

JetsonPDF.Fluent

API Reference

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

This document was authored by JetsonPDF.Fluent itself. The cover band, page header, repeating chapter tables and per-page footer are produced by the same fluent surface the chapters describe. Use the Bookmarks pane to jump between sections.

Generated 2026-04-30 17:10 UTC

Producer JetsonPDF.Fluent

Pipeline Document.Create -> GeneratePdf

Contents

One-line entries per feature area. Click an entry in the Bookmarks pane for direct navigation.

1. Document & pages ?
2. Layout containers ?
3. Text & rich text ?
4. Tables — headers, bodies & footers ?
5. Spacing & sizing ?
6. Alignment & positioning ?
7. Visual styling ?
8. Forms — AcroForm widgets ?
9. Navigation — links, sections & outlines ?
10. Pagination, layers & advanced ?

Document & pages

`Document.Create` takes a build callback that adds one or more `Page` blocks. Each page has five layered slots - `Background`, `Header`, `Content`, `Footer`, `Foreground` - drawn in that order. Content auto-paginates on overflow.

Type

`JetsonPDF.Fluent.Document`

`JetsonPDF.Fluent.IDocumentContainer`

`JetsonPDF.Fluent.IPageDescriptor`

`p.Size(PageSize) / Size(w,h)`

`p.Margin(all) / Margin(h,v)`

`p.Header() / Footer() / Content()`

`p.Background() / Foreground()`

`doc.WithMetadata(title, author)`

Description

Fluent entry point; `.Create` + `.GeneratePdf(path|stream)`.

Top-level builder; one `.Page()` call per page block.

Page builder: `Size`, `Margin`, `DefaultTextStyle`, `slots`.

Set page dimensions (defaults to Letter).

Inset around the inner page.

The three primary slots.

Behind / in front of the content for every paginated page.

Set Title/Author on the resulting PdfDocument.

Example

```
var doc = Document.Create(d =>
{
    d.Page(p =>
    {
        p.Size(PageSize.Letter);
        p.Margin(50);
        p.DefaultTextStyle(s => s.FontSize = 11);
        p.Header().Text("Quarterly Report",
            s => s.FontSize = 18);
        p.Content().Text("Body content goes here.");
        p.Footer().AlignCenter().Text("Confidential");
    });
});
doc.WithMetadata("Q1 Report", "Finance team")
    .GeneratePdf("report.pdf");
```

Layout containers

Containers compose vertically (Column), horizontally (Row), in z-order (Stack), in fixed-column flow (Grid) or in line-wrapping flow (Inlined). Row distinguishes ConstantItem (fixed pt) from RelativeItem (proportional).

Type

`c.Column(col => ...)`

`c.Row(row => ...)`

`c.Stack(stack => ...)`

`c.Grid(grid => ...)`

`c.Inlined(inl => ...)`

`col.Item() / stack.Item() /
grid.Item()`

`row.ConstantItem(width)`

`row.RelativeItem(weight=1)`

`col.Spacing(gap) / row.Spacing(gap)`

Description

Vertical flow with Spacing + Item().

Horizontal flow with ConstantItem / RelativeItem.

Z-order overlay - layers draw on top of one another.

Fixed-column grid (rows wrap when full).

Horizontal line-wrapping flow (mixed-width items).

Open a slot for one child.

Fixed-width slot in points.

Proportional slot of remaining width.

Gap between siblings (also on Inlined/Grid).

Example

```
p.Content().Column(col =>
{
    col.Spacing(8);
    col.Item().Row(row =>
    {
        row.RelativeItem(2).Text("INVOICE",
            s => s.FontSize = 24);
        row.ConstantItem(120).AlignRight()
            .Text("#" + invoiceNumber);
    });
    col.Item().LineHorizontal(1, Colors.Grey300);
    col.Item().Text("Body...");
});
```

Text & rich text

Three `.Text` overloads: a bare string, a string + style configurer, and a `TextDescriptor` builder for inline runs and embedded page numbers. Spans inherit from the descriptor's base `TextStyle` but can override `Bold/Italic/FontSize`.

Type

`c.Text(string)`

`c.Text(string, configure)`

`c.Text(td => td.Span(...).Bold())`

`JetsonPDF.Fluent.TextStyle`

`JetsonPDF.Fluent.TextDescriptor`

`JetsonPDF.Fluent.TextSpanDescriptor`

`c.PageNumber()` / `c.TotalPages()`

Description

Quick text with the default style.

Text with a one-off style override.

Rich text with mixed inline runs.

FontFamily / FontStyle / FontSize / FontColor / LineHeight.

Builder for rich text; Span / CurrentPageNumber / TotalPages.

Per-run style: `.Bold` / `.Italic` / `.FontSize` / `.FontColor`.

Stand-alone page-number leaves (no surrounding text).

Example

```
p.Footer().AlignCenter().Text(t =>
{
    t.Span("Page ").FontColor(Colors.Grey700);
    t.CurrentPageNumber().Bold();
    t.Span(" of ").FontColor(Colors.Grey700);
    t.TotalPages().Bold();
});
p.Content().Text("Bold heading", s =>
{
    s.FontSize = 18;
    s.FontStyle = FontStyle.Bold;
    s.FontColor = Colors.IndigoDark;
});
```

Tables — headers, bodies & footers

Tables are built from a ColumnsDefinition plus optional Header/Footer plus body Cells. Header repeats on every page; Footer renders only on the final page; the body auto-paginates at row boundaries. columnSpan and rowSpan are first-class.

Type	Description
<code>c.Table(t => ...)</code>	Build a table from columns + cells.
<code>t.ColumnsDefinition(cd => ...)</code>	Declare relative or constant columns.
<code>cd.RelativeColumn(weight=1)</code>	Proportional column.
<code>cd.ConstantColumn(width)</code>	Fixed-width column in points.
<code>t.Header(h => h.Cell()...)</code>	Repeating header (drawn on every page).
<code>t.Footer(f => f.Cell()...)</code>	Footer (drawn on the final page only).
<code>t.Cell(columnSpan, rowSpan)</code>	Body cell; spans flow auto-occupancy.

Example

```
p.Content().Table(t =>
{
    t.ColumnsDefinition(cd =>
    {
        cd.RelativeColumn(3);
        cd.RelativeColumn(1);
        cd.RelativeColumn(1);
    });
    t.Header(h =>
    {
        h.Cell().Text("Item");
        h.Cell().AlignRight().Text("Qty");
        h.Cell().AlignRight().Text("Price");
    });
    foreach (var (item, qty, price) in lineItems)
    {
        t.Cell().Text(item);
        t.Cell().AlignRight().Text(qty.ToString());
        t.Cell().AlignRight().Text(price.ToString("C"));
    }
    t.Footer(f =>
    {
        f.Cell(columnSpan: 2).AlignRight().Text("Total");
        f.Cell().AlignRight().Text(total.ToString("C"));
    });
});
```

Spacing & sizing

Padding, Width, Height and the Min/Max bounds are decorators - they wrap a slot and reshape it before the child measures itself. Width / Height pin a dimension exactly; Min/Max set bounds. PaddingHorizontal / PaddingVertical for symmetry.

Type	Description
<code>c.Padding(all)</code>	Equal pad on all four sides.
<code>c.Padding(horizontal, vertical)</code>	Two-axis pad.
<code>c.PaddingLeft / Right / Top / Bottom</code>	One side at a time.
<code>c.PaddingHorizontal / Vertical</code>	Both sides on one axis.
<code>c.Width(pt) / c.Height(pt)</code>	Pin the dimension exactly.
<code>c.MinWidth / MaxWidth</code>	Bound horizontal size.
<code>c.MinHeight / MaxHeight</code>	Bound vertical size.

Example

```
p.Content().Column(col =>
{
    col.Spacing(12);
    col.Item().Padding(20).Background(Colors.Grey50)
        .Text("Padded callout block.");
    col.Item().Height(80).Border(1, Colors.Grey400)
        .AlignCenter().AlignMiddle()
        .Text("Fixed-height box");
    col.Item().PaddingHorizontal(40)
        .Text("Long-form paragraph with " +
            "horizontal breathing room only...");
});
```

Alignment & positioning

Align decorators claim the available slot and align the child within it. Translate shifts a slot by an offset; Rotate spins it (CTM); AspectRatio constrains width/height to a ratio. Applied in chain order.

Type	Description
<code>c.AlignLeft / AlignCenter / AlignRight</code>	Horizontal alignment within the slot.
<code>c.AlignTop / AlignMiddle / AlignBottom</code>	Vertical alignment within the slot.
<code>c.Translate(dx, dy)</code>	Translate the child by an offset.
<code>c.Rotate(degrees)</code>	Rotate the child via CTM (about its origin).
<code>c.AspectRatio(ratio, mode)</code>	Constrain w/h to a ratio (Fit / Fill / Stretch).

Example

```
p.Foreground().AlignTop().AlignRight().Padding(10)
    .Text("DRAFT", s =>
        {
            s.FontSize = 32;
            s.FontColor = Colors.Red;
            s.FontStyle = FontStyle.Bold;
        });
p.Content().AlignCenter().Width(360).Rotate(-3)
    .Background(Colors.AmberDark).Padding(20)
    .Text("Stamped & spinning");
```

Visual styling

Border, Background and Shadow are simple decorators on top of any slot. Pair with the Colors palette - a Material-flavoured set of named hues plus Colors.FromHex for arbitrary brand colours.

Type

`c.Border(width, color = null)`

`c.Background(color)`

`c.Shadow(dx, dy, color)`

`JetsonPDF.Fluent.Colors`

`Colors.FromHex("#1A2B3C")`

Description

Stroke a rectangle around the slot.

Solid fill behind the slot's child.

Drop-shadow rectangle behind the slot.

Material palette (Red/Pink/.../Brown + Greys + Light/Dark).

Parse an arbitrary hex colour.

Example

```
p.Content().Background(Colors.IndigoLight)
    .Border(1, Colors.IndigoDark)
    .Shadow(2, 2, Colors.Grey400)
    .Padding(16)
    .Text("Branded callout", s =>
    {
        s.FontStyle = FontStyle.Bold;
        s.FontColor = Colors.Grey900;
    });
```

Forms — AcroForm widgets

AsTextField / AsCheckBox / AsComboBox / AsListBox / AsPushButton attach an AcroForm widget to the slot's measured rectangle. The widget consumes the slot as a leaf; Acrobat draws its native chrome at view-time.

Type

`c.AsTextField(name, configure?)`

`c.AsCheckBox(name, configure?)`

`c.AsComboBox(name, opts, configure?)`

`c.AsListBox(name, opts, configure?)`

`c.AsPushButton(name, caption, configure?)`

Description

Single- or multi-line text field.

Toggle widget (named On state).

Drop-down (editable or fixed).

Multi-select list.

Action button (configure to set Action).

Example

```
p.Content().Column(col =>
{
    col.Spacing(10);
    col.Item().Row(row =>
    {
        row.ConstantItem(120).AlignMiddle().Text("Name:");
        row.RelativeItem().Height(22)
            .AsTextField("name", t => t.MaxLength = 64);
    });
    col.Item().Row(row =>
    {
        row.ConstantItem(20).Height(20)
            .AsCheckBox("subscribe");
        row.ConstantItem(8);
        row.RelativeItem().AlignMiddle().Text("Subscribe");
    });
    col.Item().Width(120).Height(28)
        .AsPushButton("submit", "Submit");
});
```

Navigation — links, sections & outlines

Link wraps a slot in an external URL hotspot; Section registers a named destination at the slot's top-left; SectionLink jumps to one. Pair with WithOutline to give the bookmarks pane a tree.

Type

`c.Link(uri)`

`c.Section(anchor)`

`c.SectionLink(anchor)`

`doc.WithOutline(b => b.Item(...))`

`JetsonPDF.Fluent.OutlineBuilder`

`doc.WithPageLabels(...)`

Description

External-URL hotspot over the slot.

Register a named destination at this point.

Internal jump to a named destination.

Build the bookmarks tree against named destinations.

Tree builder; Item returns an OutlineEntryBuilder for nesting.

Roman numerals / letter prefixes for the page-number bar.

Example

```
Document.Create(d =>
{
    d.Page(p =>
    {
        p.Content().Column(col =>
        {
            col.Item().Section("intro").Text("Introduction",
                s => s.FontSize = 22);
            col.Item().Text("Body...");
            col.Item().Section("toc").Text("Table of Contents",
                s => s.FontSize = 22);
            col.Item().SectionLink("intro")
                .Text("Jump to Introduction",
                    s => s.FontColor = Colors.BlueDark);
            col.Item().Link("https://www.iso.org")
                .Text("ISO website");
        });
    });
})
.WithOutline(o =>
{
    o.Item("Introduction", "intro");
    o.Item("Table of Contents", "toc");
});
```

Pagination, layers & advanced

Pagination decorators control how slots react to page breaks. Layer assigns the child to an optional-content group. Component yields an IDynamicComponent that composes per page (knows the current page number). Canvas is the escape hatch.

Type	Description
<code>c.ShowOnce()</code>	Render once - skip on subsequent pages of the same block.
<code>c.SkipOnce()</code>	Skip on the first page; render thereafter.
<code>c.ShowEntire()</code>	Treat as atomic - never split across pages.
<code>c.EnsureSpace(minHeight)</code>	Page-break before if less than minHeight remains.
<code>c.Layer(handle)</code>	Put content in an optional-content group (layer).
<code>c.Component(IDynamicComponent)</code>	Per-page composer (sees CurrentPageNumber/TotalPages).
<code>c.Canvas((surface,w,h) => ...)</code>	Drop to a low-level IDrawingSurface for custom painting.
<code>doc.WithLayer(name, visible?, intent?)</code>	Register an OCG and get a LayerHandle.

Example

```
var watermark = doc.WithLayer("Watermark", visibleByDefault: true);
Document.Create(d =>
{
    d.Page(p =>
    {
        p.Foreground().Layer(watermark).AlignCenter().AlignMiddle()
            .Text("CONFIDENTIAL", s =>
            {
                s.FontSize = 60;
                s.FontColor = Colors.RedLight;
            });
        p.Content().Column(col =>
        {
            col.Item().ShowOnce().Text("First-page intro");
            col.Item().ShowEntire().Padding(8)
                .Background(Colors.Grey100)
                .Text("Don't split me across pages.");
            col.Item().Canvas((surface, w, h) =>
            {
                surface.DrawRectangle(0, 0, w, h,
                    fill: PdfColor.Rgb(0.95, 0.95, 1.0));
            });
        });
    });
});
```