JetsonPDF.Flow

API Reference

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

This document was authored by JetsonPDF.Flow itself. The cover band, running header, repeating chapter tables, named styles, automatic outline, and per-page footer are all produced by the same retained-mode tree the chapters describe. Use the Bookmarks pane to jump between sections.

```
Generated  2026-06-13 00:10 UTC
Producer   JetsonPDF.Flow
Pipeline   FlowDocument.Save
```

Contents

One entry per chapter, with the page number on the right. Click any entry to jump there; the same destinations also populate the Bookmarks pane.

1. Document, sections & page setup

FlowDocument is a mutable retained-mode tree. Add Sections; each Section is a flat list of block items that auto-paginate within its page setup. Title and Author flow into the resulting PDF /Info dictionary; Save serializes to disk or stream.

API surface

Type	Description
`JetsonPDF.Flow.FlowDocument`	Root document: Title/Author, default fonts, sections, styles.
`JetsonPDF.Flow.Section`	PageSize / PageMargins / Header / Footer / Body block list.
`JetsonPDF.Flow.Margins`	Margins(all) / Margins(h, v) / Margins.OneInch / Margins.Zero.
`doc.AddSection(configure?)`	Append a section and return it for further configuration.
`doc.DefaultRunProperties`	Cascades into every paragraph (font family, size, color).
`section.PageSize / PageMargins`	Per-section paper size and margins.
`section.ColumnCount / ColumnGap`	Multi-column body flow within a section.
`doc.Save() / Save(path | stream)`	Renders via the internal FlowRenderer; the no-arg overload returns a byte[].

Example

```
var doc = new FlowDocument
{
    Title  = "Q3 Quarterly Report",
    Author = "Acme Corp",
    DefaultRunProperties = new RunProperties
    {
        FontFamily = "Helvetica",
        FontSize   = 11,
        Color      = Color.Black,
    },
};
var cover = doc.AddSection();
cover.PageSize    = PageSize.Letter;
cover.PageMargins = new Margins(72);
var twoCol = doc.AddSection();
twoCol.PageSize    = PageSize.Letter;
twoCol.PageMargins = new Margins(72);
twoCol.ColumnCount = 2;
twoCol.ColumnGap   = 16;
doc.Save("report.pdf");
// …or serialize in-memory: byte[] bytes = doc.Save();
```

2. Paragraphs & runs

Paragraph is the bread-and-butter block: a sequence of Runs plus paragraph-level formatting. Two ergonomic ctors keep the common cases short: a string for a single TextRun, or a builder lambda that hands you a ParagraphBuilder.

API surface

Type	Description
`JetsonPDF.Flow.Paragraph`	Block: list of Runs + Properties + DefaultRunProperties.
`new Paragraph("text")`	One-shot single-run paragraph.
`new Paragraph(p => ...)`	Builder ctor: hands you a ParagraphBuilder.
`JetsonPDF.Flow.ParagraphBuilder`	AddText / AddPageNumber / AddTab / AddLineBreak / AddImage / ...
`JetsonPDF.Flow.Run`	Abstract base. TextRun is the workhorse.
`JetsonPDF.Flow.TextRun`	Plain text run with its own RunProperties.
`JetsonPDF.Flow.LineBreakRun`	Soft line-break inside a paragraph.
`JetsonPDF.Flow.TabRun`	Advances to the next TabStop > current X.
`JetsonPDF.Flow.PageNumberRun`	Resolves to current page index at render time.
`JetsonPDF.Flow.PageCountRun`	Resolves to total page count via two-pass renderer.
`JetsonPDF.Flow.PageReferenceRun`	Deferred reference to the page hosting a Bookmark.
`JetsonPDF.Flow.InlineImageRun`	Image inline with text; baseline-aligned.

Example

```
section.Body.Add(new Paragraph("This is the simplest paragraph."));
section.Body.Add(new Paragraph(p =>
{
    p.AddText("This report describes ");
    p.AddText("quarterly performance").Bold();
    p.AddText(" across our three primary business lines.");
    p.AddLineBreak();
    p.AddText("Period ending September 30, 2025.").Italic();
}));
section.Body.Add(new Paragraph(p =>
{
    p.AddText("Page ");
    p.AddPageNumber();
    p.AddText(" of ");
    p.AddPageCount();
}) { Alignment = TextAlignment.Center });
```

3. Character formatting (RunHandle)

Every AddX call on ParagraphBuilder returns a RunHandle - a chainable struct that tunes a single run's RunProperties. Bold / Italic / FontSize / FontFamily / Color / Underline / Strikethrough / Highlight, plus link, comment and tracked-changes helpers.

API surface

Type	Description
`JetsonPDF.Flow.RunHandle`	Return value of every ParagraphBuilder.Add* call.
`.Bold(value=true)`	Toggle bold on this run only.
`.Italic(value=true)`	Toggle italic.
`.FontSize(pt) / .FontFamily(name)`	Per-run font overrides.
`.Color(Color)`	Per-run text color.
`.Underline() / .Strikethrough()`	Underlined or struck-through run.
`.Highlight(Color)`	Background fill behind the run's glyphs.
`.Link(uri)`	Wrap the run in an external-URL hotspot.
`.SectionLink(anchor)`	Wrap the run in an internal-destination jump.
`.Comment(author, body)`	Anchor a review comment to this run (collected by CommentList).
`.AsInsertion(author?)`	Tracked-change insertion (underline + author colour).
`.AsDeletion(author?)`	Tracked-change deletion (strikethrough + author colour).
`JetsonPDF.Flow.RunProperties`	Nullable bag - unset entries inherit from paragraph then doc.

Example

```
section.Body.Add(new Paragraph(p =>
{
    p.AddText("Numbers are unaudited.")
     .Comment("Reviewer A", "Add audit timestamp before publish.");
    p.AddText(" See ");
    p.AddText("our methodology")
     .Link("https://www.example.com/methodology");
    p.AddText(" or jump to the ");
    p.AddText("appendix").SectionLink("appendix-a");
    p.AddText(".");
}));
section.Body.Add(new Paragraph(p =>
{
    p.AddText("Initial draft figure was ");
    p.AddText("$53.6M").AsDeletion("Editor");
    p.AddText(" ");
    p.AddText("$53.9M").AsInsertion("Editor");
    p.AddText(".");
}));
```

4. Paragraph formatting & pagination

ParagraphProperties carries the OOXML pPr concepts: alignment, line spacing, indents, before/after spacing, and the three pagination flags. The ergonomic pass-through accessors on Paragraph let you set these without poking at .Properties.

API surface

Type	Description
`JetsonPDF.Flow.ParagraphProperties`	Alignment / Indents / Spacing / LineSpacing / Pagination flags / TabStops.
`JetsonPDF.TextAlignment`	Left / Center / Right / Justify (shared with Page.DrawTextBlock).
`p.Alignment / SpaceBefore / SpaceAfter`	Pass-through accessors on Paragraph.
`p.LineSpacing`	Multiplier (1.2 = 120% of font size).
`p.LeftIndent / RightIndent`	Indents in points applied to every line.
`p.FirstLineIndent`	Extra indent on line one only (after LeftIndent).
`p.PageBreakBefore`	Force a fresh page before this paragraph.
`p.KeepLinesTogether`	Never split this paragraph across pages.
`p.KeepWithNext`	Keep this paragraph and its next sibling on the same page.
`p.TabStops.Add(new TabStop(...))`	TabStop(position, alignment): Left / Right / Center.

Example

```
section.Body.Add(new Paragraph("Introduction")
{
    Style = ParagraphStyle.Heading1,
});
section.Body.Add(new Paragraph("Body text wraps and justifies cleanly...")
{
    Alignment       = TextAlignment.Justify,
    LineSpacing     = 1.4,
    FirstLineIndent = 18,
    SpaceAfter      = 6,
});
var threeCol = new Paragraph(p =>
{
    p.AddText("Region"); p.AddTab();
    p.AddText("Q3");     p.AddTab();
    p.AddText("Notes");
});
threeCol.TabStops.Add(new TabStop(150));
threeCol.TabStops.Add(new TabStop(420, TabAlignment.Right));
section.Body.Add(threeCol);
```

5. Built-in & user-defined styles

Six built-in ParagraphStyle presets cover the everyday: Heading1/2/3, Body, Caption, Quote. NamedStyle lets you register reusable bundles with a basedOn graph - the cascade walks doc default ? style preset ? BasedOn chain ? paragraph ? run.

API surface

Type	Description
`JetsonPDF.Flow.ParagraphStyle`	Body / Heading1 / Heading2 / Heading3 / Caption / Quote.
`p.Style = ParagraphStyle.Heading1`	Apply a built-in preset to this paragraph.
`p.StyleName = "Callout"`	Reference a user-defined style (combines with Style).
`JetsonPDF.Flow.NamedStyle`	Reusable ParagraphProperties + RunProperties bundle.
`doc.AddStyle(name, basedOn?, ...)`	Register a style; chain via basedOn.
`doc.Styles[name]`	Lookup table - keyed case-insensitively.
`doc.AutomaticOutline`	Heading1/2/3 paragraphs are auto-bookmarked into the PDF outline.
`p.Bookmark = "intro"`	Manually register a named destination at the paragraph.

Example

```
doc.AddStyle("Callout", configure: s =>
{
    s.RunProperties.Italic   = true;
    s.RunProperties.FontSize = 10;
    s.RunProperties.Color    = Color.FromBytes(60, 60, 60);
    s.ParagraphProperties.LeftIndent  = 24;
    s.ParagraphProperties.RightIndent = 24;
});
doc.AddStyle("CalloutImportant", basedOn: "Callout", configure: s =>
{
    s.RunProperties.Bold  = true;
    s.RunProperties.Color = Color.FromBytes(180, 30, 30);
});
section.Body.Add(new Paragraph("Forward-looking statements may change.")
    { StyleName = "Callout" });
section.Body.Add(new Paragraph("Audit findings publish in November.")
    { StyleName = "CalloutImportant" });
section.Body.Add(new Paragraph("Introduction")
    { Style = ParagraphStyle.Heading1, Bookmark = "intro" });
```

6. Lists

Decorate a paragraph with a ListMarker and the renderer prepends an auto-numbered marker, tabs to a body indent, and configures a hanging indent so wrapped lines align under the marker's right edge. Counters reset between lists and across kinds.

API surface

Type	Description
`JetsonPDF.Flow.ListMarker`	Decoration: Kind, Level, GlyphOverride, RestartNumbering.
`JetsonPDF.Flow.ListKind`	Bullet / Number / LowerAlpha / UpperAlpha / LowerRoman / UpperRoman.
`p.ListMarker = new(ListKind.X)`	Make this paragraph a list item.
`new ListMarker(kind, level)`	Nested list level (counters per level + extra indent).
`marker.GlyphOverride = "?"`	Replace auto-generated marker text.
`marker.RestartNumbering = true`	Reset counter to 1 at this paragraph.

Example

```
section.Body.Add(new Paragraph("Highlights")
    { Style = ParagraphStyle.Heading2, KeepWithNext = true });
section.Body.Add(new Paragraph("Total revenue grew 12% YoY.")
    { ListMarker = new ListMarker(ListKind.Number) });
section.Body.Add(new Paragraph("Hardware margin expanded by 180 bps.")
    { ListMarker = new ListMarker(ListKind.Number) });
section.Body.Add(new Paragraph("Two new product lines launched.")
    { ListMarker = new ListMarker(ListKind.Number) });
// Nested sub-list (Roman, level 1):
section.Body.Add(new Paragraph("Cloud migration accelerator")
    { ListMarker = new ListMarker(ListKind.LowerRoman, level: 1) });
section.Body.Add(new Paragraph("Compliance reporting suite")
    { ListMarker = new ListMarker(ListKind.LowerRoman, level: 1) });
```

7. Tables

Block-level Table with weighted columns, repeating headers, optional borders and per-cell backgrounds. A cell's body is a list of BlockItems, so you can nest paragraphs, tables or images inside one cell. Column- and row-spans are first-class.

API surface

Type	Description
`JetsonPDF.Flow.Table`	Block: Columns weights, Header rows, Body rows, borders, padding.
`JetsonPDF.Flow.TableRow`	One row: list of TableCells.
`JetsonPDF.Flow.TableCell`	Children block list + ColumnSpan + RowSpan + Background.
`table.Columns.Add(weight)`	Relative-weight columns ({2.0, 1.0, 1.0} ? 50/25/25%).
`table.RepeatHeader`	Re-emit Header rows on each page the table spans.
`table.CellBorderWidth / Color`	Per-cell borders (zero suppresses).
`table.CellPadding`	Inner padding applied to every cell.
`cell.ColumnSpan / RowSpan`	Span across columns / rows.
`cell.Background`	Optional per-cell fill colour.

Example

```
var table = new Table
{
    Columns = { 2.0, 1.0, 1.0, 1.0 },
    RepeatHeader    = true,
    CellBorderWidth = 0.5,
    CellPadding     = 5,
};
var hdr = new TableRow();
hdr.Cells.Add(new TableCell(new Paragraph("Region")
    { Style = ParagraphStyle.Heading3 }));
hdr.Cells.Add(new TableCell(new Paragraph("Q2")
    { Style = ParagraphStyle.Heading3, Alignment = TextAlignment.Right }));
hdr.Cells.Add(new TableCell(new Paragraph("Q3")
    { Style = ParagraphStyle.Heading3, Alignment = TextAlignment.Right }));
hdr.Cells.Add(new TableCell(new Paragraph("?")
    { Style = ParagraphStyle.Heading3, Alignment = TextAlignment.Right }));
table.Header.Add(hdr);
foreach (var (region, q2, q3, delta) in regions)
{
    var row = new TableRow();
    row.Cells.Add(new TableCell(region));
    row.Cells.Add(new TableCell(new Paragraph(q2)
        { Alignment = TextAlignment.Right }));
    row.Cells.Add(new TableCell(new Paragraph(q3)
        { Alignment = TextAlignment.Right }));
    row.Cells.Add(new TableCell(new Paragraph(delta)
        { Alignment = TextAlignment.Right }));
    table.Body.Add(row);
}
section.Body.Add(table);
```

8. Headers, footers & page numbers

Each section has a Header and Footer - both are HeaderFooter holders of BlockItems. Use PageNumberRun and PageCountRun (via ParagraphBuilder.AddPageNumber / AddPageCount) to render running page numbers; a two-pass renderer resolves PageCount on the second pass.

API surface

Type	Description
`JetsonPDF.Flow.HeaderFooter`	Body list of BlockItems; one per side per section.
`section.Header.Body.Add(...)`	Append a paragraph to the running header.
`section.Footer.Body.Add(...)`	Append a paragraph to the running footer.
`p.AddPageNumber()`	Inline page number (1-based, current page).
`p.AddPageCount()`	Inline total page count - set on the second pass.
`p.AddPageReference("anchor")`	Page where a paragraph bookmarked anchor renders.

Example

```
section.Header.Body.Add(new Paragraph(p =>
    p.AddText("Acme Corp — Quarterly Report").Italic())
    { Alignment = TextAlignment.Right });
section.Footer.Body.Add(new Paragraph(p =>
{
    p.AddText("Page ");
    p.AddPageNumber();
    p.AddText(" of ");
    p.AddPageCount();
})
{ Alignment = TextAlignment.Center });
// Cross-reference: "see page 7" without hardcoding.
section.Body.Add(new Paragraph(p =>
{
    p.AddText("See ");
    p.AddText("the methodology section").SectionLink("appendix-a");
    p.AddText(" on page ");
    p.AddPageReference("appendix-a");
    p.AddText(".");
}));
```

9. Auto-generated lists: TOC, endnotes, comments

Three special block items expand at render time. TableOfContents collects every Heading1/2/3 paragraph into one entry per heading. EndnoteList collects every EndnoteRun into a numbered list. CommentList collects every commented run.

API surface

Type	Description
`JetsonPDF.Flow.TableOfContents`	Block: Title, MaxLevel (1..3), LevelIndent.
`JetsonPDF.Flow.EndnoteList`	Block: Title, TitleStyle. Renders one paragraph per endnote.
`JetsonPDF.Flow.CommentList`	Block: Title, TitleStyle. Renders one paragraph per comment.
`JetsonPDF.Flow.EndnoteRun`	Inline superscript marker; body collected by EndnoteList.
`JetsonPDF.Flow.FootnoteRun`	Inline superscript marker; body renders in the footnote area.
`section.FootnoteAreaHeight`	Reserve a strip at page bottom for FootnoteRun bodies.
`p.AddFootnote(text)`	Append a footnote run via the builder.
`p.AddEndnote(text)`	Append an endnote run via the builder.

Example

```
// ?? Reserve 70pt at the bottom of every body page for footnotes:
body.FootnoteAreaHeight = 70;
body.Body.Add(new Paragraph(p =>
{
    p.AddText("Numbers reflect a constant-currency basis");
    p.AddFootnote("Constant-currency rates are locked at the start of FY25.");
    p.AddText("; methodology in Appendix A");
    p.AddEndnote("EMEA recon re-run completed 2025-10-04.");
    p.AddText(".");
}));
// ?? Auto-generated tables in a later section:
appendix.Body.Add(new Paragraph("Contents")
    { Style = ParagraphStyle.Heading2, PageBreakBefore = true });
appendix.Body.Add(new TableOfContents { Title = null });
appendix.Body.Add(new Paragraph("Notes")
    { Style = ParagraphStyle.Heading2 });
appendix.Body.Add(new EndnoteList { Title = null });
appendix.Body.Add(new Paragraph("Review Comments")
    { Style = ParagraphStyle.Heading2 });
appendix.Body.Add(new CommentList { Title = null });
```

10. Decorations: drop caps, anchored & inline images

DropCap renders a paragraph's first character large at top-left with body wrap. AnchoredImage floats an image to one side of a paragraph and wraps text around it. BlockImage and InlineImageRun cover non-floating image insertion.

API surface

Type	Description
`JetsonPDF.Flow.DropCap`	Decoration: LineSpan (how many lines wrap around it).
`p.DropCap = new() { LineSpan = 3 }`	Apply the drop-cap decoration.
`JetsonPDF.Flow.AnchoredImage`	Floating image; body text wraps around it.
`JetsonPDF.Flow.AnchorSide`	Left or Right side anchor.
`p.FloatImage = new(image, w, h)`	Float an image at the paragraph's top-left/right.
`JetsonPDF.Flow.BlockImage`	Stand-alone image block (Width/Height/BlockAlignment).
`p.AddImage(image, w, h)`	Inline image baseline-aligned with the surrounding text.
`JetsonPDF.Flow.PageBreak`	Block that forces a fresh page (no content).

Example

```
var swatch = Image.FromFile("swatch.png");
section.Body.Add(new Paragraph(p =>
{
    p.AddText("This report describes quarterly performance ");
    p.AddText("across our three primary business lines: ");
    p.AddText("hardware, services, and software.");
})
{
    DropCap     = new DropCap { LineSpan = 3 },
    Alignment   = TextAlignment.Justify,
    FloatImage  = new AnchoredImage(swatch, width: 80, height: 60)
                  { Side = AnchorSide.Right },
});
section.Body.Add(new BlockImage(swatch)
{
    Width     = 240,
    Alignment = BlockAlignment.Center,
});
// Force a fresh page anywhere in the body:
section.Body.Add(new PageBreak());
```

11. Hyphenation

Set FlowDocument.Hyphenator to a Knuth-Liang Hyphenator and the renderer pre-processes every plain TextRun by inserting U+00AD soft hyphens at allowed breakpoints. The line breaker prefers them over hard char-breaks on overflow.

API surface

Type	Description
`JetsonPDF.Flow.Hyphenation.Hyphenator`	Liang-style pattern matcher; emits soft hyphens (U+00AD).
`Hyphenator.EnglishDefault()`	Starter set of common English suffixes; convenient for demos.
`h.AddPattern("1tion")`	Register one TeX-style pattern (digits encode priorities).
`h.AddPatterns(IEnumerable<string>)`	Bulk-register many patterns - load full ushyph.tex this way.
`h.MinPrefix / MinSuffix / MinWordLength`	Edge constraints (defaults: 2 / 3 / 5).
`h.HyphenateText(string)`	Walk a string and insert SHYs into letter substrings.
`doc.Hyphenator = ...`	Document-wide hyphenation toggle.

Example

```
doc.Hyphenator = JetsonPDF.Flow.Hyphenation.Hyphenator.EnglishDefault();
// For production-quality breaks, load full TeX patterns:
var custom = new JetsonPDF.Flow.Hyphenation.Hyphenator
{
    MinPrefix = 2,
    MinSuffix = 3,
};
custom.AddPatterns(File.ReadAllLines("ushyph.tex")
                       .Where(l => !l.StartsWith("%")));
doc.Hyphenator = custom;
```

12. Custom fonts

Standard 14 PDF fonts (Helvetica, Times-Roman, Courier and friends) work without registration. To embed your own TTF/OTF, set FlowDocument.ConfigureFonts and call RegisterFromFile / RegisterFromBytes on the supplied JetsonPDF.Fluent.FontRegistry.

API surface

Type	Description
`doc.ConfigureFonts`	Action<FontRegistry> invoked at render time.
`JetsonPDF.Fluent.FontRegistry`	Maps family names to TTF/OTF faces (regular/bold/italic/bold-italic).
`registry.RegisterFromFile(family, regularPath, ...)`	Register a family by file path.
`registry.RegisterFromBytes(family, regularBytes, ...)`	Register a family by in-memory bytes.
`RunProperties.FontFamily = "X"`	Reference a registered family from any run.

Example

```
doc.ConfigureFonts = registry =>
{
    registry.RegisterFromFile("Inter",
        regular:    @"C:\fonts\Inter-Regular.ttf",
        bold:       @"C:\fonts\Inter-Bold.ttf",
        italic:     @"C:\fonts\Inter-Italic.ttf",
        boldItalic: @"C:\fonts\Inter-BoldItalic.ttf");
};
doc.DefaultRunProperties = new RunProperties
{
    FontFamily = "Inter",
    FontSize   = 11,
};
section.Body.Add(new Paragraph(p =>
{
    p.AddText("Body in Inter; this run is ");
    p.AddText("bold").Bold();
    p.AddText(" and this one is ").Italic();
    p.AddText("italic.").Italic();
}));
```

13. Saving, round-tripping & inspection

FlowDocument.Save serializes via the internal FlowRenderer (a JetsonPDF.Fluent pipeline). The output is a normal PDF - reload it with JetsonPDF.Reading.Reader to inspect text items, outline entries, page count and so on, just as for any other PDF.

API surface

Type	Description
`doc.Save(path)`	Render to a file.
`doc.Save(stream)`	Render to any writable Stream.
`doc.AutomaticOutline`	Heading paragraphs auto-bookmark into the PDF outline.
`JetsonPDF.Reading.Reader.Load(stream)`	Reload the rendered bytes for verification.
`ReadDocument.Outlines`	Outline tree built from auto-bookmarked headings.
`ReadDocument.Pages.Count`	Final page count after pagination.

Example

```
doc.Save("FlowReport.pdf");
var bytes = File.ReadAllBytes("FlowReport.pdf");
var read  = JetsonPDF.Reading.Reader.Load(new MemoryStream(bytes));
Console.WriteLine($"Wrote FlowReport.pdf — " +
                  $"{bytes.Length:N0} bytes, " +
                  $"{read.Pages.Count} pages, " +
                  $"{doc.Sections.Count} sections.");
Console.WriteLine($"  Outline entries: {read.Outlines.Count}");
int textItemsPage1 = read.Pages[0].Items
    .OfType<JetsonPDF.Reading.PageTextItem>().Count();
Console.WriteLine($"  Page 1 text items: {textItemsPage1}");
```