# JetsonPDF.Composition

## API Reference

*Built from scratch against ISO 32000-2 (PDF 2.0)*

This document was authored by JetsonPDF.Writer. The cover gradient, the chapter headings, the two-column tables, the navigable bookmarks and the page-number footer are all produced by the same writer the chapters describe. Use the Bookmarks pane to jump between sections.

```
Generated  2026-06-13 00:10 UTC
Producer   JetsonPDF.Writer
Format     PDF 1.7 with object & cross-reference streams
```

# Contents

One-line entries per feature area. Click an entry in the Bookmarks pane for direct navigation; this page is a flat human-readable index.

# Page-level assembly

JetsonPDF.Composition does two things: PageExtractor pulls a subset of pages out of a PDF into a new file, and Merger concatenates whole PDFs into one. Both are a lossless COS object-graph copy (§7.3) — a page's content streams, resources, fonts, images and annotations are copied in their original encoded form, never re-rendered. The package depends only on the Reader (not the Writer); both types are static and thread-safe.

| Type | Description |
|---|---|
| `JetsonPDF.Composition.PageExtractor` | Static. Copy chosen 1-based pages into a brand-new PDF. |
| `JetsonPDF.Composition.Merger` | Static. Concatenate whole PDFs into one, in order. |
| `Lossless COS copy` | Deep-copies each page's reachable object graph; refs remapped, shared resources deduped. |
| `Inherited attributes materialized` | /Resources, /MediaBox, /CropBox, /Rotate flattened onto each page (§7.7.3.4). |
| `Fresh file emitted` | New catalog, page tree, classic xref (§7.5.4), trailer with a new /ID. |
| `Output version` | Maximum of the source PDF versions. |

## Example

```
using JetsonPDF.Composition;

// Pull pages 1, 3 and 5 out of a report into a new PDF.
byte[] excerpt  = PageExtractor.Extract(reportBytes, 1, 3, 5);

// Concatenate three PDFs into one.
byte[] combined = Merger.Merge(coverBytes, bodyBytes, appendixBytes);
```

# Extracting pages

Page numbers are 1-based, and the output keeps them in the order you list — so the same call also reorders and duplicates pages. byte[], file-to-file and stream-to-stream overloads share one core; a password overload decrypts an encrypted source (the output is not encrypted). Streams are never closed by the call.

| Type | Description |
|---|---|
| `PageExtractor.Extract(byte[], params int[])` | 1-based page numbers; order preserved (reorder / duplicate). |
| `PageExtractor.Extract(byte[], string pwd, params int[])` | Decrypt an encrypted source, then extract. |
| `PageExtractor.Extract(string in, string out, ...)` | Read a file, extract, write a file (password overload too). |
| `PageExtractor.Extract(Stream in, Stream out, ...)` | Stream variant; neither stream is closed. |
| `PageExtractor.ExtractRange(byte[], first, last)` | Convenience: inclusive 1-based range first..last. |

## Example

```
using JetsonPDF.Composition;

byte[] picked   = PageExtractor.Extract(sourceBytes, 3, 1, 5); // reordered
byte[] chapter  = PageExtractor.ExtractRange(sourceBytes, 5, 12);
byte[] unlocked = PageExtractor.Extract(sourceBytes, "secret", 1, 2);

// File to file.
PageExtractor.Extract("report.pdf", "summary.pdf", 1, 2, 10);
```

# Merging documents

Merge concatenates every page of every source, in the order supplied, into a fresh page tree. byte[] (params and IEnumerable), file-list and stream-list overloads are provided; the output stream is written but not closed. Encrypted sources must be decrypted first — Merge has no password parameter and throws on a file it cannot read.

| Type | Description |
| --- | --- |
| `Merger.Merge(params byte[][])` | Concatenate in argument order; returns the merged bytes. |
| `Merger.Merge(IEnumerable<byte[]>)` | Concatenate a sequence of in-memory PDFs. |
| `Merger.Merge(IEnumerable<string>, string out)` | Read each path, merge, write the result to out. |
| `Merger.Merge(IEnumerable<Stream>, Stream out)` | Stream variant; the output stream is not closed. |
| `Decrypt before merge` | Extract an encrypted source with its password, then merge the bytes. |

## Example

```
using JetsonPDF.Composition;

byte[] combined = Merger.Merge(firstBytes, secondBytes, thirdBytes);

// Merge a whole folder in name order.
Merger.Merge(
    Directory.EnumerateFiles("chapters", "*.pdf").OrderBy(p => p),
    "book.pdf");
```

# What carries over

Document-level features that reference pages are merged across all sources, with cross-document name collisions disambiguated so nothing silently shares state. The structure tree, viewer preferences and page labels are not carried over; /Info comes from the source on extract and from the first document on merge.

| Type | Description |
| --- | --- |
| `Outlines / bookmarks` | Appended under one /Outlines root; dests remapped, dead bookmarks pruned, sibling links rebuilt. |
| `Named destinations` | Modern /Names /Dests tree + legacy /Dests dict merged; collisions suffixed (intro, intro_2). |
| `AcroForm fields` | Unified /Fields, merged /DR, OR'd /NeedAppearances + /SigFlags, concatenated /CO. |
| `Field-name collisions` | Top-level /T collisions suffixed (signature ? signature_2) so each field stays independent. |
| `Default-resource fonts` | Identical standard fonts shared; genuine clashes renamed and the referring /DA strings rewritten. |
| `Per-page annotations` | Links, widgets and markup copied with the page that owns them. |

## Example

```
using JetsonPDF.Composition;

// Two PDFs that both define a "signature" field merge into
// "signature" + "signature_2" — each keeps its own value.
byte[] combined = Merger.Merge(formA, formB);
```