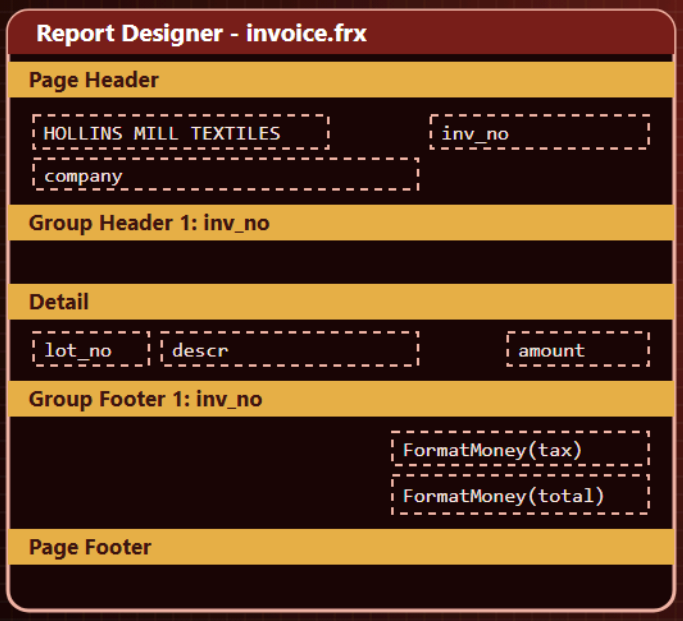
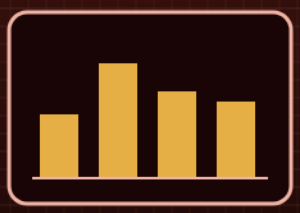





```
loListener = CREATEOBJECT("ReportListener")
loListener.ListenerType = 3
REPORT FORM invoice OBJECT loListener
loListener.OutputPage(1, "inv.png", 104)
```



A PRACTICAL DEEP DIVE · VFP 9

Visual FoxPro Reports and Output

Report Designer, ReportListeners, Labels, Charts, and Office Export in VFP 9

Report data · Bands and groups · Invoices · Statements · Labels
Listeners · PNG · HTML · XML · Printing · PDF · Excel · CSV

DESIGN · DOCUMENTS · LISTENERS · PRINT · EXPORT

THENEXBOOKS

Visual FoxPro Reports and Output

Report Designer, ReportListeners, Labels, Charts, and Office Export in VFP 9

From a prepared cursor to invoices, statements, images, HTML, Excel, and a month-end report pack

First Edition

THENEXBOOKS

Visual FoxPro Reports and Output
Report Designer, ReportListeners, Labels, Charts, and Office Export in VFP 9

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. Code examples may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Visual FoxPro 9.0 is no longer sold or supported by Microsoft; behavior can differ between service packs, printer drivers, Windows versions, and Office versions. Test every report and export with your own data, printers, and software before relying on it, and back up your files before running examples that change them.

Microsoft, Visual FoxPro, Windows, Excel, Word, Outlook, and Office are trademarks of the Microsoft group of companies. Other product names are trademarks of their respective owners. Their use in this book is for identification only and does not imply endorsement. This book is independent and is not affiliated with, sponsored by, or endorsed by Microsoft Corporation.

Hollins Mill Textiles, the textile mill used as the running example in this book, is fictional, as are its people, customers, fabrics, orders, invoices, addresses, and all sample data. Phone numbers use the 555 range and email addresses use reserved example domains. The report pages shown as figures were rendered by Visual FoxPro 9 from this fictional sample data.

Preface

Every business application is judged by what comes out of it. Users rarely see your tables, your indexes, or your carefully normalized design; they see the invoice that goes to a customer, the packing slip that travels with a pallet, the statement that tells an account how much it owes, and the spreadsheet a manager opens on Monday morning. In a Visual FoxPro application, that output is often the part that has aged the most: reports designed years ago for a printer that no longer exists, totals that only one person trusts, and requests such as "can we get this as a PDF?" or "can it go straight into Excel?" that nobody quite knows how to answer.

Visual FoxPro 9 is far better at output than its reputation suggests. Its Report Designer handles grouping, running totals, conditional printing, stretching text, labels, and multi-column layouts. Its ReportListener architecture, introduced in VFP 9, turns every report run into an object you can inspect and extend: you can render pages to images, produce HTML and XML, change formatting record by record, and draw watermarks without touching the layout. Around the report engine sit the tools every developer needs for the rest of the job: text merge for CSV and fixed-width files, COM automation for Excel and Word, and the simple fact that an FRX report is just a table you can read and generate with code.

This book is a deep dive into all of it. We take one fictional company and build its complete output layer, from the first prepared cursor to a month-end report pack, explaining not only which options to set but why the report engine behaves the way it does. Every canonical report in this book was built and run with Visual FoxPro 9, and the report pages you see as figures are real output rendered by the report engine itself.

Who This Book Is For

- **Developers who maintain VFP applications** and are responsible for dozens of existing reports, labels, and exports that must keep working.
- **Developers extending a VFP system** who need new documents, statements, dashboards, or file exports for customers, accountants, and partners.
- **Teams moving output off paper** who need images, HTML, Excel files, CSV, or PDF instead of a printer in the corner.
- **Developers from other platforms** who have inherited a VFP application and want to understand its reports quickly. You should already know basic VFP commands, tables, and SQL SELECT; we refresh only what reporting needs.

What You Will Learn

- How to prepare report data properly: one cursor per report, built with SQL, ordered for grouping, and checked before it ever reaches a layout.

- How to use the Report Designer in depth: bands, groups, variables, calculated fields, Print When, stretch and float, title and summary bands, multiple detail bands, multi-column reports, and labels.
- How to write report expressions and format pictures that print exactly what you expect.
- How to build real business documents: invoices, packing slips, production lot reports, customer statements, receivables aging, cross-tabs, and reports with logos and charts.
- How SET REPORTBEHAVIOR 90 and the ReportListener architecture work, how to use the FFC listener classes and the ReportBuilder, ReportPreview, and ReportOutput applications, and how to write your own listeners for dynamic formatting and watermarks.
- How to render reports to PNG images, HTML, and XML; how to print reliably and avoid printer environment pitfalls; and what your realistic options are for PDF.
- How to export to Excel and Word through automation, and to CSV, text, and fixed-width files with TEXTMERGE and COPY TO.
- How to read, audit, and generate FRX files with code, distribute reports in batches, keep them fast, troubleshoot them, and assemble a complete month-end report pack.

How This Book Is Organized

Part I — Getting Data Ready for Output (Chapters 1–5) introduces VFP's output options, report data preparation, the Report Designer, report expressions, and running reports from code. **Part II — The Report Designer in Depth** (Chapters 6–10) covers grouping, report variables, adaptive layouts, title, summary, and detail bands, and multi-column reports and labels. **Part III — Business Documents** (Chapters 11–15) builds invoices, packing slips, production reports, statements, aging, cross-tabs, and reports with images and charts. **Part IV — The ReportListener Architecture** (Chapters 16–20) explains object-assisted output, the FFC listener classes and reporting applications, custom listeners, image output, and HTML and XML. **Part V — Printing and Export** (Chapters 21–24) covers printers, PDF, Excel and Word automation, and text and CSV files. **Part VI — Production Reporting** (Chapters 25–28) treats the FRX as a table, distribution, performance and troubleshooting, and ends with a capstone month-end report pack. The appendices provide a cheat sheet, an FRX and listener reference with templates, a production checklist, and a glossary.

Each chapter ends with exercises and a **Key Takeaways** box for quick review.

The Running Example: Hollins Mill Textiles

Throughout the book we build the output layer of **Hollins Mill Textiles**, a fictional mill that weaves cotton, wool, linen, and blended fabrics on three looms and sells them by the yard to upholsterers, quilt shops, costume makers, sailmakers, and interior designers. Margaret Hollins runs the business; Rosa Delgado in accounts receivable needs statements and an aging report; Owen Pike in shipping needs packing slips and labels; Nell Hargrove on the mill floor needs production lot reports; and Theo

Nakamura, the developer, has to make all of it work. The sample database is small enough to check by hand — six customers, eight fabrics, ten production lots, twelve orders, and ten invoices — so every total in this book can be verified, from invoice INV-26103 for $1,363.83 to an aging total of $8,735.94.

Versions and Conventions

Examples target **Visual FoxPro 9.0 SP2**, the final release of the product. Most of the book also applies to VFP 9 without SP2; where a detail differs or depends on a service pack, a Note says so. Report output depends on Windows, fonts, and printer drivers, so your pages may differ slightly from the figures. The examples assume US settings — `SET DATE AMERICAN`, `SET CENTURY ON`, and `$` as the currency symbol — so dates print as `02/19/2026`.

> **TIP**
> A recommended practice or a technique that makes your work easier.

> **NOTE**
> Background information, version differences, or a useful aside.

> **WARNING**
> A common mistake or something that can cause wrong output, lost data, or printing problems.

VFP commands and code appear in `monospace`, with keywords in uppercase; long lines are continued with a semicolon. Menu paths are written in bold, such as **Report > Data Grouping**; menu and dialog labels can differ slightly between service packs. Program files live in `C:\HollinsMill\Prog\`, reports in `C:\HollinsMill\Reports\`, data in `C:\HollinsMill\Data\`, and generated files in `C:\HollinsMill\Output\`.

Contents

Preface

PART I — GETTING DATA READY FOR OUTPUT

1 Output in Visual FoxPro 9: The Reporting Landscape and the Hollins Mill Project

2 Preparing Report Data: Cursors, Queries, and One Row per Printed Line

3 The Report Designer Tour: Bands, Controls, Properties, and Quick Reports

4 Report Expressions, Functions, and Format Pictures

5 Data Environments, Data Sessions, and Running Reports from Code

PART II — THE REPORT DESIGNER IN DEPTH

6 Data Grouping: Group Bands, Group Totals, and New-Page Rules

7 Report Variables and Calculated Fields: Running Totals, Percentages, and Resets

8 Print When, Stretch, Float, and Remove-Line Rules: Layouts That Adapt

9 Title, Summary, and Multiple Detail Bands

10 Multi-Column Reports and Labels: The Label Designer and LBX Files

PART III — BUSINESS DOCUMENTS

11 Invoices: Header Blocks, Line Items, and Totals That Match the Ledger

12 Packing Slips and Production Lot Reports

13 Customer Statements and Receivables Aging

14 Cross-Tab and Pivot Reports: vfpxtab, Pivot Cursors, and Matrix Layouts

15 Images, Logos, and Charts in Reports

PART IV — THE REPORTLISTENER ARCHITECTURE

16 SET REPORTBEHAVIOR 90 and Object-Assisted Output

17 The FFC Listener Classes and the ReportBuilder, ReportPreview, and ReportOutput Apps

18 Writing Custom Listeners: Render, EvaluateContents, and Watermarks

19 Rendering Reports to Images: ListenerType 3 and OutputPage

20 HTML and XML Output with Report Listeners

PART V — PRINTING AND EXPORT

21 Printing: SET PRINTER, APRINTERS, PRTINFO, and Printer Environment Pitfalls

22 PDF Output Without a Native PDF Engine

23 Excel and Word Automation for Report Exports

24 Text, CSV, and Fixed-Width Output: TEXTMERGE, STRTOFILE, and COPY TO

PART VI — PRODUCTION REPORTING

25 The FRX as a Table: Reading, Auditing, and Generating Reports with Code

26 Report Distribution: Batch Runs, File Naming, Archiving, and Email

27 Report Performance and Troubleshooting

28 Capstone: The Hollins Mill Month-End Report Pack

APPENDICES

A Reporting Cheat Sheet

B FRX Reference, Templates, and Listener Skeletons

C Report Production Checklist

D Glossary

PART I

Getting Data Ready for Output

What output means in Visual FoxPro 9, why every good report starts with one prepared cursor, a guided tour of the Report Designer, report expressions and format pictures, and running reports reliably from code.

1 · The Reporting Landscape
2 · Preparing Report Data
3 · The Report Designer Tour
4 · Expressions and Format Pictures
5 · Data Environments and Running Reports

CHAPTER 1 · PART I — GETTING DATA READY FOR OUTPUT

Output in Visual FoxPro 9: The Reporting Landscape and the Hollins Mill Project

Visual FoxPro is very good at getting data in and keeping it safe. This book is about the other half of the job: getting data out, on paper, on screen, and in files that other people and other programs can use. In this chapter we map the territory, meet the mill whose output we will build, create its database, and produce our first report in a few lines of code.

What "Output" Means in a VFP Application

When a business asks for "a report", it rarely means a single thing. The accounts receivable clerk wants a statement she can mail. The shipping desk wants a packing slip that goes in the carton and a label that goes on the outside. The owner wants a one-page summary she can read on a tablet, and the accountant wants "the same numbers in Excel". Each of those is output, and Visual FoxPro 9 gives you a different tool for each one.

It helps to name the kinds of output before we start, because the right tool depends on who reads the result and what they will do with it next. Here is the landscape this book covers.

Kind of output	Main VFP tools	Typical reader	Chapters
Banded reports (FRX)	Report Designer, `REPORT FORM`	People, on paper or screen	3–15
Labels (LBX)	Label Designer, `LABEL FORM`	Carriers, mail rooms	10
Listener output	`ReportListener` classes	Previews, custom rendering	16–18
Images, HTML, XML	`OutputPage()`, ReportOutput.app	Web pages, archives, other programs	19–20
Printed pages	`SET PRINTER`, `APRINTERS()`	Anyone with a printer	21
PDF	PDF printer drivers, Office export	Customers, email	22
Office files	Excel and Word automation	Analysts, managers	23
Text, CSV, fixed width	TEXTMERGE, `STRTOFILE()`, `COPY TO`	Banks, other systems	24

Most of the book is about the first row, because the banded report is still the workhorse of VFP output. But the other rows matter more every year. An invoice that used to go to a dot-matrix printer now often has to become an image, a PDF attachment, or a row in a spreadsheet. A good output layer in a

VFP application is one where the same prepared data can feed any of those destinations without being computed twice.

That last idea runs through the whole book: **prepare the data once, in a cursor, then send it wherever it has to go.** Chapter 2 builds the data preparation layer. Everything after that consumes it.

Two Report Engines in One Product

Visual FoxPro 9 ships with two ways of running the same report file. Which one you get is controlled by `SET REPORTBEHAVIOR`, and understanding the difference early saves a lot of confusion later.

SET REPORTBEHAVIOR 80: the classic engine

With `SET REPORTBEHAVIOR 80`, `REPORT FORM` uses the same engine that earlier versions of Visual FoxPro used. It renders through the older Windows GDI drawing interface, it is fast, and it is what most legacy reports were designed and tested against. It supports printing, previewing, and text output with `TO FILE ... ASCII`. It does not give your code any chance to step in while a page is being drawn. On a fresh VFP 9 installation this is the startup default unless someone has changed it in Tools > Options > Reports or in the configuration file, and we confirmed it on our test machine:

Command Window

```
? SET("REPORTBEHAVIOR")
```

Output

```
        80
```

SET REPORTBEHAVIOR 90: object-assisted output

With `SET REPORTBEHAVIOR 90`, or whenever you add an `OBJECT` clause to `REPORT FORM`, VFP 9 runs the report through a `ReportListener` object. The engine still reads the same FRX file and walks the same bands, but every band and every object passes through the listener, which renders with GDI+. Your own listener classes can watch those events, change how an object looks, draw extra things such as a watermark, or turn the pages into images, HTML, or XML. Part IV of this book is devoted to this architecture.

The two engines usually produce nearly identical pages, but not exactly identical ones. GDI+ measures and spaces text differently from GDI, so a field that fit perfectly under the old engine may wrap or be trimmed with an ellipsis under the new one. That is the most common surprise when an old application is switched to `REPORTBEHAVIOR 90`, and Chapter 4 and Chapter 16 show how to design around it.

> **NOTE**
>
> The system variables `_REPORTOUTPUT`, `_REPORTPREVIEW`, and `_REPORTBUILDER` point to three helper applications in the VFP home folder: ReportOutput.app, ReportPreview.app, and ReportBuilder.app. They provide the default listeners, the preview window, and the VFP 9 report property dialogs. If your application is distributed as an EXE, those helpers have to travel with it or be built into it; Chapter 17 covers the details.

Report Files Are Tables

A report layout lives in two files: an **FRX** file and its **FRT** memo file. Labels use the same format with different extensions, **LBX** and **LBT**. The most useful fact about these files is that they are ordinary Visual FoxPro tables. You can `USE` one, `LIST` it, `BROWSE` it in the development environment, and change it with `REPLACE`.

Each record describes one thing in the layout. Record 1 is the report header, which holds report-wide settings such as the default font, margins, and (if someone saved it) the printer environment. Other records describe bands, labels, fields, lines, boxes, pictures, report variables, and the data environment. The `OBJTYPE` field says which kind of record it is: 5 for a label, 6 for a line, 7 for a box, 8 for a field, 9 for a band, 18 for a report variable, and 25 and 26 for the data environment and its cursors. Positions and sizes are stored in FRUs, units of 1/10,000 inch.

Because FRX files are tables, a lot of the work in this book can be done from code. The layouts for every canonical Hollins Mill report are generated by a program, `buildreports.prg`, using a small library, `frxlib.prg`, that writes FRX records the way the Report Designer would. You can still open every one of those reports with `MODIFY REPORT` and edit it by hand. Chapter 25 explains the whole format; for now it is enough to know that nothing about an FRX is hidden from you.

> **TIP**
>
> Before you open someone else's FRX as a table, copy it (both the FRX and the FRT) to a scratch folder and work on the copy. A report file you damage with a careless `REPLACE` can stop loading in the designer, and the copy costs nothing.

Meet Hollins Mill Textiles

Every chapter in this book works on the same fictional business, so the examples build on each other instead of starting over. Hollins Mill Textiles weaves cotton, wool, linen, and blended fabrics on three looms and sells them by the yard to upholsterers, quilt shops, costume makers, sailmakers, and interior designers. Its order and accounting system is a VFP 9 application that we will simply call the Hollins Mill system. The data entry screens already exist. What the mill needs now is a dependable output layer.

The people who will ask for that output are:

- **Margaret Hollins**, the owner and the third generation of her family to run the mill. She wants a clear picture of sales by fabric category and a month-end pack she can read in one sitting.
- **Rosa Delgado**, accounts receivable. She wants invoices that match the ledger to the cent, customer statements, and an aging report she trusts.
- **Owen Pike**, shipping. He needs packing slips without prices and shipping labels that line up on the label sheets.
- **Nell Hargrove**, the mill production manager. She wants a lot production report grouped by loom, with failed quality checks impossible to miss.
- **Theo Nakamura**, the developer who maintains the Hollins Mill system. Theo is your stand-in: when the book says "Theo opens the Report Designer", picture yourself doing it.

Every printed document carries the mill's letterhead:

```
HOLLINS MILL TEXTILES
1 Millrace Road, Alder Falls, NC
Phone 555-0140   accounts@example.com
```

The customers are six businesses with codes C1001 to C1006, from Bramble & Thread Quilt Co. in Asheville to Redfern Sailmakers in Portland, Maine. Notice the ampersand in the first company name. It is there on purpose. Every time we write HTML, XML, or CSV in later chapters, that one character tests whether our escaping is correct.

The folder layout

The Hollins Mill system keeps its files in four folders, and the book uses the same names throughout:

Folder	Contents
`C:\HollinsMill\Data\`	The database `hollins.dbc` and its tables
`C:\HollinsMill\Prog\`	All program files (`.prg`)
`C:\HollinsMill\Reports\`	Report and label layouts (FRX/FRT, LBX/LBT)
`C:\HollinsMill\Output\`	Generated files, in month folders such as `2026-04\`

Keeping generated output in its own tree is not just tidiness. It means a batch run can be repeated safely, an archive can be backed up separately from the data, and nobody accidentally overwrites a report layout with a generated file of the same name. Chapter 26 turns this folder convention into a function.

The data model in one paragraph

The database has eight tables. `customers` and `fabrics` are the masters. `lots` records each production run: which fabric, which loom, how many yards, how many defects, and whether the lot passed quality control. `orders` and `orditems` hold the orders and their lines, and every order line names the lot it was cut from. When an order ships, a `shipments` record and an `invoices` record are created on the same day. `payments` records the money that has come in. Money fields use the Currency type throughout, so totals are exact to four decimal places and never suffer from floating-point drift.

The sample data covers February to April 2026: twelve orders, ten of them shipped and invoiced, one still open, and one cancelled. Rosa's statements and aging reports are always run "as of" April 30, 2026. Those dates and amounts are fixed, so the numbers you see in this book are the numbers you will get when you run the code.

Building the Database

The sample database is created by a single program, `setup.prg`, which ships with the book's code. It creates `hollins.dbc`, all eight tables with their indexes, and every sample row. It always starts from scratch, so you can run it again whenever an experiment leaves the data in a state you do not like.

Command Window

```
CD C:\HollinsMill\Prog
DO setup WITH "C:\HollinsMill\Data\"
```

The program returns the folder it used. To check the result, open the database and count the rows in each table. This short program does exactly that, using `ADBOBJECTS()` to get the table names from the database container:

C:\HollinsMill\Prog\countrows.prg

```
OPEN DATABASE C:\HollinsMill\Data\hollins SHARED
LOCAL lnN, i, laTables[1]
lnN = ADBOBJECTS(laTables, "TABLE")
ASORT(laTables)
FOR i = 1 TO lnN
   USE (laTables[i]) IN 0 ALIAS curT SHARED AGAIN
   ? PADR(LOWER(laTables[i]), 10), RECCOUNT("curT")
   USE IN curT
ENDFOR
CLOSE DATABASES ALL
```

Output

```
customers           6
fabrics             8
invoices           10
lots               10
orders             12
```

```
orditems          24
payments           6
shipments         10
```

If your counts match, the data is exactly the data used for every example and figure in this book. If they do not, run `setup.prg` again before going further; an output problem is much harder to diagnose when you are not sure about the input.

> **WARNING**
>
> `setup.prg` deletes and recreates the Hollins Mill tables in the folder you pass to it. Point it only at a folder you use for the book's examples, never at a folder that holds real application data.

Your First Report in Three Commands

Before we build anything elaborate, let us see the whole cycle once: choose data, create a layout, and run it. Visual FoxPro can generate a simple column layout for you with a **quick report**. In the designer this is the Report > Quick Report menu command; from code it is `CREATE REPORT ... FROM`, which writes a complete FRX without opening any window.

Owen has asked for a stock list he can glance at before he promises a delivery date. The data is a simple query on `fabrics`:

`C:\HollinsMill\Prog\fabstock.prg`

```
OPEN DATABASE C:\HollinsMill\Data\hollins SHARED
SELECT fabric_id, descr, stock_yd, reorder_yd ;
   FROM fabrics ;
   ORDER BY fabric_id ;
   INTO CURSOR curFabList NOFILTER
CREATE REPORT C:\HollinsMill\Reports\fabstock FROM curFabList
REPORT FORM C:\HollinsMill\Reports\fabstock ;
   TO FILE C:\HollinsMill\Output\fabstock.txt ASCII NOCONSOLE
```

The `SELECT` builds a cursor. `CREATE REPORT ... FROM` looks at the cursor's fields and writes `fabstock.frx` and `fabstock.frt` with one column per field, a heading for each column, the date and page number in the page footer, and a detail band that prints one line per record. `REPORT FORM ... TO FILE ... ASCII` runs the report and writes the result as plain text instead of sending it to a printer. `NOCONSOLE` keeps the output from also being echoed to the main VFP window.

Here is the text file, with the blank lines at the end of the page removed:

`C:\HollinsMill\Output\fabstock.txt`

```
Fabric_id  Descr                              Stock_yd Reorder_yd
BL-4001    Cotton-Linen Blend Sage                 0        150
BL-4002    Poly-Wool Gabardine Black             520        200
CT-1001    Mill Canvas 10 oz Natural            1200        300
CT-1002    Quilting Cotton Indigo Dot            860        400
```

```
CT-1003    Duck Canvas 12 oz Olive                150     150
LN-3001    Heavy Linen Oatmeal                    410     200
WL-2001    Worsted Flannel Charcoal               240     150
WL-2002    Melton Wool Navy                        95     100
10/08/20                                                                1
```

Three commands, one report. The headings are the field names with an initial capital, the rows are in `fabric_id` order, and the page number sits at the right of the footer. Look closely at the last line, though. The date reads `10/08/20`: the quick report sized the `DATE()` field for the layout's font, and the text renderer, which maps the page to a fixed grid of characters, had too few columns for the four-digit year. Text output from a graphical layout is always approximate. It is fine for quick checks and for logs, and Chapter 24 shows better tools when a text file is the real deliverable.

The ASCII text grid is controlled by two system variables, `_ASCIICOLS` and `_ASCIIROWS`. On our test machine they were 80 and 63, which is why the page above is 80 characters wide and why the file contains 63 lines including the blank ones.

The same FRX can also be drawn as a real page. Figure 1.1 is `fabstock.frx` rendered to an image through a `ReportListener`, the technique Chapter 19 explains, with the empty middle of the page cut out. It shows what the quick report really is: field names as bold headings, a fixed-pitch font, numeric columns that sit slightly lower than the text columns beside them, and a date field so narrow that the graphical engine trims it with an ellipsis. That is a perfectly usable working list, and nothing you would send to a customer.

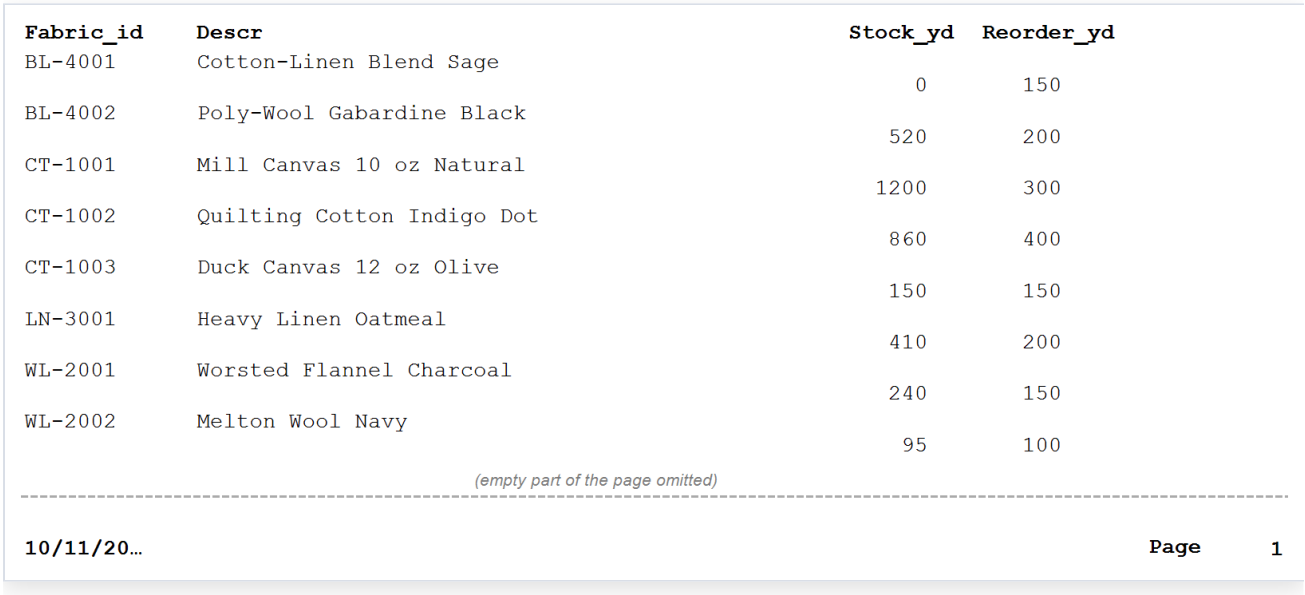


Figure 1.1 — fabstock.frx, the quick report generated by CREATE REPORT ... FROM, rendered with RenderPng() (top of the page and the page footer)

Why the NOFILTER clause is there

The `NOFILTER` clause at the end of the query looks like a detail, but leaving it out produces a report that silently ignores your `ORDER BY`. Here is what we observed when we ran the same steps without it. When a query reads a single table, selects plain fields, and can be satisfied using an existing index,

VFP may not build a new cursor at all. It opens the original table again under the new alias with a filter, which is fast but means that `DBF("curFabList")` returns the path of `fabrics.dbf` itself.

A quick report built from such a cursor records the underlying table, `fabrics`, in its data environment. When `REPORT FORM` runs, the data environment opens `fabrics` in its natural order and the report prints from that, not from your cursor. On our test machine the version without `NOFILTER` printed `CT-1001` first, the physical order of the table, even though the query asked for `ORDER BY fabric_id`. With `NOFILTER` the cursor is a real temporary file, `DBF()` returns a `.TMP` path, and the report prints `BL-4001` first, exactly as ordered.

Chapter 2 explains filtered cursors properly and makes `NOFILTER` or `READWRITE` part of every report query. Chapter 5 deals with report data environments in general. The lesson for now is simple: a report should print from the cursor you prepared, and you should know when it does not.

The quick report as a table

Because the FRX is a table, we can look inside the report we just generated. Listing a few fields of `fabstock.frx` shows the structure described earlier: a header record, three band records (page header, detail, page footer), one label and one field per column, and the date and page-number objects in the footer.

Command Window

```
USE C:\HollinsMill\Reports\fabstock.frx AGAIN SHARED NOUPDATE
LIST objtype, objcode, LEFT(expr, 20) FOR INLIST(objtype, 9, 5, 8) OFF
USE
```

You should see something like the following (the full list also shows font resource records, type 23, and the data environment records, types 25 and 26):

```
OBJTYPE OBJCODE LEFT(expr,20)
      9       1
      9       4
      9       7
      5       0 "Fabric_id"
      8       0 fabric_id
      5       0 "Descr"
      8       0 descr
...
      8       0 DATE()
      5       0 "Page "
      8       0 _PAGENO
```

Band `OBJCODE` 1 is the page header, 4 is the detail band, and 7 is the page footer. Labels store their text with quotes, because the `EXPR` field of a label is a string constant; fields store an expression that VFP evaluates for each record. That small distinction explains many puzzling reports: a "field" that prints its own name in quotes is a label, and a label that should change per row needs to become a field.

Where We Are Going

A quick report is a starting point, not a finished document. The first real report we will build is the customer list grouped by region. It has a title block, a shaded column heading strip, one group per region with a count of customers in that region, and a grand total of credit limits in the summary band. Chapter 3 builds it by hand in the Report Designer and shows the finished page as Figure 3.1; Chapter 6 and Chapter 7 explain the grouping and the counts.

The figures in this book are real VFP output. They were produced by running the canonical reports through a `ReportListener` and saving each page as a PNG image, a technique Chapter 19 explains in full. That matters for you as a reader: every number you see in a figure came from the same data and code you can run.

The plan of the book

The book is organized in six parts that follow the life of a report, from data to delivery.

1. **Part I — Getting Data Ready for Output** (Chapters 1–5). The data preparation layer, the Report Designer, report expressions and format pictures, data environments and data sessions, and running reports from code.
2. **Part II — The Report Designer in Depth** (Chapters 6–10). Grouping, report variables, Print When and Stretch rules, title and summary bands, multiple detail bands, multi-column reports, and labels.
3. **Part III — Business Documents** (Chapters 11–15). Invoices, packing slips, production lot reports, statements, aging, cross-tabs, images, and charts.
4. **Part IV — The ReportListener Architecture** (Chapters 16–20). Object-assisted output, the FFC listener classes, custom listeners, image output, and HTML and XML output.
5. **Part V — Printing and Export** (Chapters 21–24). Printers and printer environments, PDF without a native PDF engine, Excel and Word automation, and text, CSV, and fixed-width files.
6. **Part VI — Production Reporting** (Chapters 25–28). The FRX as a table, distribution and archiving, performance and troubleshooting, and a capstone month-end report pack for April 2026.

The book assumes you know basic VFP: tables, indexes, `SELECT - SQL`, procedures and functions, and the Command Window. When a language feature matters for output, we give a short refresher and move on.

The code that comes with the book

The Hollins Mill programs are small and each has one job. You will meet them chapter by chapter, but here is the map so you know where things live:

Program	Purpose	Introduced
`setup.prg`	Builds `hollins.dbc` and the sample data	Chapter 1

`hmlib.prg`	Shared helpers: money formatting, due dates, aging, invoice math	Chapter 2
`rptdata.prg`	One `Prep*()` function per report cursor	Chapter 2
`frxlib.prg`, `buildreports.prg`	Generate report layouts from code	Chapter 3, Chapter 25
`hmoutput.prg`	`RenderPng()`: report pages to PNG files	Chapter 19
`hmlisteners.prg`	Custom listeners: watermark and highlighting	Chapter 18

Later parts add programs for HTML, printing, PDF, Office export, auditing, distribution, and the month-end run. All of them follow the same conventions: Hungarian-style variable names such as `lcName` and `lnTotal`, parameters prefixed with `tc`, `tn`, `td`, and so on, and report cursors whose names start with `cur`.

A Working Session for the Rest of the Book

Most examples assume the same starting state: the database is open, the library programs are loaded, and the default folder is the program folder. A short session at the top of your test programs sets that up:

Command Window

```
SET DEFAULT TO C:\HollinsMill\Prog
SET CENTURY ON
SET DELETED ON
OPEN DATABASE C:\HollinsMill\Data\hollins SHARED
SET PROCEDURE TO hmlib, rptdata ADDITIVE
```

`SET CENTURY ON` makes dates print with four-digit years, which is how every date in this book is shown. `SET DELETED ON` hides deleted records, which every report should honor. The output in this book uses United States settings: dates as `MM/DD/YYYY` and money with a dollar sign, a comma for thousands, and a period for decimals. If your Windows regional settings or your `SET DATE`, `SET SEPARATOR`, and `SET POINT` commands differ, your dates and numbers will look different even though the values are the same.

> **NOTE**
>
> This book targets VFP 9 (SP2), the final release of Visual FoxPro. Microsoft announced in 2007 that there would be no version 10, and extended support ended in January 2015. The examples were verified on a VFP 9 installation; where a behavior might differ between service packs, the text says so.

Four Questions Before Every Report

When someone at Hollins Mill asks Theo for a new report, he asks four questions before he opens the designer. They are worth adopting, because the answers decide most of the design.

1. **Who reads it, and where?** A packing slip is read on a loading dock by someone holding a roll of fabric; it needs large type and no prices. An aging report is read at a desk by someone checking numbers; it needs aligned columns and totals that reconcile.
2. **What is one printed line?** An invoice prints one line per order line; an aging report prints one line per customer. The answer tells you the grain of the report cursor, which is the subject of Chapter 2.
3. **Where does it go after it is produced?** Paper, a PDF attachment, a web page, a spreadsheet, or another program. If the answer is "several of those", plan one data preparation step and several output steps.
4. **How will we know it is right?** Every business report has a number that must match something else: an invoice total must equal `invoices.total`, an aging total must equal invoiced minus received. Decide the check before you build the layout.

Common Mistakes

- **Running a report against whatever table happens to be selected.** A report with no data environment prints from the current work area. If the wrong alias is selected, the report runs happily against the wrong data. Prepare a cursor and select it, every time.
- **Trusting a quick report's data environment.** As we saw, a quick report from a filtered cursor can remember the underlying table and reopen it, discarding your order and filter.
- **Judging a report by its ASCII output.** Text output maps a graphical page onto a character grid. Columns shift and narrow fields lose characters. Use it for checks, not as the reference for what the printed page will look like.
- **Assuming both engines give identical pages.** Moving a report from `REPORTBEHAVIOR 80` to `90` can make text wrap or trim. Test both if your application can run either way.
- **Editing a production FRX directly.** Copy report files before you open them as tables or change them in the designer. An FRX has no undo once you close it.

Try It Yourself

1. **Rebuild the data.** Run `DO setup WITH "C:\HollinsMill\Data\"`, then run the row-count program and compare with the output above. *Solution:* six customers, eight fabrics, ten lots, twelve orders, 24 order lines, ten invoices, ten shipments, and six payments.
2. **A quick customer report.** Select `cust_id`, `company`, `region`, and `terms` from `customers` ordered by company into a cursor with `NOFILTER`, create a quick report from it, and write it to a text file. *Solution:* the first data line should be C1004, Ada Whitcombe Design, because companies