") ]
# Step 3: Start listening on port
Set Variable [ $r ;
  MBS("WebHook.Listen"; $hook;
    8080) ]
# External system calls:
# POST http://fm-server:8080/
```

```
# — Handle_PDF_Request Script —————
# ScriptParameter = HTTP request body

Set Variable [ $params ;
  Get(ScriptParameter) ]
Set Variable [ $action ;
  JSONGetElement($params; "action") ]
Set Variable [ $recordId ;
  JSONGetElement($params; "recordId") ]

If [ $action = "generate_pdf" ]
  # DynaPDF operations
  Set Variable [ $pdf ; MBS("DynaPDF.New") ]
  # ... generate/merge PDF ...
  Set Field [ DOC::PDF_c ;
    MBS("DynaPDF.Save"; $pdf;
      "document.pdf") ]
  MBS("DynaPDF.Release"; $pdf)
End If
```

- WebHook.Create → WebHook.SetScript → WebHook.Listen (3-step setup)
- Incoming requests trigger the specified FileMaker script with the request body as parameter
- Works across verticals: e-commerce receipts, patient intake forms, shipping documents, inspection certs
- Combine with SendMail to auto-email PDFs when webhook fires

Integration Patterns & Automation

Combining Data API, Webhooks, and Scheduled Scripts

DATA API / WIP

37 / 54

On-Demand via API

External system calls Data API
→ Script generates PDF
→ Returns via container field

- ▶ Patient portal downloads lab results
- ▶ E-commerce pulls packing slips
- ▶ Legal system requests contracts

Event-Driven via Webhook

External event fires webhook
→ Script generates + emails PDF
→ Logs to audit table

- ▶ Deal closes → compliance packet
- ▶ Inspection passes → certificate
- ▶ Shipment dispatched → bill of lading

Scheduled Batch

Server schedule triggers script
→ Finds pending records
→ Batch-generates PDFs

- ▶ Nightly: 500 invoices, zero clicks
- ▶ Weekly: compliance reports filed
- ▶ Monthly: client statements mailed

SECTION 7

Best Practices & Troubleshooting

Error handling, memory management, performance, and common pitfalls

IN THIS SECTION

- › Error Handling Pattern
- › Memory Management on Server
- › Troubleshooting Guide
- › Performance Optimization

Error Handling Pattern

Every MBS call returns "OK" or an error string starting with "[MBS]"

```
# — RECOMMENDED ERROR HANDLING SKELETON —————

# Step 1: Check initialization
If [ MBS("DynaPDF.IsInitialized") <> 1 ]
  Set Variable [ $r ; MBS("DynaPDF.Initialize"; "dynapdf.dylib"; $license) ]
  If [ MBS("IsError") ]
    Exit Script [ Text Result: JSONSetElement( "{}" ; ["error"; "DynaPDF init failed: " & $r; JSONString] ) ]
  End If
End If

# Step 2: Create context with error check
Set Variable [ $pdf ; MBS("DynaPDF.New") ]
If [ MBS("IsError") ]
  Exit Script [ Text Result: JSONSetElement( "{}" ; ["error"; "Cannot create PDF: " & $pdf; JSONString] ) ]
End If

# Step 3: PDF operations (each with error check)
Set Variable [ $r ; MBS("DynaPDF.AppendPage"; $pdf) ]
If [ MBS("IsError") ]
  Set Variable [ $err ; MBS("DynaPDF.GetLastErrorMessage"; $pdf) ]
  Set Variable [ $r ; MBS("DynaPDF.Release"; $pdf) ] // ALWAYS release even on error!
  Exit Script [ Text Result: JSONSetElement( "{}" ; ["error"; $err; JSONString] ) ]
End If

# Step 4: Save and cleanup
Set Field [ Table::PDF_c ; MBS("DynaPDF.Save"; $pdf; "output.pdf") ]
Set Variable [ $r ; MBS("DynaPDF.Release"; $pdf) ] // ALWAYS release!
```

Memory Management on Server

Critical for multi-user, long-running server processes

- ALWAYS call DynaPDF.Release after use
- ALWAYS call DynaPDF.Table.Release too
- ALWAYS call CURL.Release + SendMail.Release
- Release in EVERY path (success AND error)
- NEVER use DynaPDF.ReleaseAll on server!
 - Affects every PSOS, scheduled, WebDirect, and API session
- ReleaseAll frees ALL objects, ALL sessions
- This can corrupt parallel PDF operations
- Monitor: MBS("Plugin.ObjectCounts")
- Get(TemporaryPath) auto-cleans on exit
- Never hold PDF objects across pauses

Server File Paths

Function	Path Type	Lifetime
Get(TemporaryPath)	Session-specific temp folder	Deleted on session end
Get(DocumentsPath)	Server Documents folder	Persistent

OS	Documents Path
macOS	/Library/FileMaker Server/Data/Documents/
Windows	C:\Program Files\FileMaker\...\Data\Documents\
Linux	/opt/FileMaker/FileMaker Server/Data/Documents/

- ⚠ Server: ONLY Documents + Temp folders
- ⚠ No Desktop, Home, or user folders
- ⚠ Paths with ".." are invalid

Troubleshooting Guide

Common errors and their solutions

Symptom	Cause	Solution
"[MBS] DynaPDF not initialized"	Library not loaded in this context	Call DynaPDF.Initialize with correct path
"[MBS] DynaPDF function not available"	DynaPDF Starter used for merge ops	Upgrade to DynaPDF Lite or higher
Error 193 (Windows)	32/64-bit mismatch	Use 64-bit dynapdf.dll for 64-bit FM Server
PDF renders blank	No EndPage call before Save	Always call DynaPDF.EndPage after page content
Plugin works client, fails server	Plugin not in correct server folder	Install in all 3 server plugin locations
PSOS returns "?" result	Script error not captured	Wrap in error handling, check MBS("IsError")
WebDirect can't download PDF	Container field not on layout	Place container on WebDirect-accessible layout
Memory grows over time	DynaPDF objects not released	Audit all Release calls, check error paths
Parallel PDFs corrupted	Using ReleaseAll on server	NEVER use ReleaseAll — use Release per object

Performance Optimization

Keeping PDF generation fast and efficient on server

42 / 54

- Initialize DynaPDF ONCE per session
- OpenPDFFromFile for large PDFs (faster)
- Compress images before inserting
- DynaPDF.Optimize to reduce file size
- Break batches into chunks (avoid holds)
- Release objects immediately after Save
- Use PSOS for WebDirect heavy ops
- Monitor: MBS("Plugin.ObjectCounts")
- External containers keep DB lean

PDF Size Drivers

Component	Impact	Mitigation
Embedded images	HIGH — biggest driver	Compress before inserting
Font embedding	MODERATE	Embed only used fonts
Page count	LOW	Minimal impact
Encryption	NEGLIGIBLE	No significant overhead

Plugin Memory Footprint

Component	Memory
MBS Plugin binary	120–160 MB
DynaPDF per document	2–50 MB
CURL per connection	50–200 KB
SendMail per email	10 KB + attachments

SECTION 8

Looking Ahead

PDFs as immutable data artifacts, AI-ready document repositories,
and the Model Context Protocol (MCP)

IN THIS SECTION

- › PDFs Are Data
- › Claris MCP + FileMaker 2025
- › The Business Case
- › The Virtuous Cycle

PDFs Are Not Just Output — They Are Data

Every PDF you generate is an immutable, timestamped business artifact

The Traditional View

PDFs are output. You generate them, email them, file them. They sit in container fields or folders. They are the end of a workflow.

The Emerging View

PDFs are frozen snapshots of business state. Unlike live database records that change with every edit, a PDF captures exactly what was true at a specific moment: what the invoice said, what the inspection found, what the contract contained. That immutability is not a limitation — it is the feature that makes PDFs uniquely valuable as data.

What You Are Actually Building

- Every MBS-generated PDF is a permanent record
- Container fields become a document repository
- Scheduled scripts build that repository automatically
- Thousands of documents accumulate over months and years
- Each one is searchable, parseable, and time-stamped
- Together they form a complete history of your business

The Compounding Value of Document Generation — Every document immutable. Every document timestamped. Every document AI-ready.

Year 1

7,200 documents

5,000 invoices
200 compliance packets
1,200 inspection certs
800 contracts

Year 3

21,600 documents

15,000 invoices
600 compliance packets
3,600 inspection certs
2,400 contracts

Year 5+

50,000+ documents

35,000+ invoices
1,500+ compliance packets
8,000+ inspection certs
5,500+ contracts

Claris MCP + FileMaker 2025: What Exists Today

Launched December 2025 — AI agents already have a path into your FileMaker data

45 / 54

Model Context Protocol (MCP)

An open standard created by Anthropic, donated to the Agentic AI Foundation (AAIF) under the Linux Foundation. Co-founded by Anthropic, OpenAI, and Block. Over 10,000 open-source MCP servers already exist.

Claris MCP for FileMaker (December 2025)

Claris shipped an official MCP server that bridges FileMaker to AI assistants like Claude. It auto-generates tools from your database structure — data tools for each table (search, list, create, update, delete) and script tools that expose your existing FileMaker scripts as AI-accessible functions. Security respects your FileMaker account privileges. All actions are logged.

FileMaker 2025: GetTextFromPDF()

A native function that extracts text from PDFs in container fields. Point it at a container → get plain text back. That text can be indexed, searched, fed into RAG pipelines, or used for AI-powered summaries. This is the bridge between MBS-generated PDFs and AI.

Container Fields Meet AI

```
AI Agent (Claude, ChatGPT, etc.)
├── Claris MCP Server
│   ├── Auto-generated data tools
│   ├── Container field tools (upload/download)
│   └── Script tools (trigger FM scripts)
├── GetTextFromPDF() extracts content
└── AI reasons over the documents:
    ├── "Summarize this quarter's inspection results"
    ├── "Which contracts expire this month?"
    └── "Compare revenue trends Q1 vs Q2"
```

Why PDFs Are Ideal for AI Context

- Immutable — data never changes after generation
- Timestamped — AI knows when each snapshot was taken
- Self-contained — all context in one file, no joins needed
- Already generated — no ETL or export required
- Text-extractable — GetTextFromPDF() makes them searchable

The Business Case: MBS as Strategic Infrastructure

Every PDF you generate today becomes AI-accessible context tomorrow

46 / 54

The Licensing Question, Reframed

The traditional case for MBS Plugin + DynaPDF is operational: generate invoices, merge documents, automate reporting. That alone justifies the investment.

But there is a second, compounding return. Every PDF your MBS scripts generate is stored in a container field. Over months and years, thousands of immutable, timestamped business documents accumulate. That growing archive is not dead storage — it is a structured knowledge base that AI agents can now access through Claris MCP and GetTextFromPDF().

What This Means for Your Clients

- MBS investment pays twice: operational value now, AI value later
- No new infrastructure needed — FileMaker IS the repository
- Organizations generating documents now build richer AI context than those starting later
- Competitive advantage compounds over time, not overnight
- The \$1,247 Year 1 investment seeds a multi-year AI asset

Two Returns on One Investment

RETURN 1: OPERATIONAL

**Automate PDF
generation today**

Invoices, contracts, reports, compliance docs — generated, merged, and emailed without manual effort.

RETURN 2: STRATEGIC

**Build an AI-ready
document archive**

Every PDF becomes a searchable, timestamped artifact. Claris MCP + GetTextFromPDF() turn your archive into AI context.

\$1,247

Year 1 total cost

\$454/yr renewal after Year 1
MBS Server + Developer + DynaPDF Lite

The Virtuous Cycle

MBS Plugin is not just a cost — it is an investment that compounds

47 / 54

TODAY

Generate

MBS Plugin + DynaPDF create PDFs across all three server contexts.

Invoices, reports, contracts, certificates flow automatically into container fields.

Your FileMaker solution is the single source of truth.

TOMORROW

Accumulate

Every generated PDF is an immutable business artifact.

Over months and years, your repository grows into a comprehensive archive.

GetTextFromPDF() makes each one searchable and AI-accessible.

THE FUTURE

Intelligence

AI agents connect via Claris MCP and reason over your documents.

Trend analysis, compliance auditing, contract review, anomaly detection —
powered by documents you already have.

Today's documents become tomorrow's intelligence. Every PDF deepens the archive AI will reason over.

Quick Reference: Three Contexts

Database Server

`Extensions/`

- PSOS execution
- Scheduled scripts
- Batch PDF generation
- Server-side email
- Automated workflows

Web Publishing

`cwpc/Plugins/`

- WebDirect PDF downloads
- User-triggered generation
- Email from web interface
- Container-based delivery
- Server-side processing

Data API

`wip/Plugins/`

- API-triggered scripts
- Webhook handlers
- Container retrieval
- Integration pipelines
- Automated responses

Resources & Documentation

Resource	URL / Location
MBS Plugin Homepage	monkeybreadsoftware.com/filemaker/
DynaPDF Function Reference	mbsplugins.eu/component_DynaPDF.shtml
DynaPDF Editions Comparison	monkeybreadsoftware.com/filemaker/dynapdf-editions.shtml
MBS Plugin Pricing	monkeybreadsoftware.com/filemaker/pricing.shtml
Server Installation Guide	monkeybreadsoftware.com/filemaker/files/Installation.pdf
SendMail / Email Guide	monkeybreadsoftware.com/filemaker/files/Guides/Send Email.pdf
MBS Example Databases (600+)	Included in MBS Plugin download under Examples/
DynaPDF Manual (PDF)	Included in download as dynapdf_help.pdf

Key Example Databases:

- Merge PDFs.fmp12 — Merge multiple PDF files
- Tables.fmp12 — PDF table creation
- WriteFText.fmp12 — Formatted text output
- Add Page Numbers.fmp12 — Post-processing
- Watermark pages.fmp12 — Watermark application

SECTION 9

Q & A

Glossary, frequently asked questions, and common pitfalls

IN THIS SECTION

- › Glossary
- › FAQ (1 of 2)
- › FAQ (2 of 2)

Glossary: Key Terms Used in This Guide

Term	What It Means
PSOS	Perform Script on Server — runs a FM script on the server from a client call
CWPC	Custom Web Publishing Core — the server process that runs WebDirect
WIP	Web Integration Process — the server process that handles Data API calls
DynaPDF	A PDF engine (648 functions) embedded inside the MBS Plugin. Separate license.
MBS Plugin	MonkeyBread Software Plugin — adds 7,700+ functions to FileMaker
Container field	A FileMaker field type that stores files (PDFs, images, etc.)
Bearer token	A temporary credential returned by the Data API after login. Expires after 15 min idle.
\$pdf context	A DynaPDF object in memory. Created by DynaPDF.New, freed by DynaPDF.Release.
Extensions/	The folder where server-side plugins live for PSOS and scheduled scripts

Frequently Asked Questions (1 of 2)

The questions developers always ask first

52 / 54

Q: Why can't I just use Save Records as PDF on the server?

Save Records as PDF renders a layout visually. On the server there is no screen, so it fails or produces blank pages. DynaPDF builds the PDF programmatically — no layout rendering required.

Q: Do I need BOTH an MBS Plugin license AND a DynaPDF license?

Yes. The MBS Plugin is the host (\$599/server + \$149/dev). DynaPDF is an add-on engine with its own license (\$499 minimum for Lite, which is the minimum for merging existing PDFs). Starter (\$249) can only create from scratch.

Q: Do I really have to install the plugin in three separate folders?

Only the folders you need. Extensions/ is for PSOS and schedules. cwpc/ is for WebDirect. wip/ is for Data API. Most solutions start with Extensions/ only and add others as needed. Same binary, just copied to each folder.

Q: Can I use the same script for PSOS and Data API?

Absolutely. The script receives a parameter via Get(ScriptParameter), builds the PDF, and saves to a container. It has no idea whether it was called by a button click, PSOS, a schedule, or a REST API call.

Q: What if two users generate PDFs at the same time?

Each server session gets its own DynaPDF context (\$pdf). They run in parallel without interfering. Just remember to always Release your context when done — otherwise you leak memory across sessions.

Q: What's the difference between Starter and Lite?

Starter (\$249) creates PDFs from scratch only — text, tables, images on blank pages. Lite (\$499) adds the ability to open, import, and merge existing PDFs. If you need to combine a contract with an appendix, you need Lite minimum.

Q: Do I need DynaPDF Pro?

Only if you need DynaPDF.Optimize (compress/shrink file size) or the rendering engine. Most production use cases — creating invoices, contracts, reports, and merging document packages — are fully covered by Lite.

Q: Does this work on FileMaker Cloud?

No. FileMaker Cloud (Claris-hosted) does not support server-side plugins. You need FileMaker Server (self-hosted or customer-hosted) for MBS Plugin and DynaPDF. This is a hard requirement — there is no workaround.

Frequently Asked Questions (2 of 2)

Common pitfalls and practical considerations

53 / 54

Q: My PDF comes back blank / has errors. What's wrong?

Most common causes: (1) `DynaPDF.Initialize` was never called — check `MBS("DynaPDF.IsInitialized")` first. (2) The `dynapdf.dylib/.dll` file is missing from the server folder. (3) The DynaPDF license key is expired or wrong edition. (4) You forgot `DynaPDF.AppendPage` before writing content. Always check `MBS("IsError")` after every call.

Q: What about `DynaPDF.ReleaseAll` — is that a good safety net?

Dangerous on server! `ReleaseAll` frees ALL DynaPDF objects across ALL sessions. If another PSOS or API script is building a PDF at the same time, you just destroyed their work. Only use it in a startup script that runs when no other scripts are active, or as a last-resort emergency cleanup.

Q: How big can the PDFs get?

DynaPDF can handle very large documents (hundreds of pages). The practical limit is server RAM. Each PDF context is in memory until saved. For large batches, use `DynaPDF.OpenOutputFile` to stream directly to disk instead of holding the entire PDF in memory via `DynaPDF.Save`.

Q: How does the external system get the PDF back via Data API?

Two-step process: (1) After the script saves the PDF to a container field, do a GET on the record. The container field returns a URL (not the file itself). (2) Make a second GET request to that URL with the same Bearer token to download the actual binary PDF. The URL is session-scoped — it only works with the token that requested it.

Q: What if my token expires mid-generation?

The 15-minute idle timer resets with each API call. Since the script trigger IS an API call, the timer resets when you invoke it. You only risk timeout if the script itself takes more than 15 minutes to run (very unusual) or if you wait too long between triggering and downloading. Build your integration to download promptly after generation.

Q: Where should I start if I'm brand new to this?

Start with the MBS example databases (600+ included in the download). Open Merge PDFs.fmp12 and Tables.fmp12 — they have working scripts you can run immediately. Get comfortable with the 5-step lifecycle (Initialize → New → Build → Save → Release) on your desktop first, then deploy to the server.

Remember: if you can write a FileMaker script, you can generate PDFs on the server. The DynaPDF API is well-documented (648 functions, all searchable at mbsplugins.eu), the example databases give you working code to start from, and Christian Schmitz at MonkeyBread Software provides direct support for licensed users. The learning curve is real but manageable — most developers are productive within a day.

MBS Plugin Server Edition

PDF Creation & Merging Guide

One engine. Any document. Every industry.

© 2026 Database Success | Matthew Hardy | March 2026

Many thanks to Christian Schmitz of [MonkeyBread Software](#) for his great software and invaluable input to this guide.