JetsonPDF.Forms

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.

Opening a form & discovering fields

JetsonPDF.Forms opens an existing PDF, exposes its AcroForm fields, queues mutations against them, and saves via a single incremental-update layer. Within one Form instance repeated Save() calls are byte-identical: mutations are re-applied against the original source bytes, never stacked on a prior save.

Type	Description
`JetsonPDF.Forms.Form`	Entry point. Open a PDF, enumerate Fields, queue mutations, Save.
`Form.Open(string, password = "")`	Open from disk. Optional owner/user password.
`Form.Open(byte[] | Stream, ...)`	Open from a buffer or stream (read fully into memory).
`form.Fields`	FormFieldCollection of every discovered field, stable order.
`JetsonPDF.Forms.FormFieldCollection`	Count, this[name], Contains, TryGet, IEnumerable<FormField>.
`collection["Name"]`	O(1) lookup by fully-qualified field name; throws if absent.
`form.Dispose()`	Releases the underlying parsed document (use 'using').

Example

```
using var form = Form.Open("input.pdf");

foreach (var f in form.Fields)
    Console.WriteLine($"{f.Name}: {f.Type} = {f.Value}");

Console.WriteLine($"{form.Fields.Count} field(s)");
```

Inspecting a field

FormField is one fully-qualified field name. A radio group is a single field whose SelectRadio targets the chosen option. Read-only properties describe the field as parsed; Value reflects any mutation already queued this session.

Type	Description
`JetsonPDF.Forms.FormField`	One AcroForm field; fluent setters return the same instance.
`field.Name`	Fully-qualified field name (period-joined, §12.7.4.2).
`field.Type`	FormFieldType — controls which setters are valid.
`field.Value`	Current value; honours a pending mutation if one is queued.
`field.Options`	(Export, Display) pairs for combo / list boxes; else empty.
`field.MaxLength`	/MaxLen for text fields, or null.
`field.IsReadOnly / IsRequired`	Source /Ff ReadOnly and Required flag bits.
`field.Rect / PageIndex`	First widget's rect (points, y-up) and page index.
`field.IsSigned`	True for a signature widget with a populated /V dict.
`JetsonPDF.Forms.FormFieldType`	Text, CheckBox, Radio, PushButton, ComboBox, ListBox, Signature, Unknown.

Example

```
var field = form.Fields["Photo"];
Console.WriteLine($"{field.Type} @ page {field.PageIndex} " +
                  $"rect {field.Rect}");

if (field.Type == FormFieldType.ComboBox)
    foreach (var (export, display) in field.Options)
        Console.WriteLine($"  {export} = {display}");
```

Mutating fields

Every setter is fluent (returns the FormField) and type-checked: a wrong-type call throws. Push buttons hold no value; signed signature widgets reject every mutation because the cryptographic byte range covers /AP. SetImage is valid on text, push-button, and unsigned signature widgets.

Type	Description
`field.SetText(value)`	Text fields and the editable text of a combo box.
`field.SetChecked(bool) / Toggle()`	Check-box on/off; Toggle reads the current state.
`field.SelectRadio(optionName)`	Radio group — name matches an /AP /N on-state key.
`field.SetChoice(export)`	Combo / single-select list box by export value.
`field.SetChoices(exports)`	Multi-select list box; throws if single-select.
`field.SetImage(image, alignment, ...)`	Bake an image into /AP /N at the widget rect.
`JetsonPDF.Forms.ImageAlignment`	Center (default), TopLeft/TopRight/BottomLeft/BottomRight, Stretch.
`field.Clear()`	Reset one field; button/signature silently skipped.

Example

```
form.Fields["FullName"].SetText("John Smith");
form.Fields["Subscribe"].SetChecked(true);
form.Fields["Plan"].SetChoice("annual");
form.Fields["Photo"].SetImage(
    JetsonPDF.Image.FromFile("sig.png"),
    ImageAlignment.Center, padding: 2);
```

Saving & resetting

Save serializes original bytes plus exactly one incremental-update section (§7.5.6). SingleLayer is the only mode and the default; it is the one to use for signed documents. Form.Clear() queues a reset for every value-bearing field.

Type	Description
`form.Save()`	Serialize to a byte[] (SaveMode.SingleLayer).
`form.Save(string | Stream)`	Write the bytes to a path or stream.
`form.Save(SaveMode)`	Explicit mode; SingleLayer is the only member.
`JetsonPDF.Forms.SaveMode.SingleLayer`	Original bytes + one incremental section; byte-identical on re-save.
`form.Clear()`	Queue a clear for every text/check/radio/choice field.

Example

```
using var form = Form.Open("input.pdf");
form.Fields["FullName"].SetText("Jordan");
form.Fields["IsActive"].SetChecked(true);
form.Save("output.pdf");

// Repeated saves within one instance are byte-identical:
var a = form.Save();
var b = form.Save();
Console.WriteLine(a.SequenceEqual(b)); // True
```